
# UNIVERSITÀ DEGLI STUDI DI PADOVA

**Dipartimento di Fisica e Astronomia "Galileo Galilei"**

**Master Degree in Physics of Data**

**Final Dissertation**

## Recurrent Neural Network approaches to model primate behavior

**Thesis supervisor**

**Prof. Samir Simon Suweis**

**Thesis co-supervisor**

**Prof. Michele Allegra**

**Candidate**

**Kiamehr Javid**

**2025-2026**

# Abstract

Recurrent Neural Networks (RNNs) are increasingly used in theoretical neuroscience to model simple animal behavior. In this context, two conceptually different approaches have been taken: in the first approach, the RNN is trained on neural activity recordings of the animal during a task, to infer the structure of a recurrent circuit giving rise to the observed activity patterns; in the second approach, the RNN is trained on stimulus-response associations of a given task, to infer how a recurrent circuit might implement the related mapping; In this thesis, we start from previous work that trained RNNs on neural activity recordings of a monkey during a simple contextual decision making task, and train RNNs only on the corresponding stimulus-response association. The structure and dynamics of the trained RNNs are analyzed and compared with previous results,with a particular focus on how biological constraints (e.g., the implementation of Dale's principle and the necessity of distributed processing ) affect RNN behavior and its agreement with previous findings.

# Contents

# Chapter 1

# Introduction

This thesis is situated within the field of machine learning applied to neuroscience, where computational models are used as analytical tools for studying biological neural systems. Since the emergence of artificial neural networks, an important question has been how these models relate to real neural activity. More specifically, researchers have asked to what extent neural recordings obtained from living systems may exhibit similarities in structure or temporal dynamics to trained artificial networks, and whether such similarities can provide insight into the principles of neural computation. Rather than treating artificial networks only as engineering tools, this perspective considers them as simplified systems that can be examined in relation to biological data.

The motivation for this kind of work does not arise only from immediate practical applications, but also from scientific curiosity. Biological brains remain difficult to understand because they produce remarkably complex behavior through processes that are still only partially observable. Artificial neural networks, while far simpler, offer a rare opportunity to isolate internal mechanisms, inspect hidden activity, and systematically alter the rules that shape learning. Comparing these two systems creates a space for exploration: by trying different computational ideas and observing what emerges, it becomes possible to ask whether some aspects of biological computation can be approximated, reproduced, or better understood through artificial models.

Although artificial neural networks are increasingly used in neuroscience, there are still relatively few studies that directly compare multiple training strategies against biological recordings within the same framework. Many existing studies either train networks to solve behavioral tasks by replicating stimulus-response association or train networks to reproduce measured neural activity, but fewer

works examine these approaches side by side. This leaves open the question of whether networks that merely perform a task develop internal solutions similar to those found in biological systems, or whether closer correspondence requires training procedures that explicitly reproduce recorded neural responses. Exploring these differences can help clarify how strongly biological structure depends on task demands or biological restrictions.

The present work compares recurrent neural networks trained under different objectives. One class of networks is trained to generate the correct response from a given stimulus, while another is trained to reproduce recorded neural activity more directly. Additional unconstrained or baseline models are included to establish a reference for the comparison. Rather than assuming a single correct modeling strategy, this study examines multiple possible paths from input to output and asks how each training approach shapes the resulting network. The analysis focuses on whether different optimization procedures produce comparable internal organization, and whether any of these artificial systems show meaningful relationships to the activity observed in the biological recordings.

A biological neural system can be represented in simplified form by a recurrent neural network, making this model a natural choice for the present study. In particular, a vanilla leaky RNN provides a flexible yet interpretable architecture with input signals, output signals, and an evolving hidden state. Because neural activity unfolds over time rather than instantaneously, recurrent models are especially suitable for capturing temporal dependencies that static models cannot represent. Their relatively simple structure also allows the internal state of the model to remain accessible for analysis, making it possible to compare not only the final outputs of the network but also the internal patterns that emerge during learning.

The contribution of this thesis is not to propose a completely new theory, but to provide another careful comparison between artificial and biological neural systems while emphasizing biologically motivated constraints. In particular, the study examines whether introducing constraints inspired by neural organization can influence how closely an artificial network resembles recorded neural activity. These comparisons may involve both static properties, such as the clustering seen in the final weight matrices, and dynamic properties helping us understand the flow of information in the network. Even a modest result can contribute to the broader effort of understanding which aspects of neural computation arise from optimization alone and which may depend on additional biological structure.

# Chapter 2

# Background

## 2.1 Artificial Neural Networks

This section introduces the core ideas and mechanisms of an artificial neural network (ANN). Readers already familiar with the subject may wish to skip it.

Originally inspired by neural biological tissue, ANNs are computational models trained to learn highly complex, multi-dimensional functions from a set of input data. We assume that some common pattern exists and is mapped to the desired output. The idea is that the model will learn the properties of the data and can be used to predict the behavior of new unseen data. In order to do so, the model would need to have some way of recognizing patterns in its input. Now suppose that this potentially complex map is approximated via numerous units performing simple computations, expressed as links between these units. This machine, which is capable of digesting some input and producing the corresponding output, is trained via adjusting and improving the connections slightly each time that it makes a mistake. we will dive deeper into a certain type of ANN, called a Recurrent Neural Network (RNN).

An RNN is most suitable for sequences and specially temporal data due to its nature. The simplest form of an RNN, often called a simple RNN, vanilla RNN, or Elman RNN is consisted of the following:

- Nodes: input layer $\mathbf{u} \in \mathbb{R}^{d^{in}}$, recurrent layer (hidden) $\mathbf{r} \in \mathbb{R}^{d^{rec}}$, and output layer $\mathbf{o} \in \mathbb{R}^{d^{out}}$.
- Weights: input $W^{in} \in \mathbb{R}^{d^{in} \times d^{rec}}$, recurrent $W^{rec} \in \mathbb{R}^{d^{rec} \times d^{rec}}$, and output $W^{out} \in \mathbb{R}^{d^{rec} \times d^{out}}$.
- Biases: recurrent $b^{rec} \in \mathbb{R}^{d^{rec}}$ and output $b^{out} \in \mathbb{R}^{d^{out}}$.

A time series compatible with the shape of the RNN's input layer is fed to the network, continuously evolving the recurrent layer [2]. The discrete time equation for the recurrent layer is as follows:

$$r^{(t+1)} = f(W^{rec}r^{(t)} + W^{in}u^{(t+1)} + b^{rec}) \quad (f \text{ applied element-wise}) \tag{2.1}$$

where the argument is called the activation and $f$ itself is called the activation function, adding non-linearity and hence more complexity to the model. What equation (2.1) expresses is that the state of the recurrent nodes at some time step $t+1$, depends on both the new inputs $(t+1)$ and the nodes at the previous step $(t)$.

The time continuous version of equation (2.1) with gaussian noise introduced is:

$$\tau\frac{d\mathbf{r}}{dt} = f(W^{rec}\mathbf{r} + W^{in}\mathbf{u} + \mathbf{b}^{rec} + \sqrt{2\tau\sigma^2}\xi) - \mathbf{r} \tag{2.2}$$

Here $\tau$ is the time constant, a property of the system being studied. For example, the neuronal time constant for certain neuron types in mammals is of the order of tens of milliseconds[8]. $\xi$ are $N_r$ noise variables sampled from $\mathcal{N}(0,1)$.

Approximating the hidden layer using equation (2.2) at time $t+\Delta t$ with $\Delta t < \tau$ results in:

$$\mathbf{r}^{(t+\Delta t)} = \alpha f(W^{rec}\mathbf{r}^{(t)} + W^{in}\mathbf{u} + \mathbf{b}^{rec} + \sqrt{2\tau\sigma^2}\xi) + (1-\alpha)\mathbf{r}^{(t)} \tag{2.3}$$

Where $\alpha = \frac{\Delta t}{\tau}$. Having $\tau$ as the neuron's memory time scale, simulating the system at some step $\Delta t$ leads to the leakage term. Equation (2.3) shows how we translate the continuous time RNN equation into the simulation work frame, considering the time scales of the system. This gives us a hyperparameter to tune with respect to the system being studied.

The outputs depend only on the hidden state and the output biases. Note that the RNN produces an output at each time step, having seen the inputs up to that step. Although one might only be interested in the final output, this is the main property that makes RNNs naturally a good choice for studying timeseries data or simulating time-variant systems.

$$\mathbf{o}^{(t)} = g(W^{out}\mathbf{r}^{(t)} + b^{out}) \quad (g \text{ applied element-wise}) \tag{2.4}$$

With $g(\cdot)$ being the activation function of the output layer. Feeding the inputs and computing the neuron values $mathbfo^{(t)}$ up until the final time $T$ is regarded as the forward pass.

## 2.2 Training

Consider an untrained network only capable of mapping the input to some output. This output being from our desired target means that we need some measure for quantifying the distance. This disparity is called the loss or the cost and the measure quantifying it, the loss function, could be something as simple as the euclidean distance or something more complex, not necessarily a formal distance. In order to train the weights of this network, one would calculate the share of each weight, the gradient of the loss, for this disparity and adjust them in such a way that minimizes the loss. An efficient and also the most common method for calculating this gradient is called the backpropagation method.

Let us denote the target output at time $t$ by $\hat{\mathbf{o}}^{(t)}$ and the mistake to be penalized, as some function of the target output and the RNN output:

$$L^{(t)} = d(\mathbf{o}^{(t)}, \hat{\mathbf{o}}^{(t)}) \tag{2.5}$$

$$L = \sum_{t=1}^{T} L^{(t)} \tag{2.6}$$

Let us also define:

$$\tilde{d}(\mathbf{o}^{(t)}) = d(\mathbf{o}^{(t)}, \hat{\mathbf{o}}^{(t)}) \tag{2.7}$$

We are ultimately interested in gradients such as $\nabla_{W^{in}} L$ or others corresponding to the rest of the trainable parameters (weights and biases). Note that some of these gradients are matrices.

Finding some intermediate derivatives first proves to be helpful. For example, we know that the hidden state at step t $\mathbf{r}^{(t)}$'s contribution to the loss can be broken down into two parts:

$$\nabla_{\mathbf{r}^{(t)}} L = \frac{\partial \mathbf{r}^{(t+1)}}{\partial \mathbf{r}^{(t)}} \nabla_{\mathbf{r}^{(t+1)}} L + \frac{\partial o^{(t)}}{\partial \mathbf{r}^{(t)}} \nabla_{o^{(t)}} L \tag{2.8}$$

This is where an RNN differs in training from a Feed-Forward Neural Network(FFNN); The backpropagation through time(BPTT) is often expressed as normal backpropagation with the unfolded network, and equation (2.8) is where the unfolding occurs, since it shows how the loss $L^{(t)}$ can depend on step $t$ and before, or $\mathbf{r}^{(t)}$ affects $L^{(t)}$ and successive losses.

The first term in equation (2.8) vanishes for $t = T$, depending solely on the final output. Using (2.1), the first term propagates through time via $\frac{\partial \mathbf{r}^{(t+1)}}{\partial \mathbf{r}^{(t)}}$. This depends on $W^{rec}$, and $f'$ evaluated at the present activation. From equations (2.6) and (2.1), we notice that the second term depends on the derivatives of $g$ and $\tilde{d}$ together with $W^{out}$. Finally, for each recurrent weight, we have:

$$\frac{\partial L}{\partial W_{ij}^{rec}} = \sum_{t=1}^{T} \frac{\partial \mathbf{r}^{(t)}}{\partial W_{ij}^{rec}} \cdot \nabla_{\mathbf{r}^{(t)}} L \tag{2.9}$$

The loss function and activation functions are usually chosen to have a simple derivative. The key takeaway is that all the material required for the computation of the gradient $\nabla_{W^{in}} L$ and the evaluation of the derivatives is already known from the forward pass. The same idea holds for the rest of the trainable parameters, making the backpropagation very efficient.

## 2.3 Biological Brain

Although recurrence captures the nature of a biological brain to some extent, a brain works differently than an RNN in many aspects. In this section, we will review some of the preliminary concepts and definitions concerning the nervous system and try to draw comparisons and analogies between an RNN and the nervous system.

### 2.3.1 Nervous system

The nervous system in all vertebrates and even some invertebrates has a clear division between the Central Nervous System(CNS) and the Peripheral Nervous system(PNS). The CNS is simply the collection of brain and spinal cord, and the PNS includes nervous tissue outside the CNS [8, Ch 1]. This partitioning immediately gives rise to a categorization of neurons, since direction becomes important:

- Afferent(sensory): Neurons that take information towards the CNS.
- Efferent(motor): Neurons that take information away from the CNS.
- Interneurons: Mainly located in the CNS and participate in local circuits.

This functional classification does not automatically put these cells under the anatomical categories of central or peripheral. Afferent or efferent neurons are

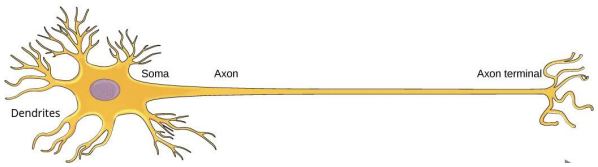



Figure 2.1: Neuron Structure [1]

in charge of carrying signals, often for long distances, letting some grow up to meters long.

Interneurons, on the other hand, form local feedback loops through which the information flow is regulated. This requires them to have different connectivity properties. The number of processes, any extension that emerges from the soma, varies greatly in different neurons according to their types.

Despite being poor electrical conductors, Neurons transmit information via electrical signals by regulating ion flow across their membranes. Neurons maintain a negative internal potential relative to their extracellular potential, known as the resting membrane potential. This signal transmission may be within a single neuron or between two neurons. Figure 2.1 shows a simple diagram of a neuron.

### 2.3.2 Signal transmission

A neuron maintains a local potential difference across its membrane while at rest, keeping it polarized. This trans-membrane potential is often very close to -70 mV. This is done by balancing the concentration of ions such as $Na^+$ or $K^+$ through dedicated channels.

A local disturbance of this voltage may propagate along the axon, given that it is large enough and having the neuron at rest. Small disturbances that fail to reach the threshold around -55 mV die out, but for a larger increase, the sodium gates react by allowing for depolarization, and this triggers a spike in the potential. The spike moves rapidly along the cell, but the local voltage has to go through a full cycle before another signal can be transmitted. Figure 2.2 shows how the voltage changes. Gates will stop responding to any stimulus regardless of its amplitude after depolarization until a few milliseconds later, this is called the refractory period which imposes an limit on how fast a neuron can shoot. The refractory period can be less than 1 millisecond or greater than 10 milliseconds. Note that the action potential always follows the same profile and does not vary in amplitude depending on the stimulus strength. Higher neuron activity is simply higher frequency.

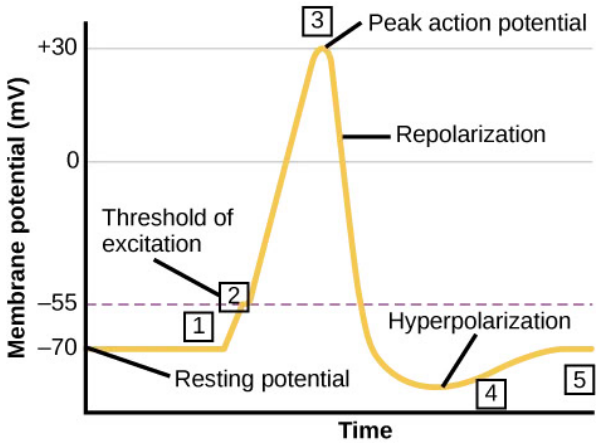



Figure 2.2: Action potential [1]

Two different neurons communicate through specialized contact called synapses and through certain chemical agents known as neurotransmitters. The synaptic cleft is a gap between the axon terminal of the pre-synaptic neuron and the receptors of the post-synaptic one. Neurotransmitters are released as the action potential reaches the axon terminal(bouton). The two types of transmitter, once received by their dedicated receptor, can cause either an increase or a decrease in potential. Helping or hindering depolarization in the post-synaptic neuron is called excitation or inhibition, respectively.

Neurotransmitters are not the only chemicals that affect signal transmission. Neuro-modulators or hormones are other chemicals that may diffuse through the nervous tissue and affect a larger area or remain for longer durations. Despite these considerations, it often holds that a neuron releases the same neurotransmitter from all of its boutons. This strong assumption known as Dale's law[3] allows us to refer to being excitatory or inhibitory as a property of the neuron itself. The ratio between the excitatory and inhibitory neurons in a brain is often close 80% excitatory and 20% inhibitory neurons. The foremost impact of the inhibitory neurons is the maintenance of network stability.

### 2.3.3 Mirror neurons

Mirror neurons are neurons that respond both during execution of an action, and the observation of a similar action performed by others. They were first identified in the macaque ventral premotor cortex and provided evidence that motor systems are involved not only in producing movements but also in representing the observed goal-directed actions. This led to the concept of an action obser-

vation network, the mirror neurons system embedded within the broader Action Observation Network, which links sensory perception to motor representations and enables inference of action goals. Rather than being localized, this system is organized as an integrated sensorimotor circuit that converts visual information about others' actions into structured motor representations.

Within the macaque Action Observation Network(AON), three main areas contribute different parts of the same process. The Anterior Intraparietal Area(AIP) extracts basic object properties (like shape and size) and turns them into information relevant for grasping. F5 uses this information to represent the intended action itself, and it also activates when observing similar actions performed by others. F6 does not specify movement; instead, it regulates whether an action should be carried out or inhibited depending on the context. Together, these areas form a coordinated system that connects perception, action planning, and control of execution[5].

# Chapter 3

# Methodology

## 3.1 Model Details

The architecture used in this project is a leaky RNN adapted from the multitask repository with additional modifications[10]. The original model is designed to perform multiple standard cognitive tasks with a single network and focuses on their representations and how various choices of these tasks can lead to the modular structure of the network[11]. Two levels of biological restrictions are introduced to the model architecture. At the primary level, we restrict the weight signs according to Dale's law. This includes both the input and the recurrent weights. On the second level, the recurrent nodes are separated into three different regions and certain connections are restricted by sign or completely removed.

## 3.2 RNN training with Dale's law

One major objective in this project is to study the effects of Dale's law on RNN training. In our model, there are three sets of weights, and we decided to impose this biological constraint on $W^{in}$ and $W^{rec}$. This is motivated by what happens in the biological nervous system and also our interpretation of these connections. The $W^{out}$ is regarded merely as a readout layer, mapping the signals to the space of the behavioral responses and not as neurons affecting another group.

Applying $W^{out}$ in equation (2.3) from right would mean that a column $j$ of the weight matrix represents the connections from the $j-th$ neuron to the rest of them, so Dale's law translates to preserving the sign for each column. Note that the activation function f should be chosen so that $f(x) = 0$ for $x \leq 0$. This means

that if the sum of the inhibitions received outweighs that of the excitations, that neuron will stay silent. Although other activation functions are not irrelevant, We have chosen ReLU as it is the safest option.

Although not all sensory neurons are excitatory, a vast majority of them are or, in some cases, a sign inversion mechanism makes it so that the input information finds its way in. This is why all input weights are chosen to be positive. To implement this constraint in the training process, we used the following approach[9]:

Consider the diagonal matrix $D$ representing the neuron type with the assigned values of $+1$ and $-1$ accordingly. Having $W_k^{rec}$ at the $k-th$ epoch, the computed gradients will take the weight matrix to $\tilde{W}_k^{rec} = W_k^{rec} + \Delta W_k^{rec}$, which now does not necessarily satisfy Dale's law. Now, let us multiply the updated weight matrix by D from left:

$$W_k^{rec\ +} = \underbrace{\begin{bmatrix} + & + & + & + & - \\ + & + & + & + & - \\ + & + & + & + & - \\ + & + & + & + & - \\ + & + & + & + & - \end{bmatrix}}_{\tilde{W}_k^{rec}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}}_{D} \tag{3.1}$$

At this point, we see that any element not satisfying Dale's law due to the update will be negative. We can either apply a ReLU on the whole matrix to completely prune them or use some less strict activation function like ELU or leaky ReLU to avoid losing the gradient, resulting in excessive dead neurons. Both ReLU and leaky ReLU with $\alpha = 0.001$ were used, where $\alpha$ is the negative slope.

We also set $diag(W^{rec}) = 0$ to ensure that there are no self loops in the network. $W^{in}$ follows a similar procedure using ReLU but much simpler, since we want all of them to be positive.

The exact ratio between inhibitory and excitatory neurons in the nervous system depends on the region and is often difficult to measure. Nevertheless, this ratio is known to be close to 1:4 for interneurons. Although not showing the best performance, we will use 25% inhibitory neurons as the standard value in this study[8].

## 3.3 Tasks

In order to probe and study the neural mechanism of cognition, the choice of tasks is crucial. Neurophysiological tasks are standardized experimental procedures paired with brain-activity measurements. Each task may require one or multiple cognitive processes. We will start by introducing the general structure of a task trial and then proceed to how these tasks are translated into data for RNN training.

### 3.3.1 Structure

A cognitive task consists of a sequence of sensory, cognitive, and motor processes. Sensory processes refer to the acquisition and processing of all input signals relevant to task performance. This may cover any input to the system under study, such as visual or auditory cues signaling when to take action, stimuli possibly in multiple modalities, and rule inputs that represent which task is desired.

Cognitive processes are less directly observable and therefore more difficult to categorize. They correspond to the internal computations required to transform sensory information into appropriate behavioral responses. Examples include proactive response, working memory, comparison, gating, etc. The precise set of cognitive processes involved depends on the task under consideration.

Finally, motor processes correspond to the observable output of the system. Depending on the experimental setting, these outputs may represent behavioral responses such as a set of available actions or some reaction such as the saccadic eye movement, or they may correspond to neural recordings of a downstream population of interest.

### 3.3.2 Task representation

While cognitive tasks are described as processes, RNNs operate on fixed-dimensional numerical vectors. Therefore, each task must be translated into a structured input-output representation that preserves the underlying computational demands while remaining suitable for training. The structure of Yang's task battery is explained in the following.

At each time step, the network receives an input vector and produces an output vector. The input has a fixed dimensionality of 85 and is composed of four

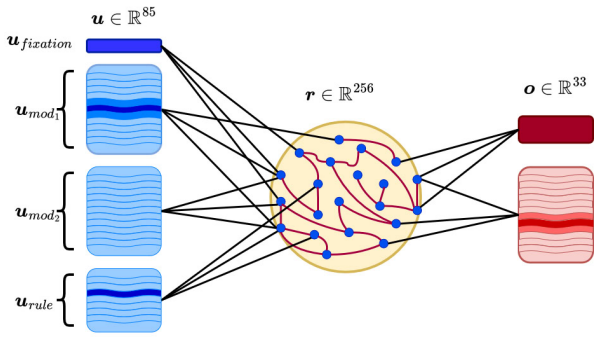



Figure 3.1: The input/output architecture at one time step used in Yang's task battery[11, 4]. In this scheme for example, the second modality is silent.

functional components: a fixation signal, two sensory modalities, and a set of rule inputs specifying the active task context.

The fixation input is a single scalar unit that indicates whether the network is required to maintain baseline activity or if it is time to respond. The sensory inputs or the stimuli are split into two modalities, each represented by 32 units arranged on a circular ring(periodic boundaries). Each direction is represented as a distributed activity pattern across all 32 units, with stronger activation in units whose preferred angles are closer to the stimulus direction. The two modalities correspond to two separate sensory channels that both encode directional information. Depending on the task condition, they can either provide the same directional cue or different ones, requiring the network to integrate or compare them.

In addition to stimuli and fixation inputs, the network receives a 20-dimensional rule input vector. This vector encodes the identity of the task in a one-hot encoded, indicating which cognitive transformation must be applied to the incoming sensory information. This mechanism allows the same network architecture to be used in multiple tasks, while selectively activating task-specific computations.

The output is a 33-dimensional vector consisting of one fixation unit and 32 directional response units. During response intervals, and only then, the network must shift the activity from the fixation to the appropriate direction , which is decoded from the response over the ring.

Task trial data are generated in training or test modes, the training mode having randomness in choice of total time length or stumili onset. These details will be mentioned for each task.

### 3.3.3 ReactGo

ReactGo is a simple sensorimotor mapping task. A single directional stimulus is presented at a specific angular location in one of the sensory modalities. After an initial fixation interval, the network is required to maintain fixation during stimulus presentation and to produce a response only when the response interval begins.

The stimulus directly defines the correct response: the network must select the same angular location as the one indicated by the input. There is no comparison between multiple stimuli required, and no decision between alternatives is involved. The task therefore reduces to learning a direct mapping from sensory input to motor output, conditioned on the temporal structure of the trial.

The fixation input starts at +1 at the beginning of the trial, indicating that the model should also keep the fixation output high(+0.8). The fixation input never goes off, and the network should respond as soon as the stimulus appears. Moving forward in time in the range of $[500, 2500](ms)$, the 32 units corresponding to $\psi_i = \frac{2\pi i}{32}$ for modality 1 or 2 are chosen randomly and start to fire, showing the direction $\psi$ according to this bell curve.

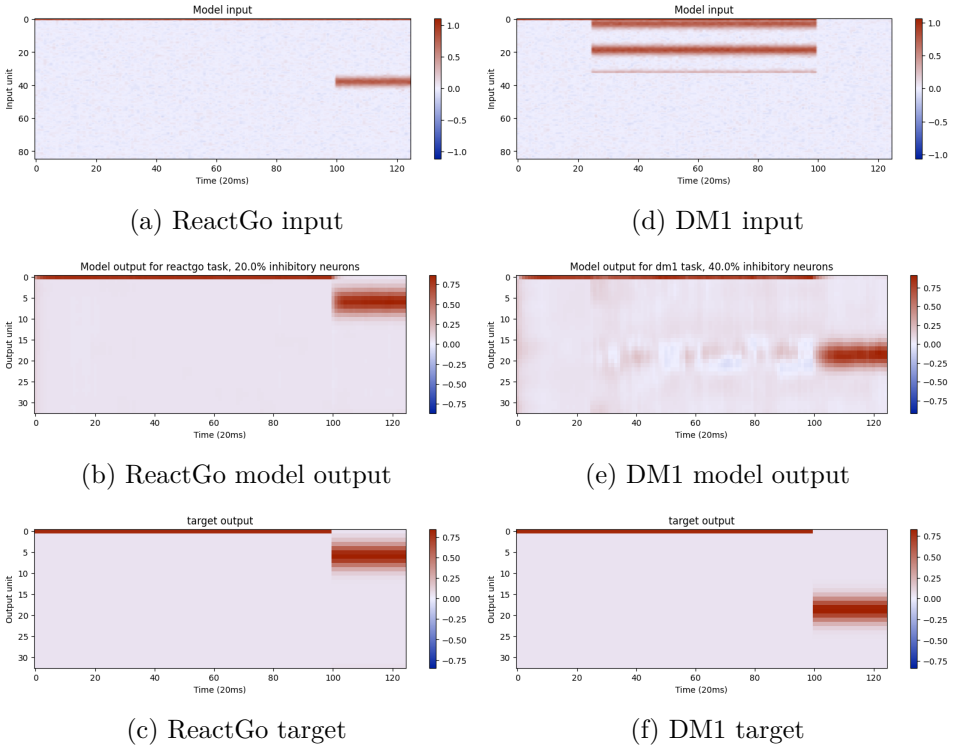
$$\mathbf{u}_i = 0.8.\gamma.\exp\left[-\frac{1}{2}\left(\frac{8|\psi - \psi_i|}{\pi}\right)^2\right] \tag{3.2}$$

### 3.3.4 DM1

DM1 is a perceptual decision-making task that involves two simultaneously presented directional stimuli, each associated with a strength value. The network is required to compare the two stimuli and select the one with the highest strength $\gamma$.

Unlike ReactGo, the correct response is not determined by a single sensory cue. Instead, the network must evaluate competing sensory evidence before selecting an action. This requires an internal comparison process in which relative stimulus strengths are integrated throughout the response interval. As a result, DM1 introduces an additional computational step beyond stimulus-response mapping, making it a more demanding task in terms of decision making and evidence evaluation.

The same encoding as in equation (3.2) holds for this task as well. In each trial, in contrast, two stimuli are shown simultaneously and stay high until the end




(a) ReactGo input (d) DM1 input

(b) ReactGo model output (e) DM1 model output

(c) ReactGo target (f) DM1 target

Figure 3.2: Example showing tests of 125 time steps for each of the tasks. The less complex task ReactGo(left) with simpler input shows a closer output to target with respect to DM1(right). The model for DM1 correctly finds the direction with the higher strength.

of the fixation epoch. Stimulus 1 is drawn randomly between 0 and $2\pi$, while stimulus 2 is drawn uniformly at least $\frac{\pi}{2}$ away from stimulus 1. In DM1, the two stimuli appear only in modality 1. Similarly to ReactGo, the onset of stimulation and the duration of stimulation depend on the trial mode and can be in the range of $[100, 400](ms)$ or $[400, 1600](ms)$, respectively. More details can be found in `_dm` in `task.py`[10].

### 3.3.5 Three object grasp

This is a task based on an experiment in macaques to examine mirror neurons[5]. The experiment includes three objects, each having a specific property and requiring a different grasping mode, such as a large object that needs the whole palm or a smaller one requiring precision grasp or hook grip for a ring. The neural recordings in this experiment are gathered in two modes of execution and observation from a total of 355 units in the previously mentioned F5, F6, and AIP regions. For the execution mode, the macaque primarily hears an auditory cue which can be of high pitch or low pitch, signaling whether to act or fixate,

respectively. The subject is then presented with one of the three objects by light shining on them. The exact same procedure is repeated with the macaque situated at a distance and a 90 degree angle, observing the human agent performing the task.

This task is encoded as follows:

The inputs consist of two units for the high- or low-pitch auditory cue. At most, one of them can be active simultaneously. The cue is active from some onset $[200, 400](ms)$, to offset time $[500, 900](ms)$, chosen randomly. Note that this requires the working memory to store this information. Three units representing each of the three objects. They start as inactive, activate at some random onset time $[1700, 2100](ms)$ and remain active for the remainder of the trial. Having three inputs at this point might seem redundant, but this is implemented in case of integration with the data. Finally, two units encode the experiment mode. Similarly to the cue, At most one of them can be active simultaneously but conversely it is active from the beginning of the trial and remains active. This choice is motivated by assuming that the macaque is always aware of its situation, hence the mode of the experiment.

There are also seven outputs similar to the input. One fixation output that maps the cue. Three preparation outputs plus three action outputs, one for each object. The preparation output of the corresponding object is expected to be high in case of the high pitch cue and low otherwise, while the action output is high provided that the execution mode is taking place. This rule is simply expressing that the macaque prepares for the action even when only observing.

The seven inputs are referred to as cue: $HP/LP$, objects: $O_i$, rule: $EXEC/OBS$. Similarly for the output, we have, fixation, action: $A_i$, preparation: $P_i$.

### 3.3.6 Region separation

The three regions under study[5] cannot receive every input information simultaneously so we restrict them according to their specialization. We assume that the auditory stimuli do not travel to AIP directly and the information carried by the visual stimuli reaches F6 only through its interactions with the other two regions. Figure 3.5 demonstrates this separation in the inputs.

The inhibitory neurons are chosen randomly to have an equal ratio in each region. The size of each region, contrary to the above, is not the same. The selected region sizes are identical to the 355 neurons recorded in the three corti-

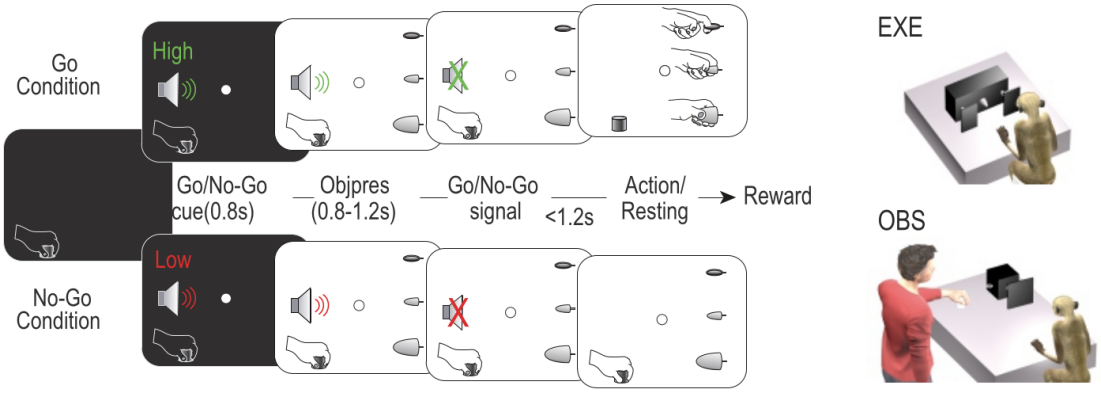



Figure 3.3: Experimental paradigm and task structure (A) Temporal sequence of the experimental task. Following the presentation of a Go or No-Go auditory cue, a light is turned on and one of three possible objects is displayed to the monkey. When the auditory cue ceases, the monkey either grasps the object in response to the Go condition or maintains fixation in the No-Go condition. After correct task completion, a reward is delivered. (B) Schematic representation of the Execution (top) and Observation (bottom) contexts[5]. Onsets and durations for the measurements differ from that of the implementation for the RNN.

cal areas that make up the AON: AIP (86 units), F5 (106), and F6 (163). The practical separation occurs in the connectivity restriction, where the inhibitory connections between two regions. This restriction does not perfectly resemble the connectivity in the AON but is rather an approximation for the short range inhibitory neurons.

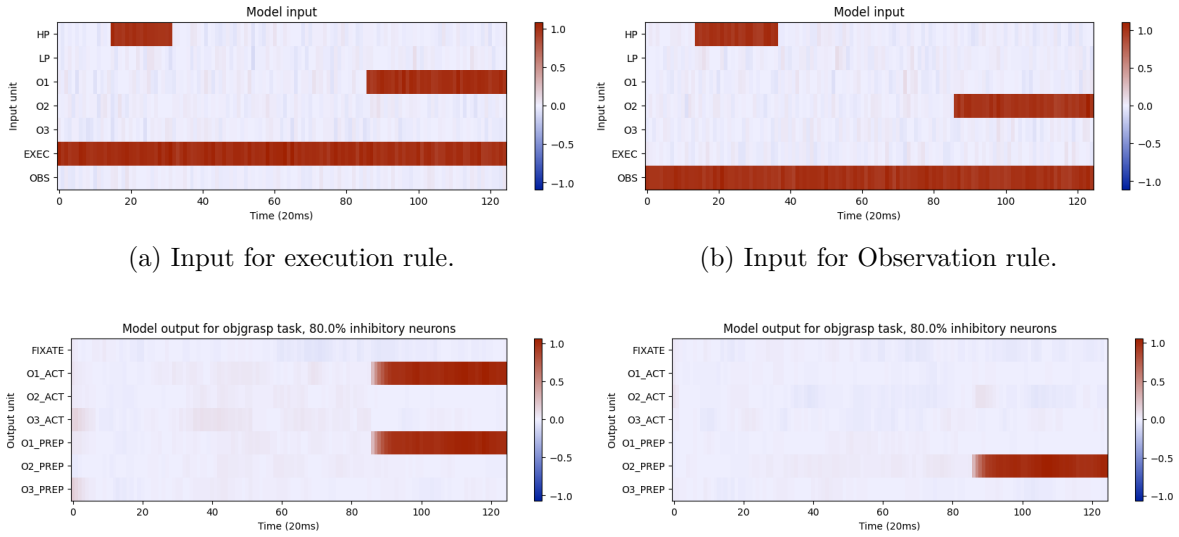
### 3.3.7 Performance measures

Two distinct measures were assumed for the comparison between various models, namely, correlation over tail and total absolute error. The correlation over tail calculates the average for every output unit $\mathbf{o}_{\text{tail}} = \frac{1}{10}\sum_{T-9}^{T}\mathbf{o}^{(t)}$ over the final 10 time steps or 200(ms) in our settings and compares them with the target units using the Pearson correlation coefficient.

$$r_{\text{tail}} = \rho(\bar{\mathbf{o}}_{\text{tail}}, \bar{\mathbf{y}}_{\text{tail}}) \tag{3.3}$$

The total absolute error is exactly as it sounds. We sum up the absolute value of the difference between the output $\mathbf{o}$ and the target $\mathbf{y}$.

$$E = \sum_{k=1}^{d^{out}}\sum_{t=1}^{T}|o_k^{(t)} - y_k^{(t)}| \tag{3.4}$$




(a) Input for execution rule.

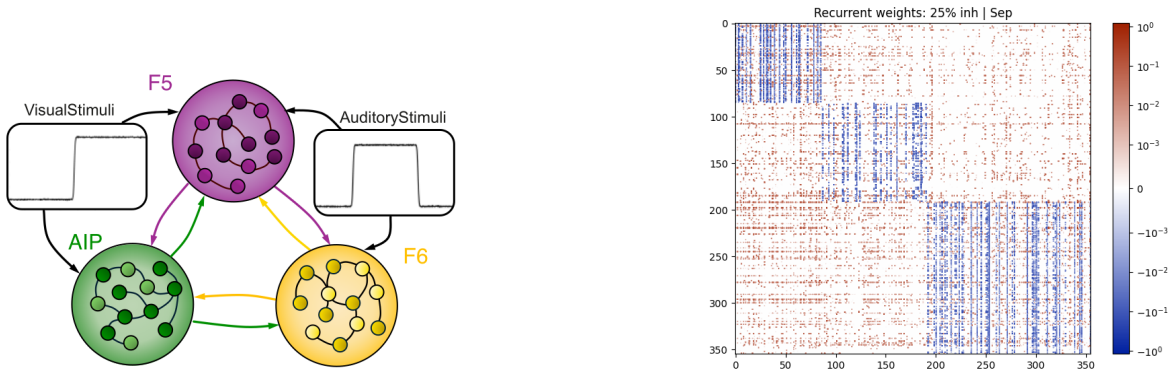
(b) Input for Observation rule.

Figure 3.4: Shows a trial with execution rule on the left. The model successfully prepares and acts on object 1. The trial with observation rule on the right requires preparation but no action taking place.

The main elements contributing to this error are usually the model's reaction time and robustness to noise.

### 3.3.8 Cluster analysis

Certain recurrent neurons form groups to perform various processes, and we want to find groups that are responsible for similar processes. The approach for clustering the nodes is fairly simple via linear tools applied to the RNN hidden states. In each instance of a trial, the neurons have some value $\mathbf{r}^{(t)}$. These values are related to the inputs and outputs, so we can try to approximate the hidden




(a)

(b)

Figure 3.5: Information flow for the dissected network[7].

activity according to them. Let us call the collection of them regressors denoted by $\mathbf{X}$ and the regression coefficients $C$, so that we can try finding $C$ such that:

$$\mathbf{r}^{(t)} \approx C\mathbf{X} \tag{3.5}$$

At this point, each column of the design matrix $C$ is a vector that characterizes a neuron, and this gives us a set of features to compare the neurons. In addition, we added the input and output weights $W^{in}, (W^{out})^{\mathsf{T}}$plus the recurrent biases of the network. Let us join the matrix of coefficients and these weights and call it the feature matrix $S$.

One approach is to feed $S$ to a K-means clustering algorithm and calculate the silhouette score for different number of clusters but since we are not primarily concerned with their euclidean distances, we can process these feature vectors more before clustering them. Take the correlation between each pair of columns of $S$ and form the $d^{rec} \times d^{rec}$ correlation matrix $\rho \in [-1, +1]$. This embedding allows for better emphasis on the behavior on these vectors. We can now form $|\rho|$ or $A = \rho + 1$ as the affinity calculated directly for the K-means algorithm.

# Chapter 4

# Results

The main aim of this chapter is to compare the structure and activity of an RNN trained to perform the ObjectGrasp task based on the input-output mapping of the task, with the corresponding properties of an RNN trained directly on neural recordings [7]. As extensively discussed in [7], this second type of training can be performed by removing the behavioral output from the process and training the recurrent nodes to match the observed neural activity. We shall look for conditions, in terms of increasingly realistic constraints, under which training the network to reproduce behavioral responses may lead the network to mimic the actual properties of a biological neural network, such as connectivity characteristics or even the underlying mechanisms. Among the key constraints we will take into account are i) Dale's law, which is often overlooked in RNN studies, and ii) the fact that in the biological case, processing is distributed among distant brain areas.
This chapter is structured as follows: in section 4.1 we will see the effect of Dale's law on training performance and the properties of the weight matrix. Section 4.2 will begin by discussing the structure of the network in more detail by looking for clusters and continue with increasing the constraints by removing full network connectivity.

## 4.1 Performance with and without Dale's law

Various models were trained using different values on the excitatory-to-inhibitory ratio (in the range $[0.1, 0.9]$) to examine the effect of Dale's law on training performance. Since Dale's law restricts the set of possible solutions for an RNN, we might expect a decrease in training performance. Actually, the restricted models still perform comparably with the case without restrictions. Figure 4.1

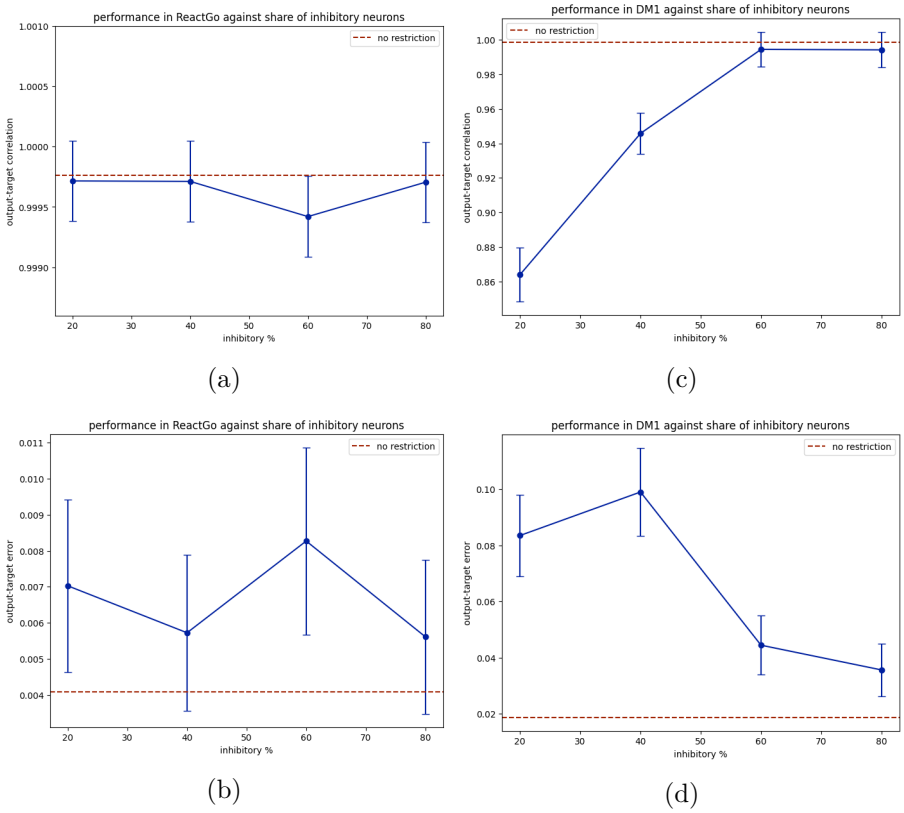



Figure 4.1: ReactGo(left) and DM1(right) models show noise like behavior over varying the share of inhibitory neurrons. DM1 being the harder task hints a trend in favor of higher inhibitory neurons. Every performance is consistently weaker than the case with no restriction at all. Correlation and error averaged over an ensemble of 5 models with 256 recurrent neurons.

shows this for the two tasks (ReactGo and DM1). The performance loss is negligible, especially for the simpler of the two.

Recurrent weights show similar features for different tasks but varying strengths for different values of inhibitory share. As a sanity check of the compliance of the training with Dale's law, Figure 4.2 demonstrates that the weights in each column share the same sign. The activation function *ReLU* naturally promotes sparsity [6]. We can see how restricting the number of excitatory neurons can also affect sparsity in Figure 4.3: indeed, for both ReactGo and DM1, sparsity decreases nearly monotonically, from roughly 0.5 (50% of the links are nonzero) to 0.25 (25% of the links are nonzero) as the fraction of inhibitory neurons goes from 0.1 to 0.9.

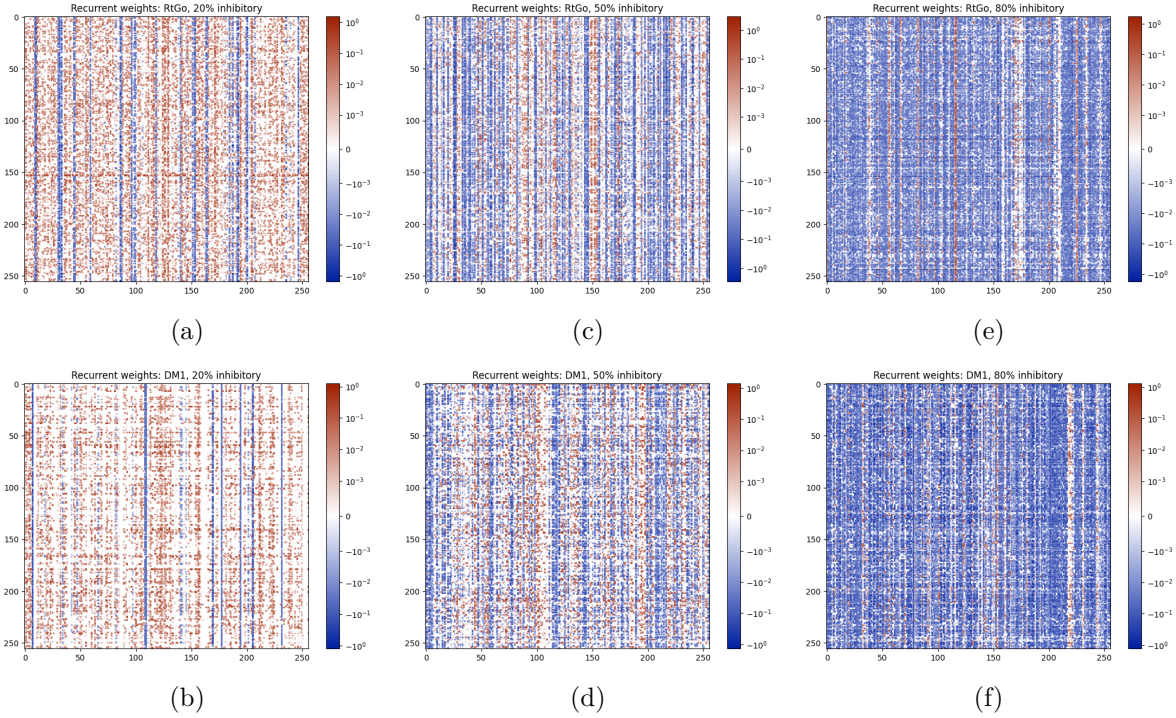



Figure 4.2: Recurrent weights for the ReactGo(top) and DM1(bottom) tasks over 20%(left), 50%(middle), and 80%(right) inhibitory neurons. Fewer inhibitory neurons result in stronger inhibiting weights and vice versa. Note that the color map axis is logarithmic.

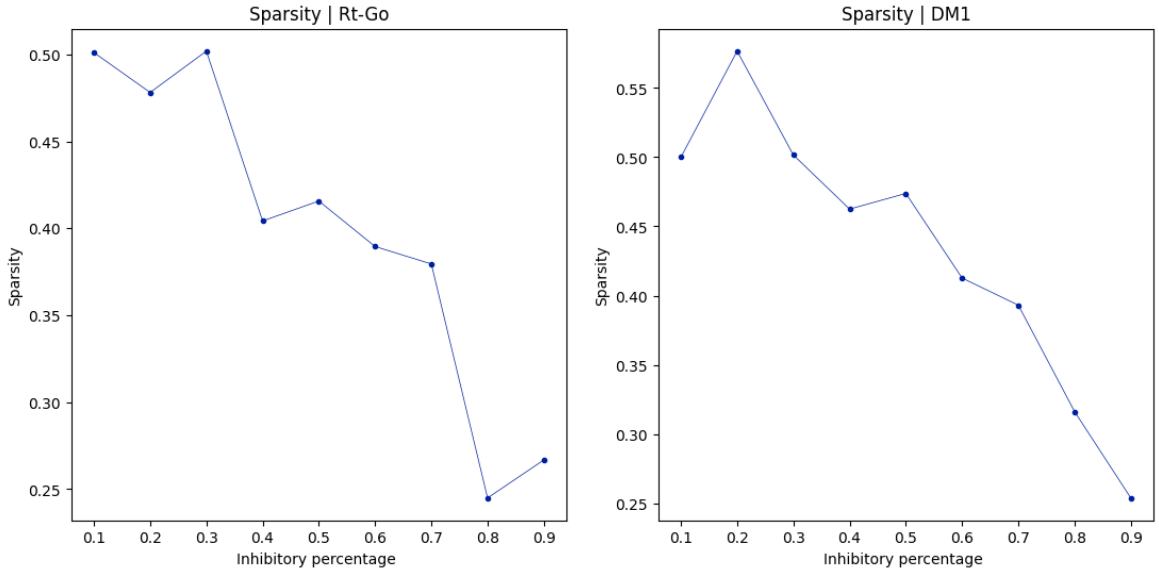
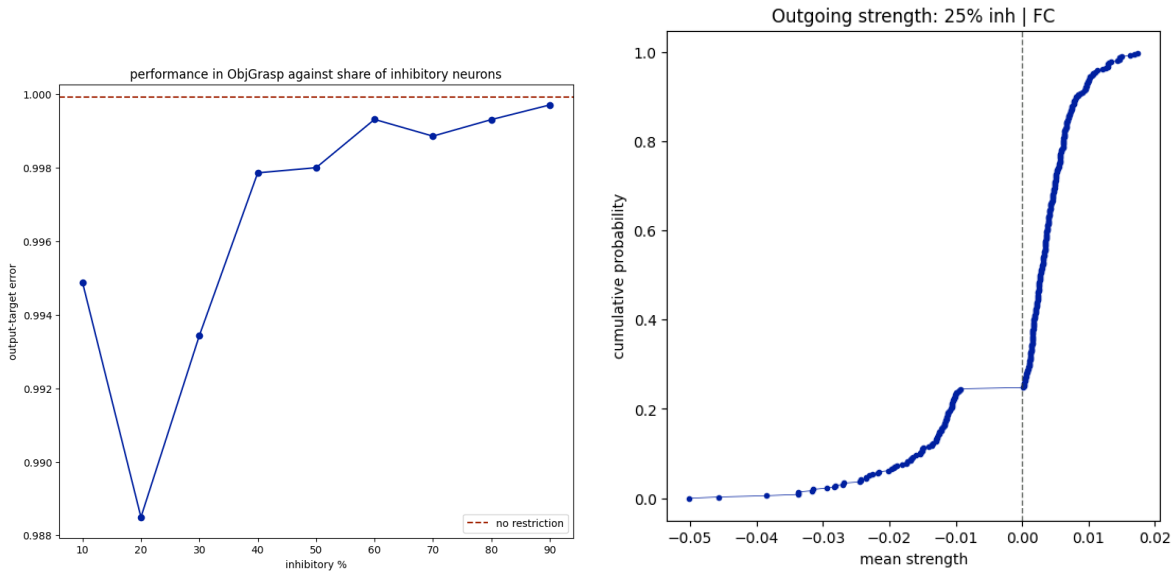



Figure 4.3: Sparsity for DM1 and ReactGo vs share of inhibitory neurons.

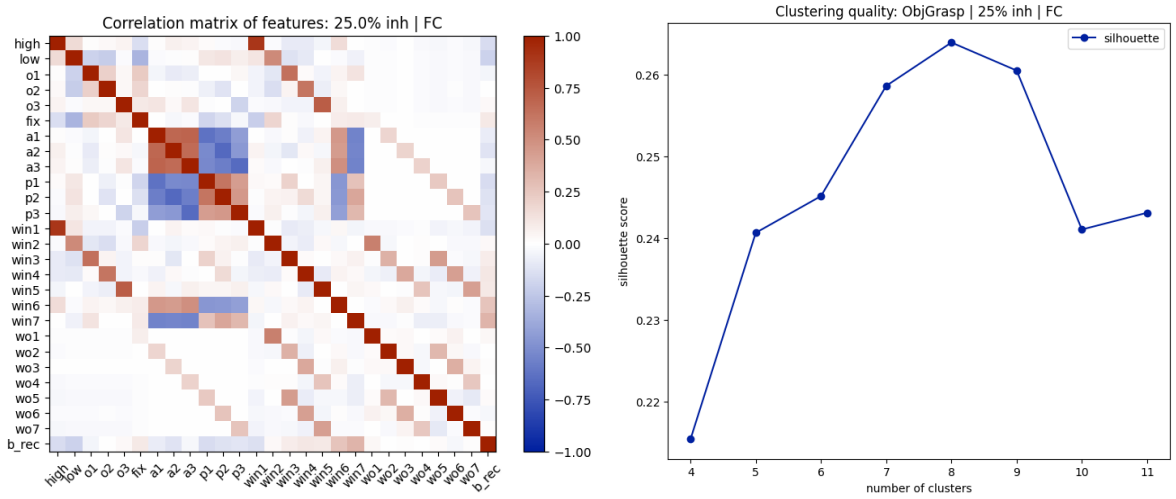



(a) (b)

Figure 4.4: Model performance against share of inhibitory neurons for ObjGrasp (a). CDF for the mean outgoing strength of each neuron. Clear separation between the excitatory and inhibitory neurons and their strengths show an inclination towards the inhibitory neurons (b).

## 4.2 Analysis of the Object Grasp task

The RNNs trained for the Three Object Grasp task have 355 recurrent weights each, mimicking the scenario in Ref. [7]. Unlike in the case of tasks adapted from Yang et al., training is clearly easier with larger fractions of inhibitory neurons (Fig. 4.4a). In our subsequent analysis, we will focus on the network with a fraction 0.25 of inhibitory neurons, mirroring the proportion of inhibitory neurons observed in the real biological network [7]. In Fig. 4.4b, we show the distribution of the strength of the nodes (computed as the sum of all outgoing connections) on all nodes for the case of a fraction 0.25 of inhibitory neurons. This plot shows that the connectivity of the inhibitory and excitatory units is organized differently. Generally, inhibitory units have a larger (absolute) nodal strength than excitatory ones. Furthermore, inhibitory units display a lower bound on absolute nodal strength, and all units have an average strength of at least $-0.01$ in magnitude. Excitatory nodes do not show the same behavior, and have strength that span a continuous interval that is lower bounded by zero, indicating that some units have are nearly disconnected from the rest of the network.




(a) (b)

Figure 4.5: Correlation matrix of the features. Strong blocks between the action and preparation show that there exists a cluster of nodes which is being used by all 3 objects similarly (a). Silhouette score for clusters over a network trained on ObjGrasp with Dale's law (b).

### 4.2.1 Clustering and features

Despite the relatively small number of parameters, simple RNNs are still quite complex to understand at the single-neuron level. In order to further simplify the network, we will perform a clustering analysis, looking for functional groups of neurons with a certain specialization. This simplification may eventually allow us to compare our artificial model with biological models or, more interestingly, artificial networks trained to replicate neural recordings.

To this end, we first used a linear regression model to identify units responding to specific inputs, activated in correspondence of specific behavioral outputs. To this end, we constructed a design matrix with 12 regressors. The first five regressors coincided with input stimuli: high/low pitch, objects o1/o2/o3. Regressors 6-12 coincided with behavioral outputs (three actions and three preparations). We used a total of 10000 trials with 125 time steps each to probe the regressor-neural activity relationship. Possible time lags between stimuli and activity, or activity and behavior, were not taken into account for simplicity and are not expected to be harmful since there are only a handful of sharp changes in the task design (Fig. 3.4).

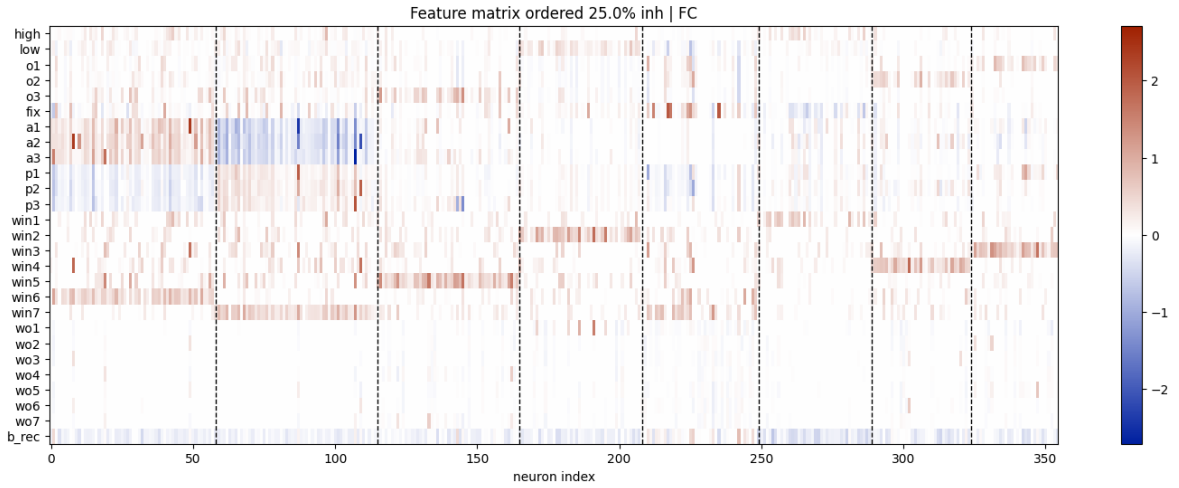



Figure 4.6: Design matrix merged with the network weights to create the feature matrix.

The coefficient matrix obtained from linear regression was then concatenated with the input and output weights $W^{in}$, $(W^{out})^{\mathsf{T}}$ and $\mathbf{b}^{rec}$, which are also expected to contain some information on the role of neurons in the network. This step yielded a $26 \times 355$ feature matrix that we employed for clustering.

Fig. 4.5 shows the correlation between the rows of the feature matrix. The presence of large correlations/anti-correlations between some pairs of features implied that neurons tend to respond with in a similar, or opposite, way to these pairs of features. Of particular prominence, we observe two strongly anti-correlated blocks comprising the entries related to action and preparation. E.g., we observe a high correlation between a1-a3, suggesting a relatively similar response to different actions, and an anti-correlation between a1-a3 and p1-p3, suggesting that neurons tend to respond in the opposite way to action and preparation. In fact, the two blocks exhibit an asymmetry in that the block related to action corresponds to a higher within-block correlation than the preparation block. This asymmetry can be linked to an inherent asymmetry in the representation of the ObjectGrasp task, where it is possible to only prepare (without acting), but there is no case in which the model acts without preparation.

To perform clustering, we instead considered the correlation between the columns of the feature matrix. We then looked for groups of functionally correlated neurons by performing clustering over the rows of this correlation matrix. This corresponds to identifying units with a similar profile in terms of response to input, coding of output, and input/output weights.

As is well known, the number of clusters is a free parameter and its selection is

not trivial. Here, we used the Silhouette score to identify a putatively optimal number of clusters. Over multiple clustering attempts, we obtained that fixing the number of clusters to 8 results in the best cluster separation. Generally (albeit not always with perfect consistency, especially for networks trained with Dale's law), 7 to 10 clusters exhibited a reasonable silhouette score ($\sim$0.4), as shown by the plots in Figure 4.5 (b). This number of clusters could be consistent with a scenario where units functionally cluster into groups that separately process one of the 7 incoming inputs, plus a 'noise' cluster containing units that participate poorly in the overall network's dynamics and that are hardly relevant for task execution. However, it would be unreasonable to expect the units to be fully specialized for a single process, and hence a separation into clusters with sharp boundaries because the processing of different incoming inputs is likely not disjoint and some neurons may respond to several inputs.

We labeled the clusters according to the input feature to which they showed the strongest response. This labeling can be inferred from figure 4.6, where we show the feature matrix, with columns sorted according to the clustering assignment. We stress that neurons in the same cluster have similar functional patterns and clustering is completely unrelated to the spatial arrangement of neurons in Euclidean space.

Three clusters respond most strongly to objects (inputs w3, w4, w5) and are named accordingly. Two clusters respond to auditory cues w1 and w2 and are named accordingly. Two clusters respond to the rule inputs w6, w7. Finally, one cluster responds to the fixation output 'Fix'.

In Fig. 4.7a we show the recurrent layer's weights, where neurons have been sorted according to clustering assignment (so that blocks of the matrix correspond to intra- and inter-cluster connections). Some clusters send few projections to the rest of the network, suggesting that they do contribute significantly to affecting the dynamics of other units (this does not imply that they are redundant, as they may perform relatively segregated processing and hence contribute to the overall network's output).

Figures 4.7 (b) and (c) allow us to have a more detailed look at the behavior of the network. In Fig. 4.7b, we averaged within- and between-cluster connections to obtain a cluster-level connectivity matrix. It should be pointed out that there has not been any attempt to exclude "dead" neurons (i.e., those with no connections with the rest of the network) from the results: therefore, due to the relatively large sparsity (around 60%), the average connection strength between

two clusters can be strongly affected by these neurons. Some clusters, such as Fix, mostly inhibit every other cluster, while clusters like w6 and w7 excite others. There are also clusters such as w3, w4, and w5, related to the objects, which can inhibit or excite parts of other clusters. This suggests the existence of other smaller partitions in each cluster, visible in Figs 4.7 (a) and (b).

Figure 4.7 (c) shows the connections within and between the clusters as a graph with directed edges. This can assist our intuition in understanding the underlying mechanism of the network during ObjGrasp. We could speculate that the inputs and biases keep the network alive and that most gating processes are carried out by the fix cluster. For example, let us consider the case where the pitch is low and the model is expected to fixate. In this case, recurrent nodes that have considerable contribution to other outputs (such as preparation and action) should be 'silenced' by the rest of the network, which implies one of two scenarios: they should either have low activity or cancel each other out. Let us assume the first scenario. In this case, active inhibition would be needed to ensure low activity, since the activation function is ReLU for the recurrent layer. One possible mechanism for this case would be as follows: a cluster like 'w2' (or possibly a sub-cluster within it) could excite 'Fix', which in turn would inhibit downstream units, suppressing their activity.

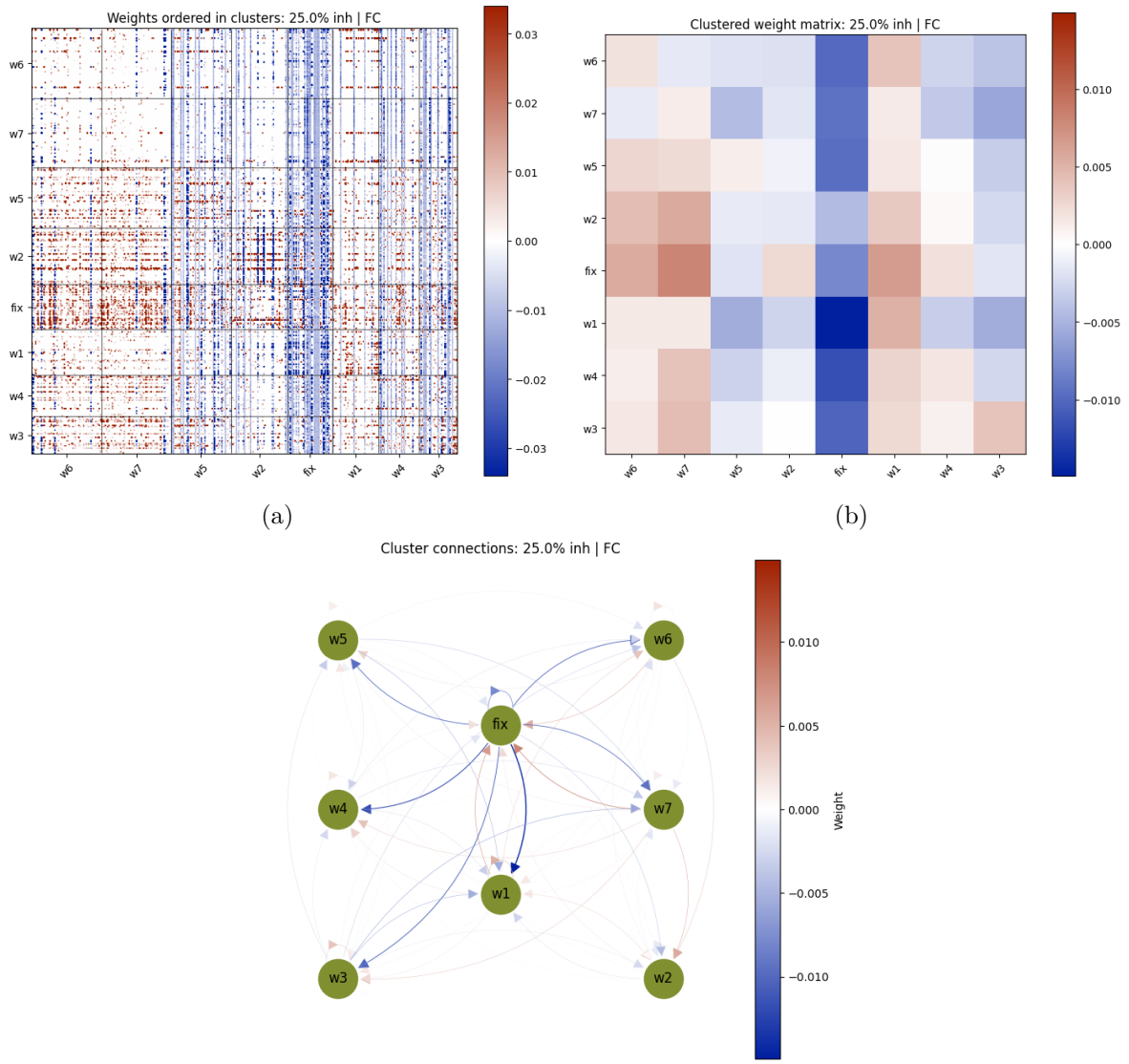
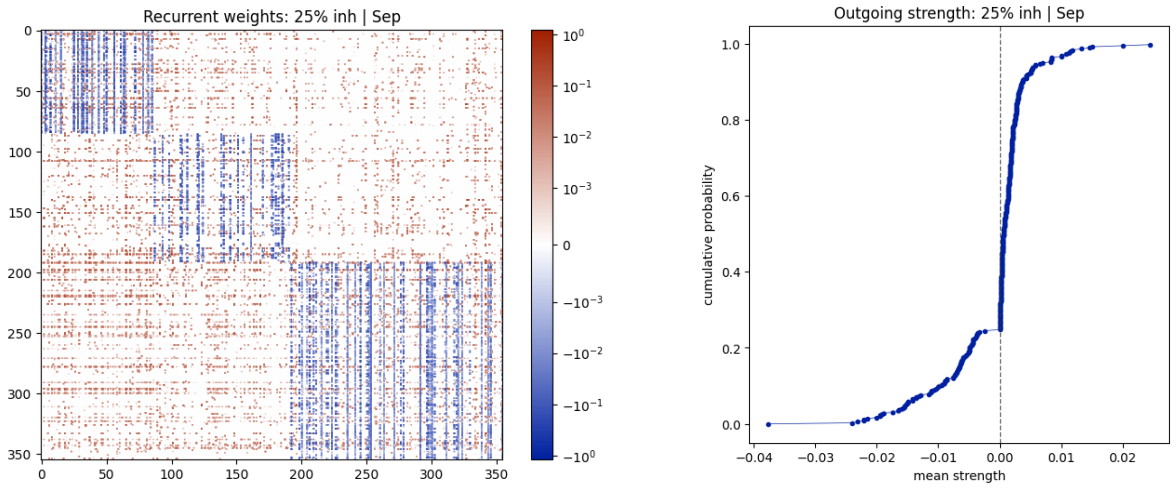



Figure 4.7: All three figures show the same attribute of the same network. In figure (a) and (b) the clusters are ordered by cluster size. For figure (b) and (c) the weight strengths are aggregated as the average of each partition in (a).




(a) (b)

Figure 4.8: Constrained inhibitory neurons (a). Inhibitory connections are more scarce so their weights are less negative. There are many excitatory neurons with outgoing weights averaging to zero which was absent previously in figure 4.5 (b).

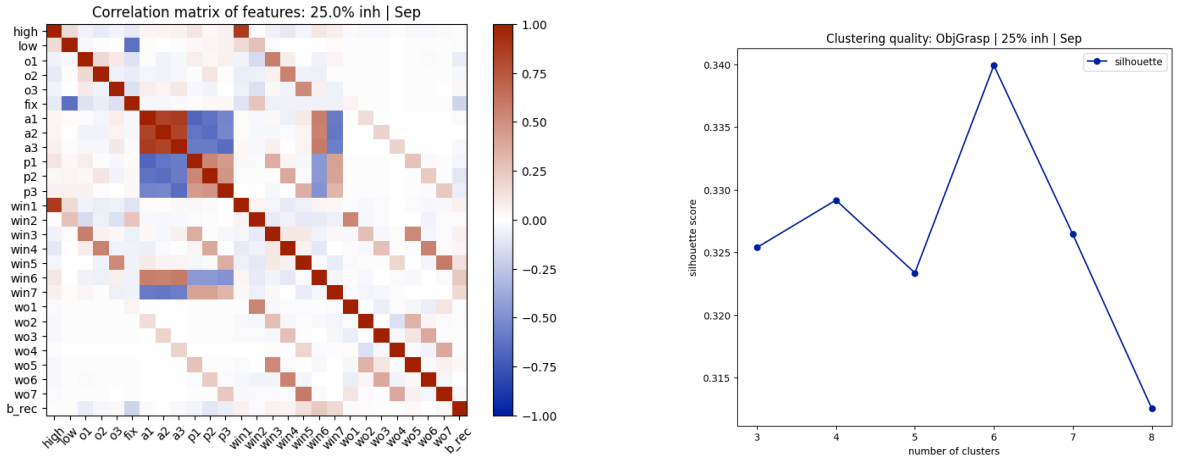
### 4.2.2 Distributed processing in separated regions

In order to build a more realistic and biologically plausible model, the next natural step is to consider the fact that a task is generally carried out by neurons arranged in spatially distant regions in a distributed fashion. Mimicking what was done in previous work [7], we postulate that three main regions are instrumental in determining the behavior of the Monkey: anterior intra-parietal (AIP), a region involved in visuospatial processing, F5, involved in motor preparation, and F6, involved in decision making. We thus labeled neurons as belonging to one of these three areas. To facilitate comparison with previous work [7], we considered 86, 106, and 163 neurons in AIP, F5 and F6. The multi-region scenario calls for two joint alterations of the connectivity structure, concerning input and recurrent connectivity, respectively: I) Information carried by input stimuli does not arrive at every region in the same manner and at the same time. II) Inhibitory neurons are short ranged compared to excitatory neurons.

Figure 4.11

The first point requires us to restrict the input weights so that the input stimuli are connected to their corresponding region according to their type. These constraints were previously discussed and shown in figure 3.5 (a) and we restrict input weights accordingly, as in [7].

The second point demands more caution. In order to impose a distance-dependent




(a) (b)

Figure 4.9: (a) Similar to figure 4.5a with no notable changes. Certain correlations are more pronounced. Noise-like correlations seem weaker in comparison (b) Silhouette score for clusters over a network trained on ObjGrasp with separated regions.

connectivity scalingrealistically, we would need knowledge of the underlying geometry plus a decay rule. Here, we limit ourselves to a more elementary approach and approximate this constraint by allowing inhibitory neurons to project outgoing connection only to neurons within the same region. This can be seen from an example in figure 4.8 (a) where all inhibitory weights across regions are removed. Note that this does not change the count of inhibitory neurons, but only limits their reach.

The results for the distributed model with separated regions are similar in some aspects and different in many others. In Fig. 4.8a, we show the connectivity of the recurrent layer. Consistently with the imposed constraint, no cross-region inhibitory connections exist. The constraint produced a significant change in the structure of the network, significantly enhancing its sparsity. Moreover, as clearly visible from the nodal strengths (Fig. 4.8b), the strengths are weak for most except for a few excitatory units, and the lower bound in absolute strength for inhibitory units nearly vanishes.

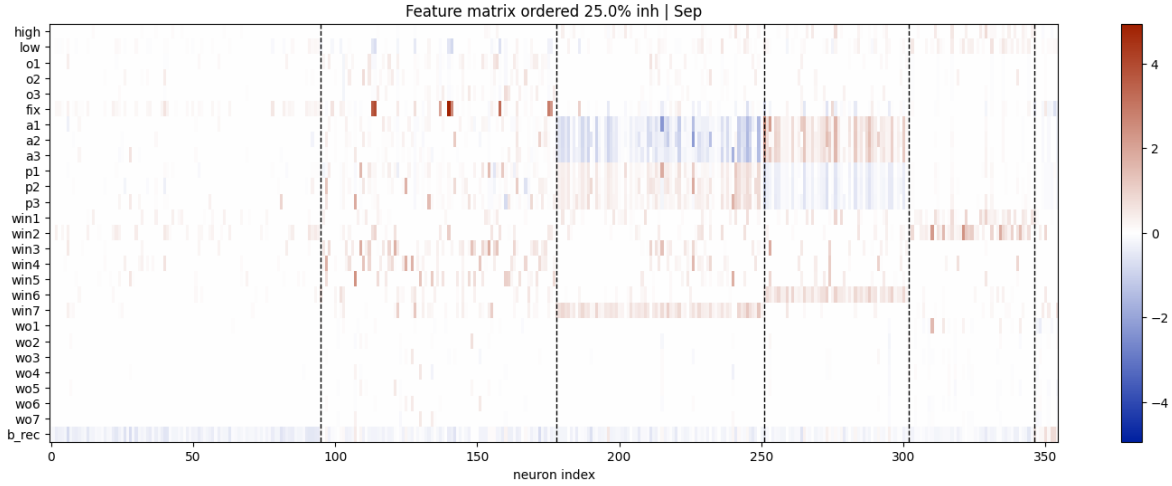
We clustered neurons with the same procedure applied for the non-distributed ObjGrasp training. Unlike the case with no region separation, the optimal clustering score is obtained in 6 clusters instead of 8 (Fig. 4.9b). We can compare the figure 4.10, representing the feature matrix for the separated regions, with its counterpart 4.6 for the fully connected network. We observe an Idle cluster that does not respond visibly to any of the descriptors. This is not entirely surprising,

given that there is a certain number of "dead" neurons (as previously seen from figure 4.9b). The second cluster, 'Fix', is quite different compared to that of the fully connected network, performing both inhibition and excitation 4.11 and responding to the three objects.

The two w6 and w7 clusters are the most conserved characteristic compared to the fully connected RNN. The presence of apparent sub-blocks within cluster like 'Fix' or 'Cue'( Fig. 4.11a) in fact suggests that these clusters could be further subdivided. However, increasing the number of clusters (e.g., by choosing 8 or more centroids for this model) would also lead to subdivide other clusters into smaller groups without revealing anything meaningful. Using a different or enriched feature matrix or resorting to more advanced clustering tools might yield some improvement.

Figures 4.11(a-c), the counterparts of Fig. 4.7, show the connectivity between clusters for a model with separated regions. The most prominent similarity between the graph obtained for separated regions and the previous one is the existence of a cluster with a thorough inhibition capacity, which provides additional supports to our mechanistic conjecture proposed for the fully connected model in the previous section. Taking a closer look at 4.11(a) especially for clusters with functionally mixed clusters like 'Fix' suggests that they may support several sub-processes. For example, processes regarding each of the three objects for the 'Fix' cluster. This makes the graph in 4.11 less straightforward to interpret, nevertheless not contradictory to our previous speculation.

Let us also look at region relevance and primarily the composition of each cluster shown in figure 4.11. The Idle cluster shows a composition very close to the proportions of the three regions, showing that these nodes are almost equally distributed. The Cue cluster, which is more sensitive to auditory stimuli, has no representatives in AIP. AIP is the region that does not receive auditory inputs. Finally, the Inh cluster has only neurons from AIP and mostly F6. This is in accordance with what happens in macaque AON with F6 determining whether an action should take place or should be inhibited.




(a)

Figure 4.10: Design matrix merged with the network weights to create the feature matrix for ObjGrasp with separated regions.

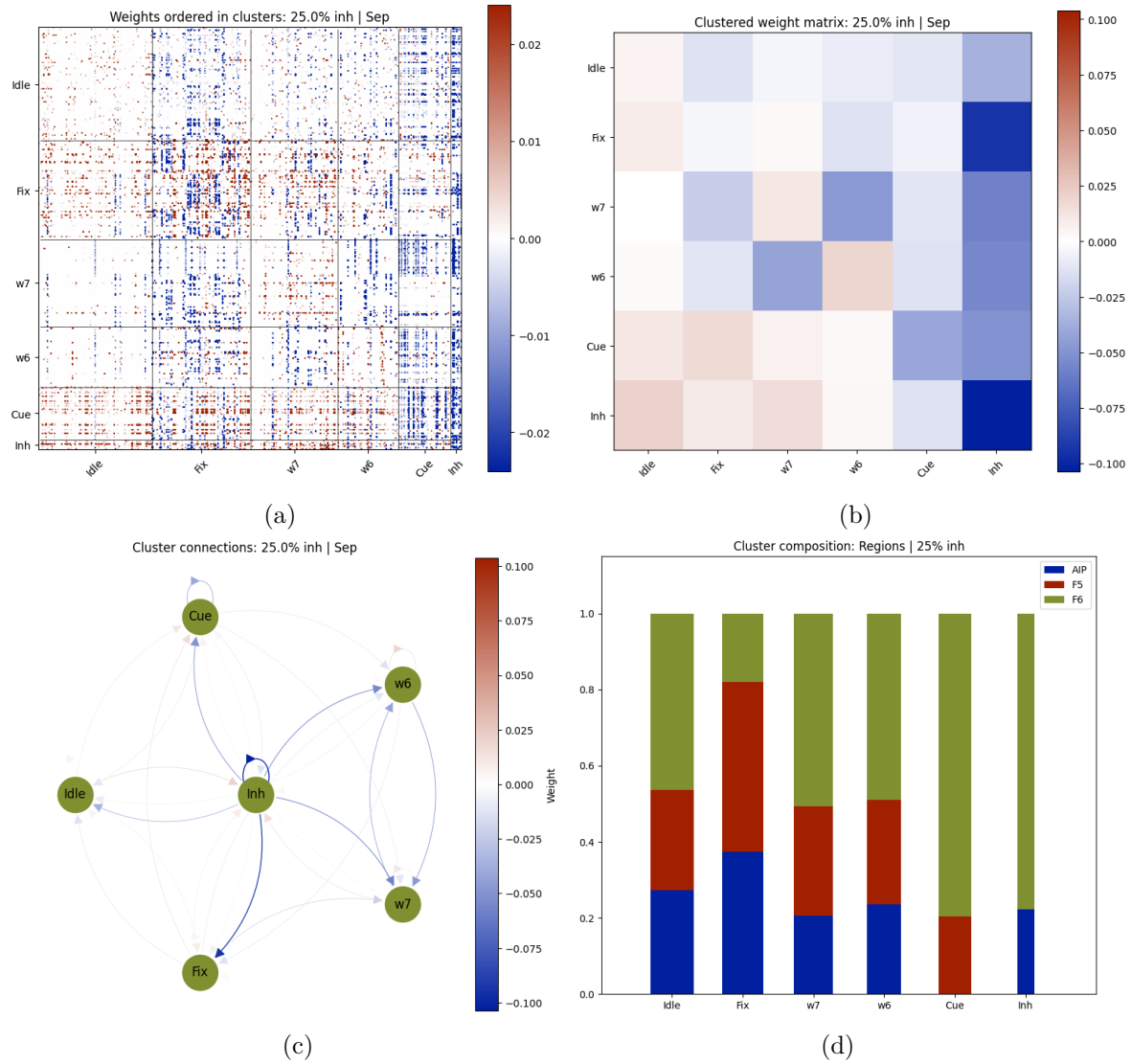



Figure 4.11: Connectivity in model with separated regions trained on ObjGrasp. Inhibition is more highlighted, similar to figure 4.7(a-c). Composition of each cluster from each AON region. Column width is proportional to square root of cluster size for better visualization(d).

# Chapter 5

# Conclusions

In summary, this thesis has demonstrated the feasibility of training Recurrent Neural Networks (RNNs) on stimulus-response associations while adhering to biological constraints, such as Dale's law and distributed regional processing. We focused our analysis on the ObjGrasp task, which was originally designed to investigate the Action-Observation network.

The findings reported in Chapter 4 reveal that imposing biological restrictions in RNN training results in negligible performance loss. For instance, we successfully managed to implement Dale's principle in RNN trained to solve the (now classical) tasks of the 'Cog-Task' battery [11], such as ReactGo and DM1. Furthermore, trained networks developed biologically realistic network features, such as the emergence of dominant inhibitory nodal strengths compared to excitatory units, even if such features were not enforced in training.

Through functional clustering, we identified specialized neuronal groups dedicated to processing specific inputs like object identity and auditory cues in the ObjGrasp task, mirroring the modularity of biological neural circuits. This clustering analysis further allowed us to obtain insight into how the trained network implements the ObjGrasp task.

The introduction of regional separation (AIP, F5 and F6) led the network to develop a distributed architecture consistent with biological observations, such as the exclusion of auditory signals from the AIP region and the concentration of inhibitory clusters within F6 to regulate action. This behavior-driven strategy provides a counterpoint to the previous work by Guglielmi et al. [7], who utilized neural activity reconstruction to show that inhibitory interneurons are central to stability and agent discrimination within the action observation network.

Collectively, our findings suggest that optimizing for behavioral tasks under biologically motivated constraints can guide artificial RNNs towards more biologically plausible structure and function. In future work, our preliminary analysis could be expanded to achieve a better understanding of the key constraints shaping neural network activity during the distributed sensorimotor processing in primates.

# Bibliography


[1] Mary Ann Clark, Matthew Douglas, and Jung Choi. *Biology 2e*. Houston, TX: OpenStax, 2018. URL: https://openstax.org/books/biology-2e/pages/1-introduction.

[2] Soumitra Das and O. Olurotimi. "Noisy recurrent neural networks: the continuous-time case". In: *IEEE transactions on neural networks* 9 5 (1998), pp. 913–36. DOI: 10.1109/72.712164.

[3] J. Eccles, P. Fatt, and K. Koketsu. "Cholinergic and inhibitory synapses in a pathway from motor-axon collaterals to motoneurones". In: *The Journal of Physiology* 126 (1954). DOI: 10.1113/jphysiol.1954.sp005226.

[4] Jacopo Fadanni et al. "Exploring neural manifolds across a wide range of intrinsic dimensions". In: *PLOS Computational Biology* 22.4 (Apr. 2026), pp. 1–28. DOI: 10.1371/journal.pcbi.1014162. URL: https://doi.org/10.1371/journal.pcbi.1014162.

[5] Carolina G. Ferroni et al. "Local and system mechanisms for action execution and observation in parietal and premotor cortices". In: *Current Biology* 31.13 (2021), 2819–2830.e4. ISSN: 0960-9822. DOI: https://doi.org/10.1016/j.cub.2021.04.034. URL: https://www.sciencedirect.com/science/article/pii/S0960982221005479.

[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. PMLR, 2011, pp. 315–323.

[7] Luca Guglielmi. "Reconstruction and causal perturbation of cortical dynamics with recurrent neural networks". PhD dissertation. Italy: University of Parma, Feb. 2026.

[8] Dale Purves et al., eds. *Neuroscience*. Sixth. New York, NY: Oxford University Press, 2018. ISBN: 978-1-60535-380-7.

[9] H. Francis Song et al. "Training Excitatory-Inhibitory Recurrent Neural Networks for Cognitive Tasks: A Simple and Flexible Framework". In: *PLoS Computational Biology* 12 (2016). URL: https://api.semanticscholar.org/CorpusID:17815145.

[10] Guangyu Yang. *multitask: Code for Task representations in neural networks trained to perform many cognitive tasks*. https://github.com/gyyang/multitask. 2016.

[11] Guangyu Robert Yang et al. "Task representations in neural networks trained to perform many cognitive tasks". In: *Nature Neuroscience* 22 (2019), pp. 297–306. URL: https://api.semanticscholar.org/CorpusID:58006968.