# MACHINE LEARNING IN NEUROSCIENCE USING INFORMATION THEORY

**A PROJECT REPORT**

***Submitted by***


**1. Aniruddha Bhattacharjee 22BCE10820**
**2. Devarsh Shah 22BCE11231**
**3. Khushboo Mittal 22BCE11247**
**4. Sivia Anzal 22BCE11350**
**5. Mariya Gracious 22BCE11427**


*in partial fulfillment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**
*in*
**COMPUTER SCIENCE AND ENGINEERING**


**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHRI KALAN, SEHORE**
**MADHYA PRADESH - 466114**


APRIL 2024

# BONAFIDE CERTIFICATE

Certified that this project report titled **"MACHINE LEARNING IN NEUROSCIENCE USING INFORMATION THEORY"** is the bonafide work of "**Aniruddha Bhattacharjee (22BCE10820), Devarsh Shah (22BCE11231), Khushboo Mittal (22BCE11247), Sivia Anzal (22BCE11350), Mariya Gracious (22BCE11427)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project / research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**PROJECT SUPERVISOR**
Dr. Mohammad Sultan Alam, Assistant Professor, Grade 2
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition II Examination is held on _____

# ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr.S. Poonkuntran, Head of the Department, School of Computer Science and Engineering for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide Dr.Mohammad Sultan Alam, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science and Engineering , who extended directly or indirectly all support.

Last, but not the least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

# LIST OF FIGURES

# ABSTRACT

Understanding how neural systems integrate, encode, and compute information is central to understanding brain function. Frequently, data from neuroscience experiments are multivariate, the interactions between the variables are nonlinear, and the landscape of hypothesized or possible interactions between variables is extremely broad. Information theory is well suited to address these types of data, as it possesses multivariate analysis tools, it can be applied to many different types of data, it can capture nonlinear interactions, and it does not require assumptions about the structure of the underlying data (i.e., it is model independent).We analyze models inspired by canonical neuroscience experiments to improve understanding and demonstrate the strengths of information theory analyses.

# TABLE OF CONTENTS

# 1. INTRODUCTION

The brain has numerous levels of interaction ranging from gene networks that control cell function to neural circuits that control behavior. While the study of each of these levels requires highly specialized data acquisition approaches, they are similar in that they all require the assessment of interactions among numerous variables that fluctuate over time. Improved data acquisition and computing technologies have produced more complex and exhaustive insights into neural processing. Data from neuroscience experiments are increasingly multivariate, such as simultaneous recordings of many neurons or voxels. Moreover, experiments that simultaneously acquire data of different types are common.For instance, an awake behaving *in vivo* calcium imaging experiment with a stimulus and a behavior possesses at least three distinct types of data (physiologic, behavioral, and stimulation data).It is often very difficult to develop hypotheses for rules or models that govern the interactions between the numerous variables in the data that can be tested in a clear and straightforward fashion. Information theory represents a valuable tool to address these increasingly common data analysis concerns.

There are several reasons why information theory is a valuable analysis tool in this field.

1. Information theory's model independence allows for the quantification of relationships within data without the need for predefined hypotheses about variable interactions, enabling a broader range of phenomena to be captured.
2. It accommodates various data types, such as action potentials, BOLD signals, or behavioral parameters, making it versatile for analyzing mixed datasets, particularly useful for studying interactions across different brain levels.
3. Information theory can detect both linear and nonlinear interactions, essential given the prevalence of nonlinear phenomena in neuroscience.
4. Its suitability for multivariate analysis allows for the examination of complex neural systems with numerous variables, including those recorded with advanced techniques like multielectrode arrays or imaging.

In neuroscience, information theory is used to analyze neural data, such as spike trains or imaging signals, to understand how information is encoded, processed, and transmitted in the brain. It provides tools to quantify relationships between neural variables, assess coding efficiency, and investigate the complexity of neural systems.

## 1.1 Problem Statement-

This project aims to address the pressing need for rigorous verification of research findings in information theory applied to neuroscience. Specifically, it focuses on systematically altering the parameters utilized in a selected research paper and examining the resultant changes in reported outcomes. By employing a comprehensive approach that encompasses both experimental and computational methodologies, this endeavor seeks to evaluate the robustness and reproducibility of the original study's conclusions.

## 1.2 Motivation for the work-

The motivation for this work is to uphold scientific integrity and advance our understanding of neural information processing by systematically verifying research findings in information theory applied to neuroscience. This effort aims to ensure the reliability and reproducibility of results, with potential implications for clinical applications and broader scientific knowledge. Additionally, promoting transparency and accountability within the scientific community is a key motive.

## 1.3 Objectives Of the Work-

1. **Parameter Variation**: Systematically modify key parameters such as signal-to-noise ratio, temporal resolution, or network connectivity within the computational models or experimental setups described in the target research paper.
2. **Data Acquisition:** Collect empirical data corresponding to the altered parameter settings through neuroimaging techniques, electrophysiological recordings, or simulated neural activity.
3. **Analysis and Comparison**: Quantitatively assess the impact of parameter variations on the observed outcomes by comparing the newly acquired data with the results reported in the original research paper.
4. **Interpretation and Validation:** Analyze the implications of discrepancies between the original findings and the experimentally validated outcomes.

# 2. LITERATURE REVIEW

## 2.1 Introduction

**Information Theory and Entropy: Foundations of Data Communication**

Information theory, established by Claude Shannon in 1948, provides a mathematical framework for quantifying information, particularly in the realm of communication systems. It builds upon probability theory to analyze the fundamental properties of information, such as uncertainty, surprise, and information content. A central concept within information theory is entropy, which measures the average level of uncertainty associated with a random variable or a message source.

**Quantifying Uncertainty: The Role of Probability**

Imagine receiving a message. The informativeness of that message depends on the level of surprise it carries. If the message is highly predictable (e.g., "The sun rose in the east today"), it conveys little new information. Conversely, an unexpected message (e.g., "You've won the lottery!") holds a high information content due to its element of surprise.

Information theory employs probability to quantify this uncertainty. The probability of an event (like receiving a specific message) reflects how likely that event is to occur. Less probable events, when they occur, carry more information because they are more surprising.

**Entropy: A Measure of Uncertainty**

Entropy (H) is a mathematical construct in information theory that captures the average level of uncertainty associated with a random variable. In simpler terms, it represents the expected amount of information contained within a single message from a particular source.

Here is the formula for entropy (H) of a discrete random variable X:

$$H(x) = -\Sigma(\pi * log_2(\pi))$$

where:

- Σ denotes the sum over all possible values (i) of the variable X
- π represents the probability of the i-th value occurring
- $log_2$ is the logarithm with base 2 (units: bits)

The base-2 logarithm is commonly used because it results in units of information being measured in bits. A bit represents the simplest unit of information, signifying the presence or absence of a signal (e.g., 0 or 1).

**Interpreting Entropy Values**

- **High Entropy (High Uncertainty):** A high entropy value indicates a high degree of uncertainty about the message content. This occurs when there are many possible messages with similar probabilities (e.g., predicting the outcome of a coin flip).
- **Low Entropy (Low Uncertainty):** A low entropy value signifies a predictable message source with little uncertainty. This happens when there's a high probability of a specific message occurring (e.g., a pre-programmed weather report).

## 2.2 Demonstration Models:

We have used Izhikevich neuron models for our analyses, which are chosen for their simplicity, computational efficiency, and widespread acceptance in literature (Izhikevich, 2003, 2007). These models are not used to test new scientific theories; rather, they are only used to provide data that resembled actual neuronal activity for demonstrative purposes.

For all models, we attempted to use a realistic number of observations and number of experiments. In all cases, we generated and analyzed 4 models, which is roughly similar to conducting experiments with 4 neural recordings. We used the equal counts method of binning the data to maximize the entropy of underlying variables and when neuron spikes are concerned, we also focused on the spike counts (rate coding) for information analyses.

We have included the necessary software and toolkit in order to generate and analyze the data used in this article.

### 1. Neuro Demo 1 : Simulating and analyzing encoding by a single neuron model

The simulation and analysis of encoding by a single neuron model are pivotal in deciphering the neural mechanisms underlying information processing in the brain. Through computational simulations, we aim to unravel how individual neurons transform sensory inputs into spiking activity, thereby encoding information for further neural processing. This report explores the dynamics of single neurons, the characterization of input stimuli, the analysis of spike trains, and the interpretation of encoding properties, shedding light on fundamental aspects of neural information processing and its implications for cognitive function.

### 2. Neuro Demo 2 : Simulating and analyzing 2 - 3 neuron information transmission figures

The simulation and analysis of information transmission between 2 to 3 neurons represent a critical step in understanding the dynamics of neural circuits and their role in processing and transmitting information within the brain. By employing computational models, researchers aim to elucidate the complex interplay between multiple neurons, investigating how signals propagate and are transformed across synaptic connections. This report delves into the simulation and

analysis of such neuronal circuits, exploring the dynamics of synaptic interactions, the emergence of network-level phenomena, and the implications for information processing and integration. Through these simulations, we seek to uncover the fundamental principles governing communication between neurons, providing valuable insights into the mechanisms underlying cognitive functions and brain disorders.

### 3. Neuro Demo 3 : Simulating and analyzing small network simulations

Simulating and analyzing small network simulations involves computational modeling of interconnected neural networks to study emergent behaviors and dynamics. These simulations play a crucial role in neuroscience research, offering insights into how neural circuits process information and generate complex behaviors. By emulating the interactions between neurons and analyzing the resulting network dynamics, researchers can investigate fundamental principles of brain function and explore the mechanisms underlying cognitive processes.

### 4. Neuro Demo 4 : Simulating and analyzing the canonical neuroscience experiment

Simulating and analyzing the canonical neuroscience experiment involves computational emulation and thorough analysis of experiments that have become foundational in neuroscience research. These experiments typically aim to investigate specific neural phenomena or behavioral responses under controlled conditions, serving as the basis for understanding fundamental principles of brain function. By replicating these experiments computationally and analyzing the resulting data, researchers can gain insights into the underlying mechanisms of neural processes and their contributions to behavior and cognition.

# 3.SYSTEM ANALYSIS

## 3.1 Methodology

**1. Set Parameters:** We manually set the various parameters for the simulations, such as the number of models to run for each subtype, pulse sizes, stimulation formats, delay, bin size, time window, and analysis parameters.

The predefined parameters and their default values are as follows:

**noiseScale = 0.1;**

This is the strength of the membrane potential noise.

The default value is 0.1, which is the ratio of standard deviations of the noise signal.

The noise is modeled as 1/f noise bound below at 10 Hz with constant power noise from **1 Hz to 10 Hz.**

**interstimT = 500;**

This sets the time between stimulation pulses in milliseconds.

The default value is 500 ms.

**intrastimT = 500;**

This sets the duration of the stimulation pulses in milliseconds.

The default value is also 500 ms.

**nStimSets = 50;**

This is the number of stimulation sets to apply. The default value is 50.

**tau = 0.1;**

This is the bin size/ sampling interval in milliseconds.

The default value is 0.1 ms.

**pulseValueSTD = 5;**

It is the standard deviation of the Gaussian white noise added to the input current received by the model neuron in picoamperes (pA).

The default value is 5 pA**.**

**pulseOnsetSTD = 5;**

This is the standard deviation of the Gaussian white noise added to the onset and offset time of the stimuli pulses received by the model neuron in milliseconds.

The default value is also 5 ms.

**stimFormat = 'ONvsOFF';**

This is the type of stimulation to apply to the neuron model.

It can take on different values, each representing a different stimulation protocol.

Possible values:

1. 'ONvsOFF': alternating periods of stimulus on and stimulus off.

2. 'StimType': multiple stimuli with different strengths will be applied to the neuron.

3. 'Delay': stimulation involves a delay before the stimulus reaches the neuron.

4. 'NonLinear': stimulation is passed through a non-linear filter before reaching the neuron.

**pulseSize = 500;**

This is the size or strength of the stimulation pulses in microamperes (uA).

It depends on the stimFormat specified.

If stimFormat is 'ONvsOFF', 'StimType', or 'NonLinear', then pulseSize can be either a single value or an array of values representing

the strength of each stimulus pulse.

If stimFormat is 'Delay', then pulseSize represents the size of the stimulus on pulse in uA.

**delay = NaN;**

It is the delay in milliseconds (ms) before the stimulus reaches the neuron.

**The variations mades to these parameters according to the EEG dataset are :**

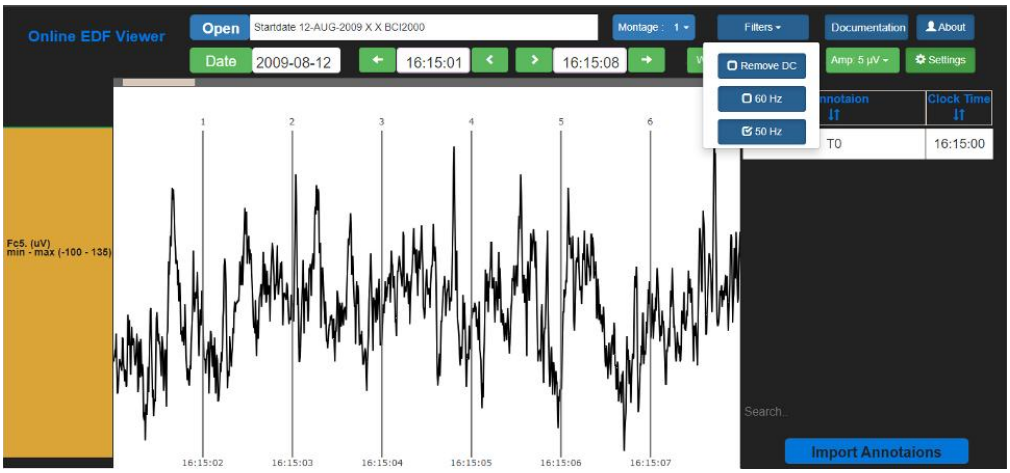
noiseScale is 1/frequency.

These frequency bands from low to high frequencies are respectively called delta (δ) 0.5–4 hz, theta (θ) 4–8 hz, alpha (α) 8–13 hz, beta (β) 13–30 hz, and gamma (γ) >30 Hz.

(citation : *Hassani M, Karami MR. Noise estimation in electroencephalogram signal by using volterra series coefficients. Journal of Medical Signals & Sensors. 2015 Jul 1;5(3):192-200.*)

**Used this motor movement dataset and edf viewer for finding frequency and calculating 1/f [1][2]:**

**We get this graph as output:**

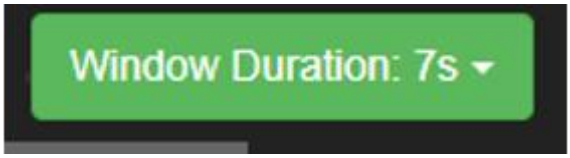



Frequency is selected as 50Hz.

We have noiseScale = 1/f = **0.02.**

**2. interstimT = 500**

In the dataset, the window duration is 7s, i.e **700ms**




**3. intrastimT = 500;**

Similarly for intrastim, we use **700ms**

**4. nStimSets = 50**

The number of stimulation sets to apply depends on the user, the more the stimulation sets the better and accurate result we get. But we have to ensure not to overfit or underfit the data.

So we do as many simulations as the dataset samples, that is 109

**5. tau = 0.1**

We let the sampling interval be 0.1 milliseconds because it is the most accurate.

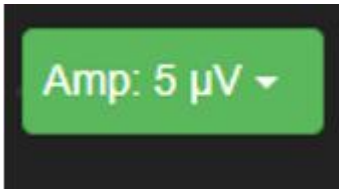
**6. pulseValueSTD = 5**

The standard deviation of the Gaussian white noise added to the onset and offset time of the stimuli pulses received by the model neuron in milliseconds. Calculated with this calculator is 7ms. [3]

**7. pulseOnsetSTD = 5**

Similarly like above, 7ms

**8. pulseSize = 500**




Since it is 5uA , we use 500

**9. stimFormat = 'ON vs OFF'**

Here we can use it for 4 different cases :

'On vs OFF'

**i) delay = NaN**

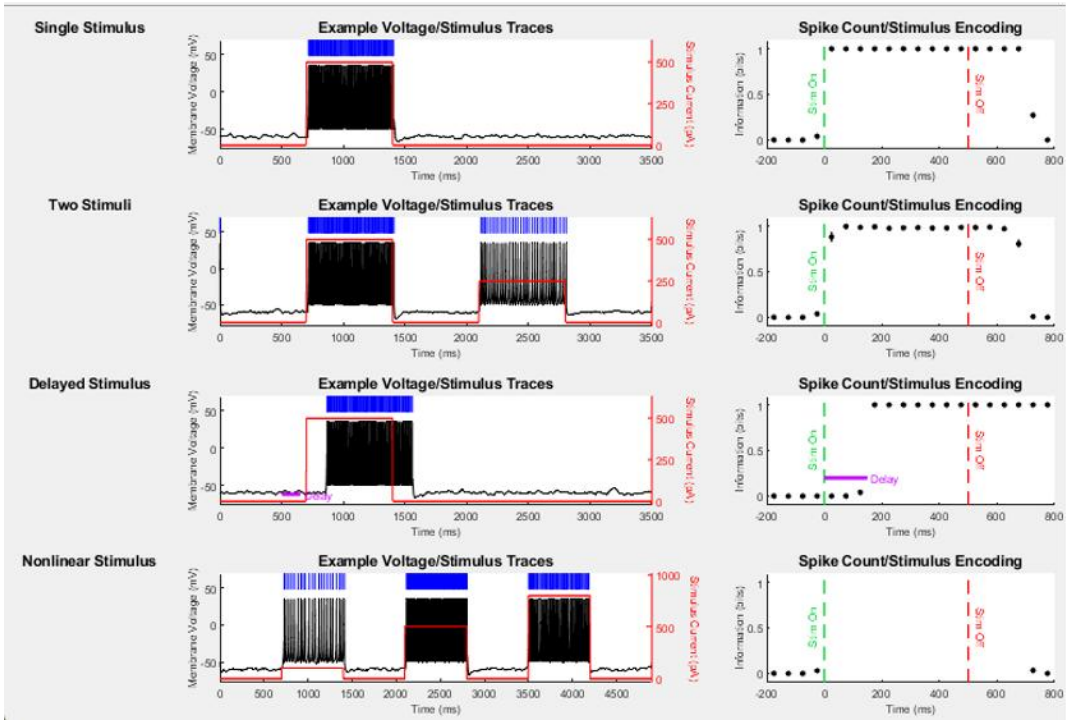



**Fig 3.1**

**ii) delay = NaN**

**StimType'**

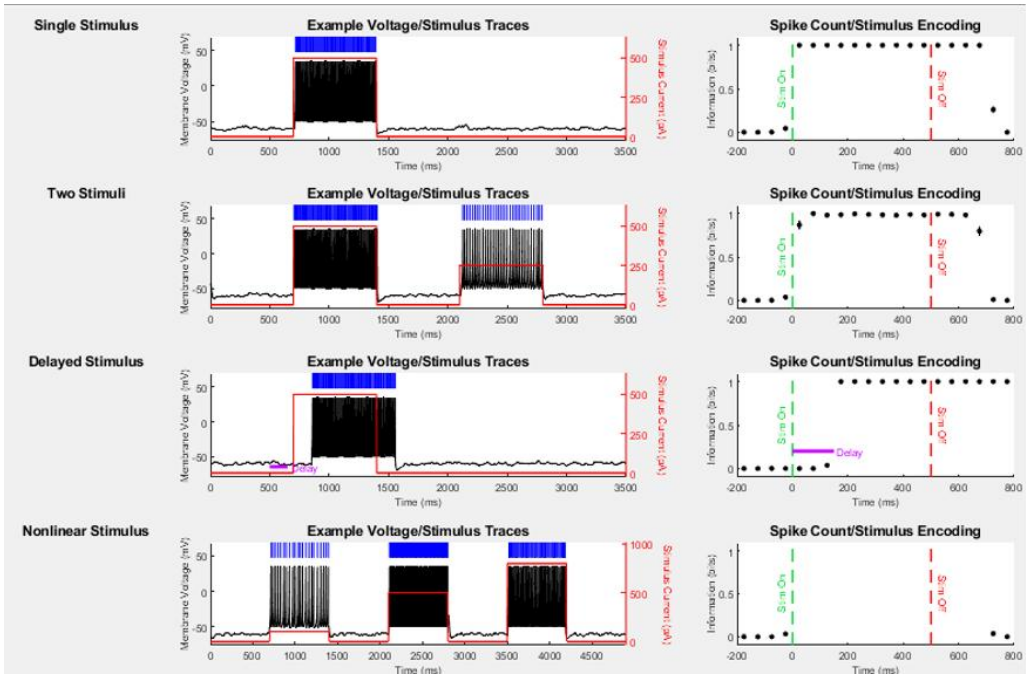



**Fig 3.2**

**iii) 'Delay'**

**delay = 5**

**We will be using 5ms as delay**

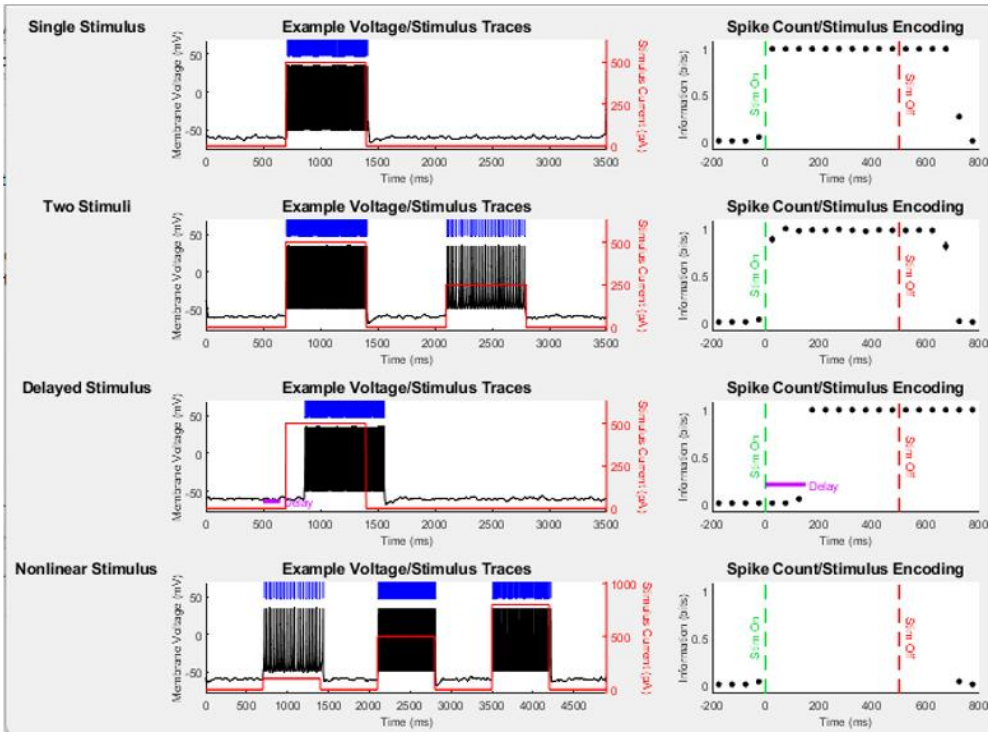



**Fig 3.3**

**iv) 'NonLinear'**

**delay = NaN**

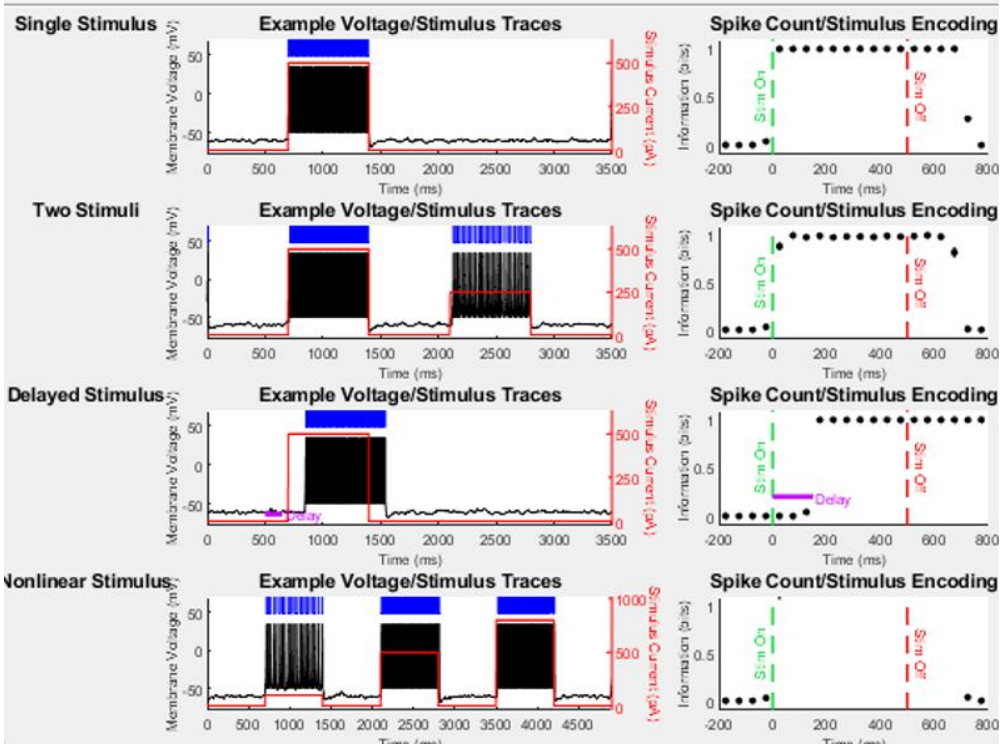



**Fig 3.4**

**2. Make the Model Data:** It generates data for different subtypes of neuron models using the `modeltype1` function. The data includes membrane voltage, spike times, membrane noise, stimulus, and model parameters. The data is saved in MATLAB files.

**3. Convert Raw Data to DataRaster Format:** The raw data is converted to a DataRaster format, which is a specific data format used by the analysis software.

**4. Convert DataRaster to StatesRaster:** The DataRaster format is further converted to a StatesRaster format, which is another data format used by the analysis software.

**5. Perform Information Theory Analysis:** The code performs an information theory analysis on the StatesRaster data to calculate mutual information values between spike counts and stimuli at different time points.

**6. Gather Data for Plotting:** The code extracts and organizes the necessary data for plotting, including example voltage traces, spike times, stimuli, and mutual information results.

**7. Make the Plot:** The code generates a figure with multiple subplots, displaying the simulation results, including example voltage traces, spike times, stimuli, and mutual information box plots for each subtype.

# 4. SYSTEM DESIGN AND IMPLEMENTATION

## 1. Neurodemo1

### Generating the model data

```
%% Make the Model Data

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        if ismember(iSubType,[1,2,4])
            [mV,spkTimes,memNoise,stim,modelParams] = modeltype1_changed_data(stimFormat{iSubType},pulseSize{iSubType});
        else
            [mV,spkTimes,memNoise,stim,modelParams] = modeltype1_changed_data(stimFormat{iSubType},pulseSize{iSubType},delay);
        end

        save([ScratchDir,'ModelType1ST',num2str(iSubType),'Run',num2str(iModel),'.mat'],'mV','spkTimes','memNoise','stim','modelParams')

        disp(['Finished Generating Data for Subtype ',num2str(iSubType),' Run ', num2str(iModel)])

    end
end
```

### Converting the data to dataraster format

```
%% Convert the Raw Data to DataRaster Format

% Go to the directory with the analysis software
cd(AnaDir)

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        % Load the data
        load([ScratchDir,'ModelType1ST',num2str(iSubType),'Run',num2str(iModel),'.mat'])

        % Make the DataRaster
        DataRaster = cell([2,1]);

        % Format the Data
        [DataRaster{1},timeboundaries] = formattool(ones(size(spkTimes)),spkTimes,stim.times,BinSize,MaxLead,MaxLag,'count');
        DataRaster{2} = reshape(stim.types,[1,1,length(stim.types)]);

        % Record the binsize
        modelParams.binsize = BinSize;
        modelParams.maxlead = MaxLead;
        modelParams.maxlag = MaxLag;
        modelParams.Time = timeboundaries;

        % Save the Data
        save([ScratchDir,'ModelType1ST',num2str(iSubType),'Run',num2str(iModel),'DR.mat'],'DataRaster','modelParams')
```

## 2. Neurodemo2

```
%% Make the Model Data

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        [mV,spkTimes,memNoise,current,stim,modelParams] = modeltype2_changed_data(pulseSize, weights(iSubType,:), 'noiseScale', noiseScale(iSubTy

        save([ScratchDir,'ModelType2ST',num2str(iSubType),'Run',num2str(iModel),'.mat'],'mV','spkTimes','memNoise','stim','modelParams')

        disp(['Finished Generating Data for Subtype ',num2str(iSubType),' Run ', num2str(iModel)])

    end
end
```

```
%% Convert the Raw Data to DataRaster Format

% Go to the directory with the analysis software
cd(AnaDir)

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        % Load the data
        load([ScratchDir,'ModelType2ST',num2str(iSubType),'Run',num2str(iModel),'.mat'])

        % Find the number of neurons
        nNeurons = size(spkTimes,1);

        % Figure out which bin size to use
        if iSubType ~= nSTs
            BinSize = BinSizeA;
        else
            BinSize = BinSizeB;
        end

        % Make the Data Raster
        DataRaster = cell([2,1]);

        % Format the Data
        [Temp,timeboundaries] = formattool(ones(size(spkTimes{1})),spkTimes{1},stim.times,BinSize,MaxLead,MaxLag,'count');
        DataRaster{1} = NaN([nNeurons,size(Temp,2),size(Temp,3)]);
        DataRaster{1}(1,:,:) = Temp;
        for i = 2:nNeurons
            DataRaster{1}(i,:,:) = formattool(ones(size(spkTimes{i})),spkTimes{i},stim.times,BinSize,MaxLead,MaxLag,'count');
```

## 3. Neurodemo3

```
%% Make the Model Data

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        if iSubType == 1

            [spkTimes,neuronID,stim,conmat,modelParams,neuronPos] = modeltype3_changed_data('OneStim','conMode','Spatial', 'spatialDist', 'Line'

        elseif iSubType == 2

            [spkTimes,neuronID,stim,conmat,modelParams,neuronPos] = modeltype3_changed_data('TwoStim','conMode','Spatial', 'spatialDist', 'Line'

        end

        save([ScratchDir,'ModelType3ST',num2str(iSubType),'Run',num2str(iModel),'.mat'],'spkTimes','neuronID','stim','conmat','modelParams','neu

        disp(['Finished Generating Data for Subtype ',num2str(iSubType),' Run ', num2str(iModel)])

    end
end
```

```
%% Convert the Raw Data to DataRaster Format

% Go to the directory with the analysis software
cd(AnaDir)

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        % Load the data
        load([ScratchDir,'ModelType3ST',num2str(iSubType),'Run',num2str(iModel),'.mat'])

        % Find the number of neurons
        nNeurons = size(spkTimes,1);

        % Make the Data Raster
        DataRaster = cell([2,1]);

        if iSubType == 1

            % Subtype 1 has only one type of stimulus

            % Format the Data
            [Temp,timeboundaries] = formattool(ones(size(spkTimes{1})),spkTimes{1},stim.times,BinSize,MaxLead,MaxLag,'count');
            DataRaster{1} = NaN([nNeurons,size(Temp,2),size(Temp,3)]);
            DataRaster{1}(1,:,:) = Temp;
            for i = 2:nNeurons
                DataRaster{1}(i,:,:) = formattool(ones(size(spkTimes{i})),spkTimes{i},stim.times,BinSize,MaxLead,MaxLag,'count');
            end
            DataRaster{2} = reshape(stim.types,[1,1,length(stim.types)]);
```

## 4. Neurodemo4

```
%% Make the Model Data

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        [mV,spkTimes,memNoise,current,stim,modelParams] = modeltype4_changed_data(pulseSize, weights(iSubType,:), neuronTypes{iSubType}, backgr
            'varStim', varStim(iSubType,:));

        save([ScratchDir,'ModelType4ST',num2str(iSubType),'Run',num2str(iModel),'.mat'],'mV','spkTimes','memNoise','current','stim','modelParam

        disp(['Finished Generating Data for Subtype ',num2str(iSubType),' Run ', num2str(iModel)])

    end
end
```

```
%% Convert the Raw Data to DataRaster Format

% Go to the directory with the analysis software
cd(AnaDir)

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        % Load the data
        load([ScratchDir,'ModelType4ST',num2str(iSubType),'Run',num2str(iModel),'.mat'])

        % Find the number of neurons
        nNeurons = size(spkTimes,1);

        % Make the Data Raster
        DataRaster = cell([2,1]);

        % Error check that the stimulation times are the same
        if ~isequal(stim.times{1},stim.times{2})
            error('Stimulation times are not identical.')
        end

        % Format the Data
        [Temp,timeboundaries] = formattool(ones(size(spkTimes{1})),spkTimes{1},stim.times{1},BinSize,MaxLead,MaxLag,'count');
        DataRaster{1} = NaN([nNeurons,size(Temp,2),size(Temp,3)]);
        DataRaster{1}(1,:,:) = Temp;
        for i = 2:nNeurons
            DataRaster{1}(i,:,:) = formattool(ones(size(spkTimes{i})),spkTimes{i},stim.times{1},BinSize,MaxLead,MaxLag,'count');
        end
```

## 5. Neurodemo5

```
%% Make the Model Data

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        data = modeltype5_changed_data(expType{iSubType},'parameters', STparams{iSubType});

        save([ScratchDir,'ModelType5ST',num2str(iSubType),'Run',num2str(iModel),'.mat'],'data')

        disp(['Finished Generating Data for Subtype ',num2str(iSubType),' Run ', num2str(iModel)])

    end
end
```

```
% Go to the directory with the analysis software
cd(AnaDir)

for iSubType = 1:nSTs
    for iModel = 1:nModelsPerSubtype

        % Load the data
        load([ScratchDir,'ModelType5ST',num2str(iSubType),'Run',num2str(iModel),'.mat'])

        % Find the number of neurons
        nNeurons = size(data.spkTimes,1);

        % Make the Data Raster
        DataRaster = cell([2,1]);

        if (iSubType == 1) || (iSubType == 2)

            % Center Out Movement Encoding

            % Format the Data
            [Temp,timeboundaries] = formattool(ones(size(data.spkTimes{1})),data.spkTimes{1},data.trialTimes,BinSize(iSubType),MaxLead(iSubType)
            DataRaster{1} = NaN([nNeurons,size(Temp,2),size(Temp,3)]);
            DataRaster{1}(1,:,:) = Temp;
            for i = 2:nNeurons
                DataRaster{1}(i,:,:) = formattool(ones(size(data.spkTimes{i})),data.spkTimes{i},data.trialTimes,BinSize(iSubType),MaxLead(iSubTy
            end
            DataRaster{2} = NaN([1,1,size(Temp,3)]);
            [waste,AngleStates] = histc(data.goalA,data.angles);
            DataRaster{2}(1,1,:) = reshape(AngleStates,[1,1,length(AngleStates)]);
```

# 4. PERFORMANCE ANALYSIS

**Outcome**

We present a general overview demonstrating the application of basic information theory measures to neuroscience data using the Matlab Neuroscience Information Theory Toolbox. Specifically, the analysis examines 13 simulations of neural spiking data to illustrate various aspects of information theory analyses in neuroscience experiments.

Key findings include:

- **Mutual information's utility in quantifying the encoding of stimulus and behavioral information by individual neurons**. It emerges as a cornerstone metric in understanding how neurons encode information from external stimuli and behavioral responses. By quantifying the statistical dependence between neuronal activity and external factors, such as sensory inputs or behavioral outcomes, mutual information provides a robust measure of the information content encoded by individual neurons. This metric offers insights into the neural mechanisms underlying perception, cognition, and decision-making processes, shedding light on how sensory stimuli are transformed into meaningful representations within the brain.

- **The use of transfer entropy and information transmission to assess information flow between neurons.** It serves as indispensable tools for unraveling the intricate dynamics of information flow within neural networks. These metrics offer a quantitative framework for assessing the directed flow of information between neurons, capturing the causal relationships and communication pathways that underlie neural processing. By analyzing the temporal dependencies and directional influences between neuronal signals, transfer entropy and information transmission provide valuable insights into the functional organization and information processing capabilities of neural circuits. From elucidating the mechanisms of sensory integration to uncovering the principles of distributed computation, these metrics offer a window into the complex dynamics of neural communication.

- **The application of partial information decomposition to dissect encoding by multiple variables into redundant, unique, and synergistic components.** It represents a sophisticated analytical approach for dissecting the complex nature of neural encoding by multiple variables. By partitioning the total information content into redundant, unique, and synergistic components, this method unveils the underlying principles governing neural information processing. Redundant information reflects shared encoding between variables, while unique information

highlights distinct contributions from each variable. Synergistic information captures emergent properties that arise from the interaction between variables, revealing non-linear and higher-order encoding mechanisms. Through the application of partial information decomposition, researchers gain a comprehensive understanding of how neural systems integrate and process information from multiple sources, unraveling the intricacies of neural representation and computation.

- **The acknowledgment of the importance of variable assignment (e.g., time delay, bin size) in interpreting information theory analyses accurately.** The accurate interpretation of information theory analyses hinges upon meticulous variable assignment, encompassing factors such as time delay, bin size, and the choice of convergence or divergence schemes. These methodological considerations profoundly impact the outcomes of analyses, influencing the detection of information dependencies and the identification of underlying neural mechanisms. By carefully controlling these variables, researchers can ensure the robustness and reliability of their findings, fostering a deeper understanding of neural information processing. Moreover, thoughtful variable assignment facilitates the comparison and integration of results across studies, advancing the collective knowledge base in the field of computational neuroscience.

## Single neuron stimulus encoding

To demonstrate a possible use for mutual information we will first examine stimulus encoding by an individual neuron. In these examples, we describe various scenarios where information theory can be used to identify neurons that encode a stimulus (or some other variable). Note that similar techniques could be used to identify other signals that encode a stimulus. In the simplest case , a square wave current pulse was applied to the neuron . The spike count during the pulse (e.g., 500–1000 ms,) was compared using mutual information to the spike count during a period with no pulse (e.g.1500–2000 ms,). One variable was the stimulus state (on versus off) and the other variable was the spike count in 50-ms bins, which was then binned into two equal count bins. As expected, very little mutual information was observed before the start of the stimulus (, 0 ms). No information was observed during this time period because there was no difference in firing rate between the stimulus on and off time periods (for instance, compare ~400 ms and ~1400 ms in). Then, during the stimulus, the spike count of the neuron provided a great deal of information about the stimulus state (compare ~600 ms and ~1600 ms in). Finally, when the stimulus ended, the mutual information dropped to near zero (compare ~1100 ms and ~2100 ms in).

A similar pattern was observed when two stimuli were applied to a neuron. In this example, the two stimulus states were strong and weak , but the spike count variable maintained a similar structure. Here, the mutual information increased during the stimulus, but the increase was not as strong as in the previous example because the spike counts of the neuron were not able to differentiate the two stimuli states as accurately.

When a delay in the stimulus was used, patterns of spiking and mutual information that were similar to were observed, except that the spiking and mutual information were delayed. Finally,

even when a nonlinear filter was applied to three stimuli , mutual information was still able to detect encoding of the stimulus by the neuron .

Please note that we intentionally used a strong stimulus in this example (and many subsequent examples) to make the interactions readily apparent. They are so strong in fact that in the examples with two stimuli, it would likely be possible to observe a significant difference in spike rate between the stimuli with a simple *t* test . However, real data are not likely to produce such strong effects, information theory allows for the quantification of the effect sizes, and information theory easily allows for the analysis of cases with more than two stimuli.

**NeuroDemo1:**

MATLAB script for simulating and analyzing the encoding of neural stimuli by single neurons. Code components:

Parameters:

1. nModelsPerSubtype: Number of models to run for each subtype (set to 20).
2. nSTs: Number of model subtypes (set to 4).
3. pulseSize: Defines the pulse sizes for each subtype (e.g., [500] for single pulse, [500, 250] for double pulse).
4. stimFormat: Defines the stimulation format labels (e.g., 'ONvsOFF' for single pulse on/off).
5. delay: Delay in milliseconds for the delay model subtype (used if iSubType is 3).
6. BinSize: Bin size in milliseconds for data analysis (set to 50).
7. MaxLead: Maximum time window before stimulus onset for analysis (set to 200 ms).
8. MaxLag: Maximum time window after stimulus onset for analysis (set to 800 ms).
9. AnalysisParams: Defines parameters for Monte Carlo simulations (e.g., significance level, maximum number of trials).

**Data Generation Loop:**

1. Loops through each subtype (iSubType) and each model run (iModel).

2. Calls modeltype1_changed_data function to generate membrane voltage, spike times, membrane noise, stimulus information, and model parameters based on the subtype and pulse size (or delay for subtype 3).

3. Saves the generated data for each model run in a separate .mat file.

**Data Conversion Loop:**

1. Loops through each subtype (iSubType) and each model run (iModel).

2. Loads the data for the current run.

3. Creates a DataRaster object to format the spike times and stimulus information for analysis.

4. Saves the DataRaster object along with model parameters for the current run.

**State Raster Conversion:**

1. Loops through each subtype (iSubType) and each model run (iModel).

2. Loads the DataRaster object for the current run.

3. Defines the state assignment method based on the number of pulses used in the stimulation.

4. Converts the DataRaster object to a StatesRaster object for state-based information analysis.

5. Saves the StatesRaster object for the current run.

**Information Theory Analysis:**

1. Defines a cell array InfoResults to store information values for each model run.

2. Defines a cell array SubTypeParams to store model parameters for each subtype (assuming all runs use the same parameters).

3. Loops through each subtype (iSubType) and each model run (iModel).

4. Loads the StatesRaster object for the current run.

5. Calculates mutual information between spike counts and stimulus information for each time bin.

6. Stores the calculated information values for the current run in InfoResults.

7. Saves InfoResults and SubTypeParams after processing all runs.

**Data Gathering for Plotting:**

1. Defines a structure DP1 to store example voltage traces for each subtype.

2. Loops through each subtype (iST).

3. Loads the data for the first model run of the current subtype.

4. Calculates the time window based on the stimulation parameters.

5. Extracts example voltage trace, spike times, stimulus current, and delay information for plotting.

6. Saves the extracted data for the current subtype in DP1.

**Information Results for Plotting:**

1. Defines a structure DP2 to store information for boxplots.

2. Loops through each subtype (iSubType).

3. Extracts delay information from model parameters.

4. Gets the time bins for information values.

5. Initializes arrays to store information values for all model runs of the current subtype.

6. Loops through each model run (iModel).

7. Loads information values for the current run.

8. Calculates boxplot statistics (quartiles, mean, and error bars) for each time bin based on information values from all models.

9. Saves boxplot statistics and mean error values for the current subtype in DP2.

**Saving Plotting Data:**

1. Saves the structures DP1 and DP2 containing data for plotting.

Generating Plots:

The provided code snippet describes the parameters for generating plots but doesn't include the actual plotting functions. It defines elements like figure size, margins, colors, line styles, font sizes, etc., for creating several subplots.

In our experiment we used similar factors that were in line with the dataset that we have used:

**noiseScale = 0.02;**
**interstimT = 700;**
**intrastimT = 700;**
**nStimSets = 109;**
**tau = 0.1;**
**pulseValueSTD = 7;**
**pulseOnsetSTD = 7;**
**stimFormat = 'NonLinear';**
**%'ONvsOFF'**
**%'StimType'**
**%'Delay'**
**%'NonLinear'**
**pulseSize = 500;**
**delay = 5;**

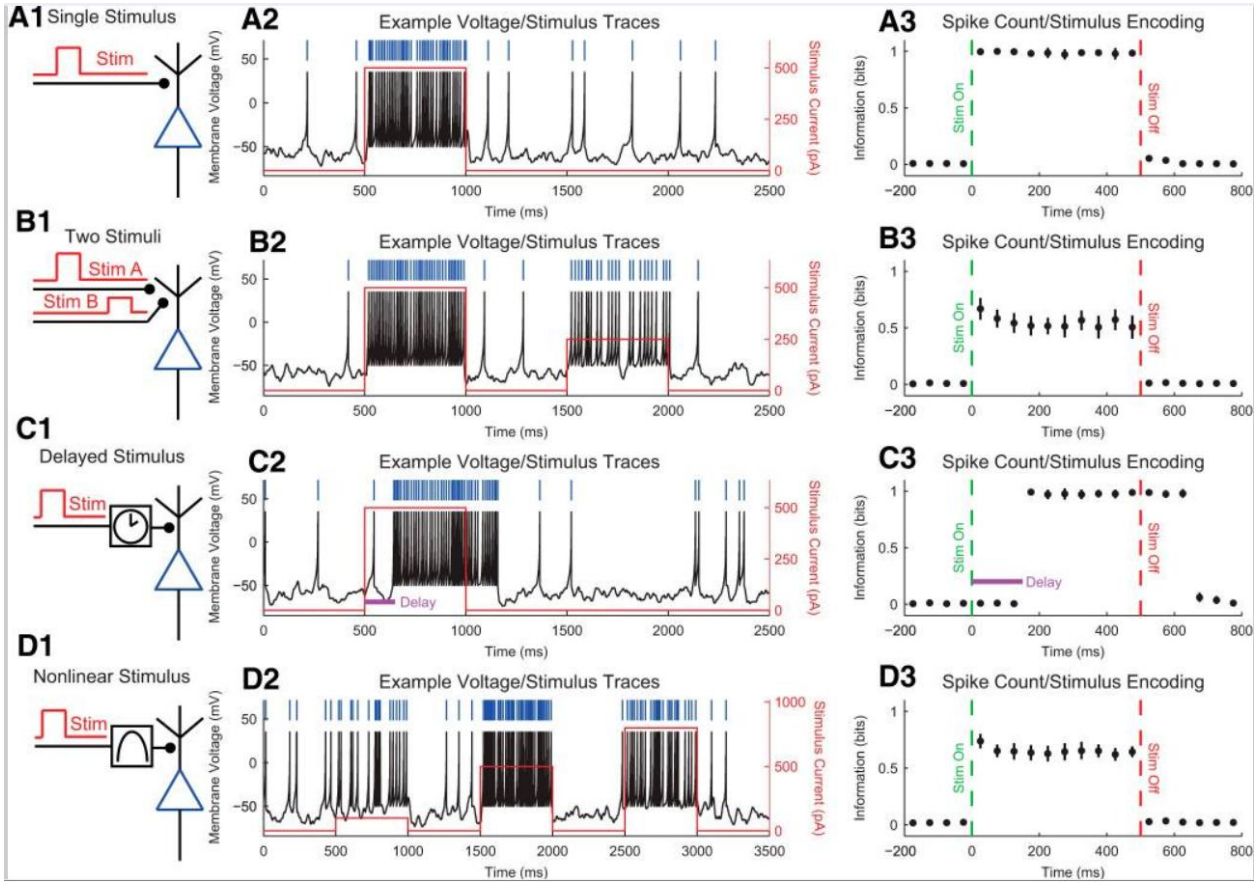



Fig 4.1

## Two-neuron information transmission

**NeuroDemo2:**

The code simulates a model of information processing in neurons. Code components:

Parameters:

1. nModelsPerSubtype: Number of models to run for each subtype.
2. nScan: Number of weights to scan for the changing encoding.
3. weights: Weights for the model subtypes.
4. noiseScale: Noise scale for the model subtypes.
5. BinSizeA, BinSizeB: Bin sizes for different subtypes in analyzing data.
6. MaxLead, MaxLag: Time window around the start of the stimuli to analyze.
7. ScratchDir: Directory to store data.

Loops:

1. The code iterates through different model subtypes and runs multiple models for each subtype.
2. For each model run, it generates data including membrane voltage, spike times, and stimulus information.
3. The data is then converted to a format suitable for information theory analysis.

Analysis:

1. The code performs information theory analysis to quantify how well the neurons encode information from the stimuli.
2. Different analyses are used depending on the subtype (e.g., stimulus encoding by a neuron, information transfer between neurons).

Data Organization:

1. The results of the simulations and analyses are saved in separate files for each model run.
2. Finally, the code gathers data for plotting, including example voltage traces and information theory results.

**noiseScale = [0.2,0.2,0.2];**
**interstimT = 700;**
**intrastimT = 700;**
**nStimSets = 60;**
**tau = 0.1;**
**pulseValueSTD = 7;**
**pulseOnsetSTD = 7;**
**gammaMean = [30,30,30];**
**gammaSTD = [20,20,20];**
**stimType = 'UniSqPulse';**
**spkCurrent = 600;**
**spkDur = 5;**

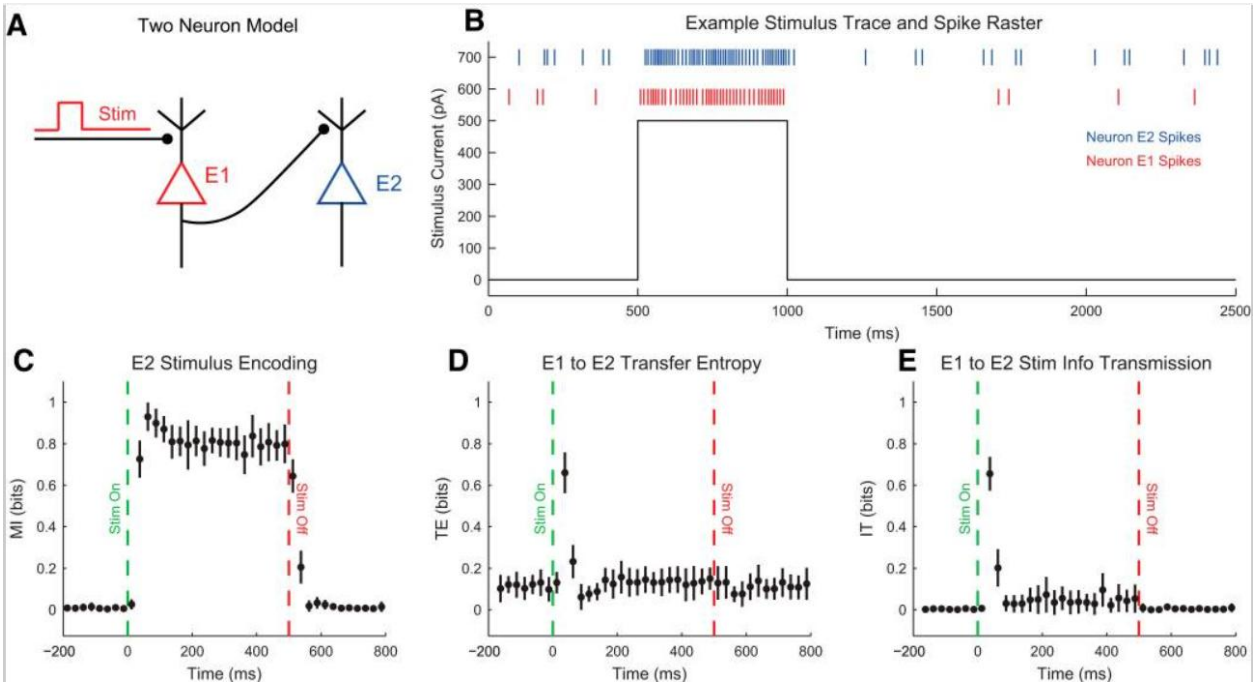



fig 4.2

## Inhibition modulated encoding

**nStimRec = 400;**
**pulseSize = 50;**
**interstimT = 2000;**
**intrastimT = 20;**
**nStimSets = 109;**
**pulseValueSTD = 0.5;**
**pulseOnsetSTD = 1;**
**stabilizeTime = 3000;**
**neuronPos = [];**
**conMode = 'AllToAll';**
**spatialDecay = 0.1;**
**weightScaling = 1;**
**spatialDist = 'Uniform';**

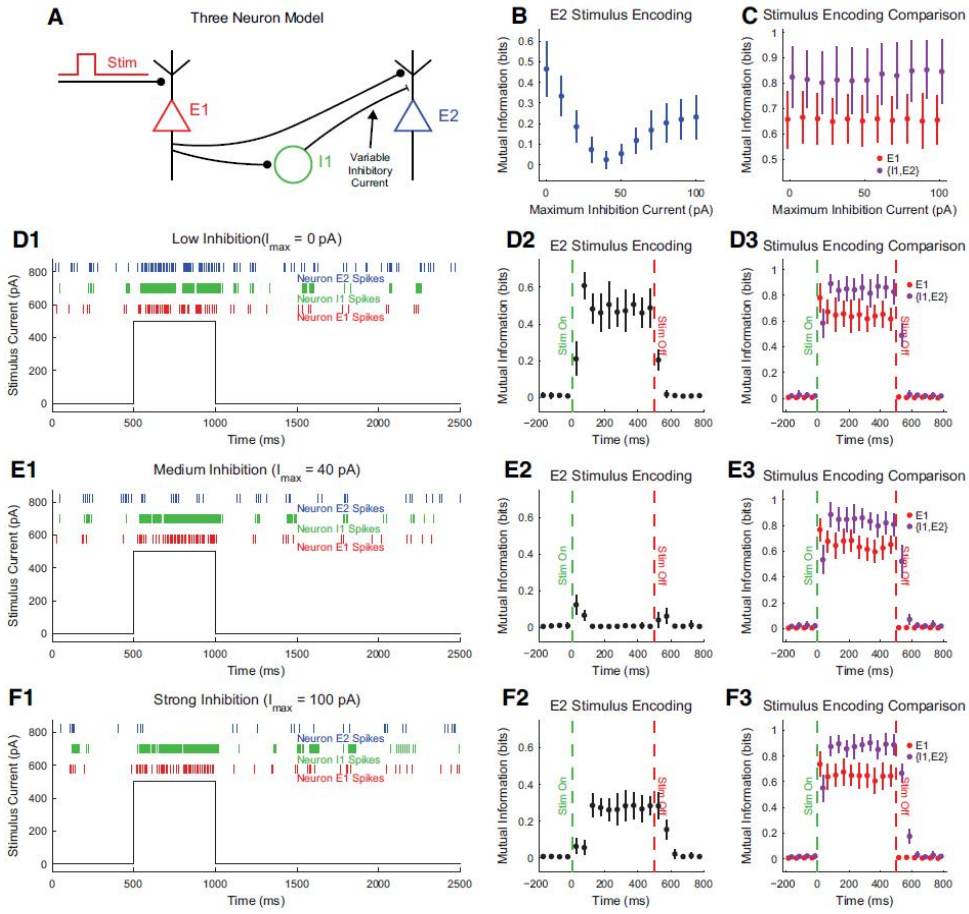



fig 4.3

## Information transmission and encoding in a large network

**NeuroDemo3:**

MATLAB Code for Neural Network Visualization

This code visualizes the results of a simulated neural network, focusing on connectivity, neuron activity, and information flow.

1. Figure Setup:
1. Defines figure dimensions (width, height)
2. Sets margins for labels and axes
3. Chooses font sizes for titles, labels, and text
4. Specifies line widths for plots
5. (Optional) Sets background color or transparency

2. Connectivity Diagram:
1. Reads a connectivity matrix representing connections between neurons
2. Uses different colors to represent connections from different types of neurons:
3. Stim A (stimulatory input A)
4. Stim B (stimulatory input B)
5. Excitatory neurons
6. Inhibitory neurons
7. Plots lines or shapes (circles, squares) to represent connections
8. (Optional) Labels different neuron types with text or legend

3. Example Spike Rasters:
1. Defines time window to display neuron activity
2. Reads data on neuron firing times for different stimuli:
3. Stim A only
4. Stim B only
5. Both Stim A and B
6. Uses different colors to represent different neuron types (same as connectivity diagram)
7. Plots horizontal lines at firing times for each neuron on separate subplots
8. Labels subplots with stimulus type
9. (Optional) Calculates and displays firing rates for each neuron and stimulus

4. Pairwise Information Decomposition (PID) Heatmaps:
1. Calculates PID, a measure of information transfer between pairs of neurons
2. Defines time bins to analyze information flow over time
3. Creates a heatmap for each time bin:
4. X-axis represents sending neuron positions
5. Y-axis represents receiving neuron positions
6. Color intensity represents the PID value, indicating information transfer strength
7. Adds a colorbar to the figure with a legend explaining PID value ranges
8. (Optional) Calculates and displays average PID for each neuron pair

Overall, this code provides a comprehensive visualization of the simulated neural network, allowing researchers to analyze:

**Network Structure:** Visualize connections between different types of neurons.
**Neuron Activity:** Observe firing patterns of neurons in response to various stimuli.
**Information Flow**: Understand how information propagates between neurons using PID heatmaps.

**noiseScale = 0.1*ones([1,4]);**
**interstimT = 5000;**
**intrastimT = 5000;**
**nStimSets = 109;**
**tau = 0.1;**
**pulseValueSTD = 10;**
**pulseOnsetSTD = 10;**
**gammaMean = 30*ones([1,4]);**
**gammaSTD = 20*ones([1,4]);**
**varStim = 0.25*ones([1,4]);**

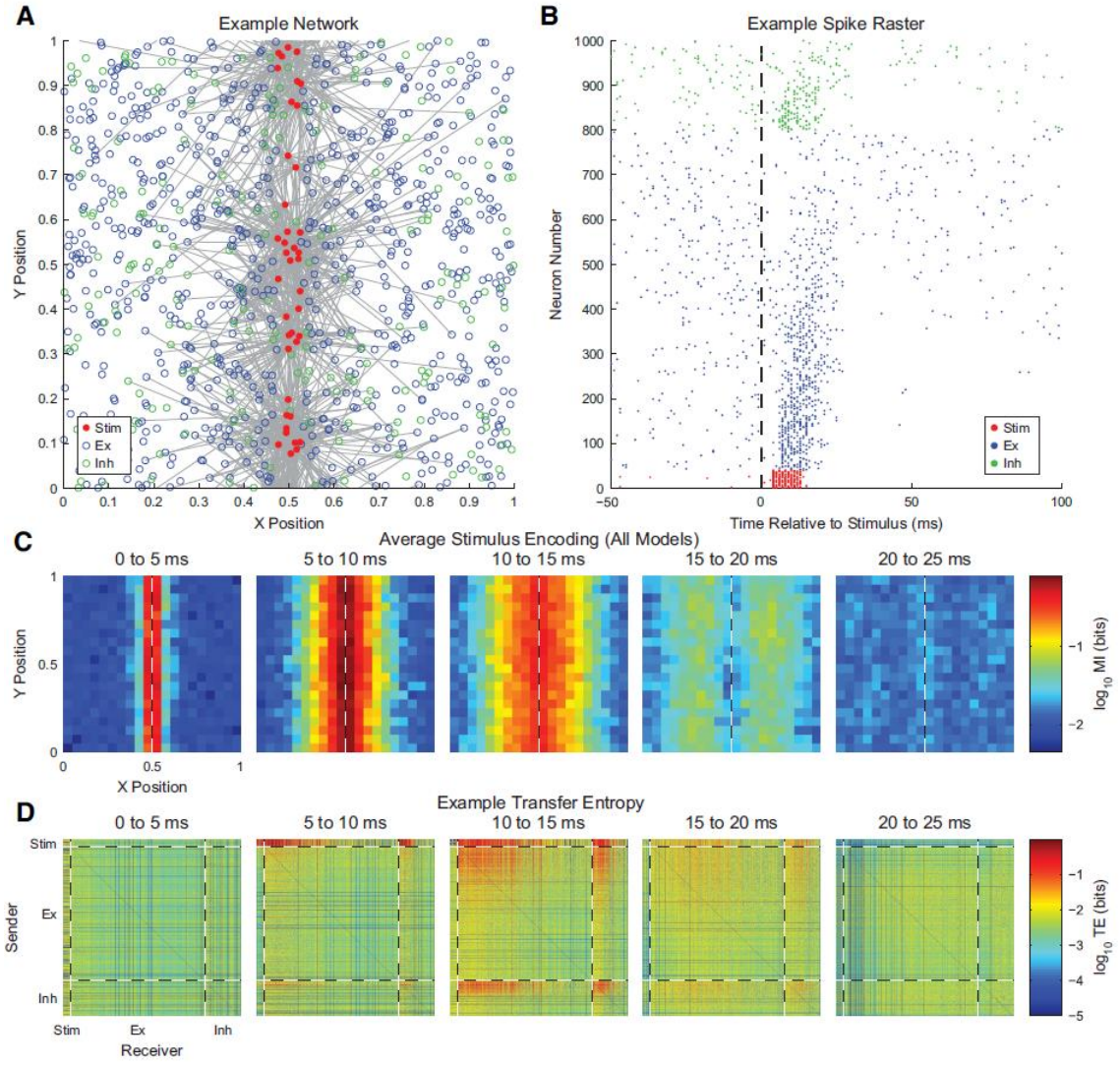



fig 4.4

## Small circuit example: unique information

The neural circuit responsible for the stretch reflex.
The stretch reflex involves a simple circuit comprising sensory neurons, interneurons, and motor neurons. This is what it includes:

1. Sensory Neurons: Specialized sensory receptors, called muscle spindles, detect changes in muscle length. When the muscle is stretched, these receptors send signals to the spinal cord.
2. Interneurons: In the spinal cord, these sensory signals are relayed to interneurons, which process the information locally. These interneurons play a crucial role in integrating sensory input and generating motor output.
3. Motor Neurons: Once the sensory information is processed, motor neurons are activated. These neurons send signals to the muscle causing it to contract, which results in the reflexive movement (e.g., the leg kicking out).

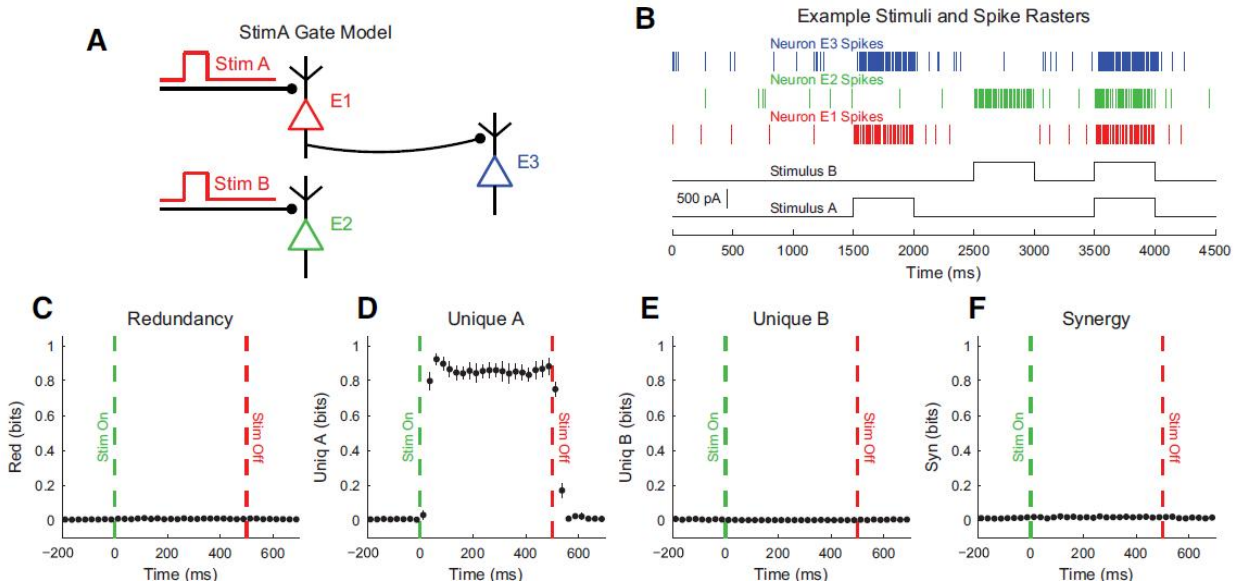



fig 4.5

## Small circuit example: synergy

It illustrates synergy in neuroscience: the neural circuit involved in coordinated movements of the hand.

1. Primary Motor Cortex (M1): This region of the cerebral cortex is crucial for initiating and coordinating voluntary movements. Neurons in M1 encode specific movement parameters such as direction, force, and speed.
2. Basal Ganglia: The basal ganglia are a group of interconnected subcortical nuclei involved in motor control. They receive input from various cortical areas, including M1, and play a key role in selecting and initiating appropriate movements while inhibiting unwanted movements. Dysfunctions in the basal ganglia can lead to movement disorders like Parkinson's disease or Huntington's disease.

3. Cerebellum: The cerebellum is involved in fine-tuning motor movements, ensuring their accuracy and coordination. It receives input from M1 and other brain regions, integrating sensory feedback to adjust ongoing movements. Damage to the cerebellum can result in ataxia and impaired coordination.

4. Spinal Cord: Neurons in the spinal cord receive signals from higher brain regions and coordinate muscle contractions to execute specific movements. Motor neurons innervate muscles, causing them to contract or relax in a coordinated manner.

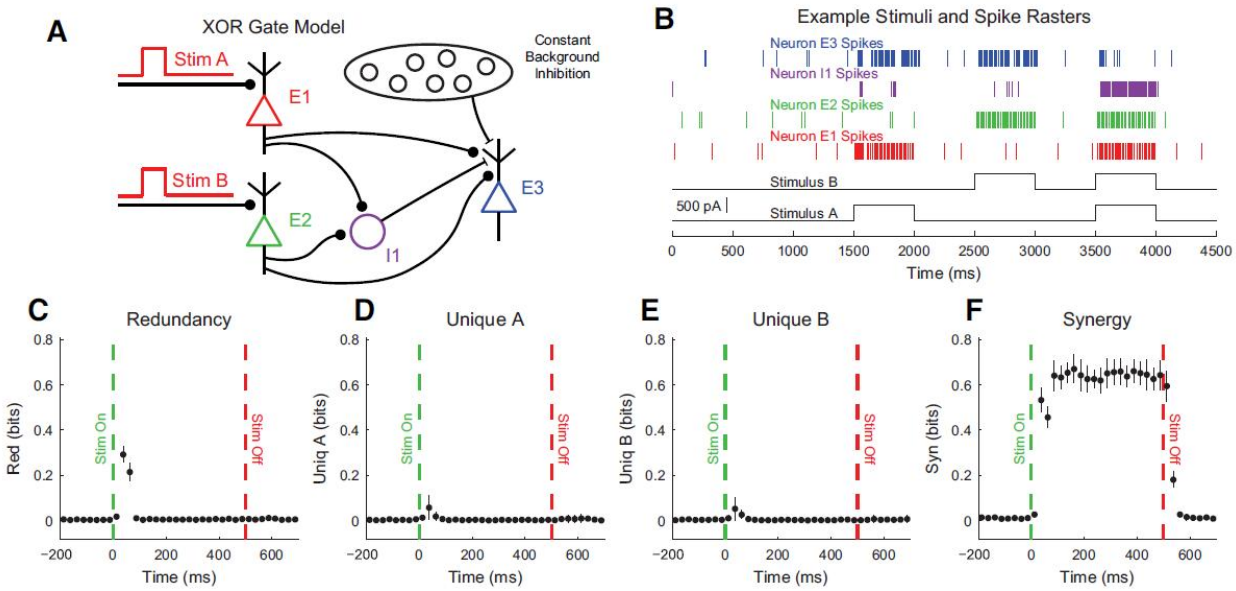



fig 4.6

## Small circuit examples: synergy and redundancy

Small circuit examples that highlight both synergy and redundancy in neuroscience:
Visual System:

- Synergy: The visual system comprises various interconnected structures, including the retina, lateral geniculate nucleus (LGN), primary visual cortex (V1), and higher visual areas. These regions work synergistically to process visual information, from basic features like edges and motion in V1 to complex visual perception in higher cortical areas.
- Redundancy: Redundancy is evident in the existence of parallel visual pathways, such as the magnocellular and parvocellular pathways, which convey different aspects of visual information (e.g., motion vs. color). This redundancy ensures robust visual processing and compensates for potential disruptions or lesions in one pathway.

**NeuroDemo 4:**

The code visualizes the results of a computational neuroscience simulation, likely focusing on spiking neurons in a network.

1. Setting Up:

- Defines colors (PColor, PColorGS) and line styles for plotting.
- Sets font sizes (LegFS, TitleFS, etc.) and line widths (VoltageLW, BPLW, etc.) for different elements.
- Defines names for performance measures (PIDNamesLong, PIDNamesShort).

2. Configuring the Layout:
- Sets up margins, spacing (hspace, vspace) and paper size for the figure.
- Calculates the width and height for each subplot based on the number of rows and columns.
- Determines the left and bottom coordinates for each subplot, considering margins.

3. Creating the Figure:
- Creates a figure (F1) with the specified dimensions and paper size.

4. Plotting Spike Rasters:
  - Focuses on a specific trial (iST).
  - Plots the stimulus currents (red and blue) on the y-axis.
  - Plots spike times from three neurons as lines on the y-axis with distinct colors.
  - Adjusts the y-axis to accommodate all traces.
  - Adds axis labels and a title for this subplot.

5. Plotting PID Through Time:
  - Iterates through different performance metrics (PIDs) and trials.
  - Finds the minimum and maximum values for the y-axis based on the data.
  - Sets the x-axis limits based on the simulation time points.
  - Loops through PIDs and trials, creating subplots for each combination.
  - Plots the mean error for the chosen PID and trial within each subplot.
  - Includes dashed lines to indicate stimulus on and off times.
  - Adds axis labels and titles for each subplot.

6. Plotting PID vs. Correlation:

- Similar to the previous section, finds minimum and maximum values for the y-axis based on the data.
- Sets the x-axis limits based on the correlation parameter between inputs.
- Loops through PIDs, creating subplots for each.
- Plots the variation in inhibition for the chosen PID at different correlation parameter values within each subplot.
- Adds axis labels and titles for each subplot.

7. Saving the Figure:

- Saves the entire figure (F1) as a PDF file named "neuroDemo4OR" in a specified directory.

Overall, the code creates a multi-panel figure that visualizes the performance of the spiking neural network model under various conditions. It displays how the network responds to different stimuli and how its performance relates to the correlation between inputs.

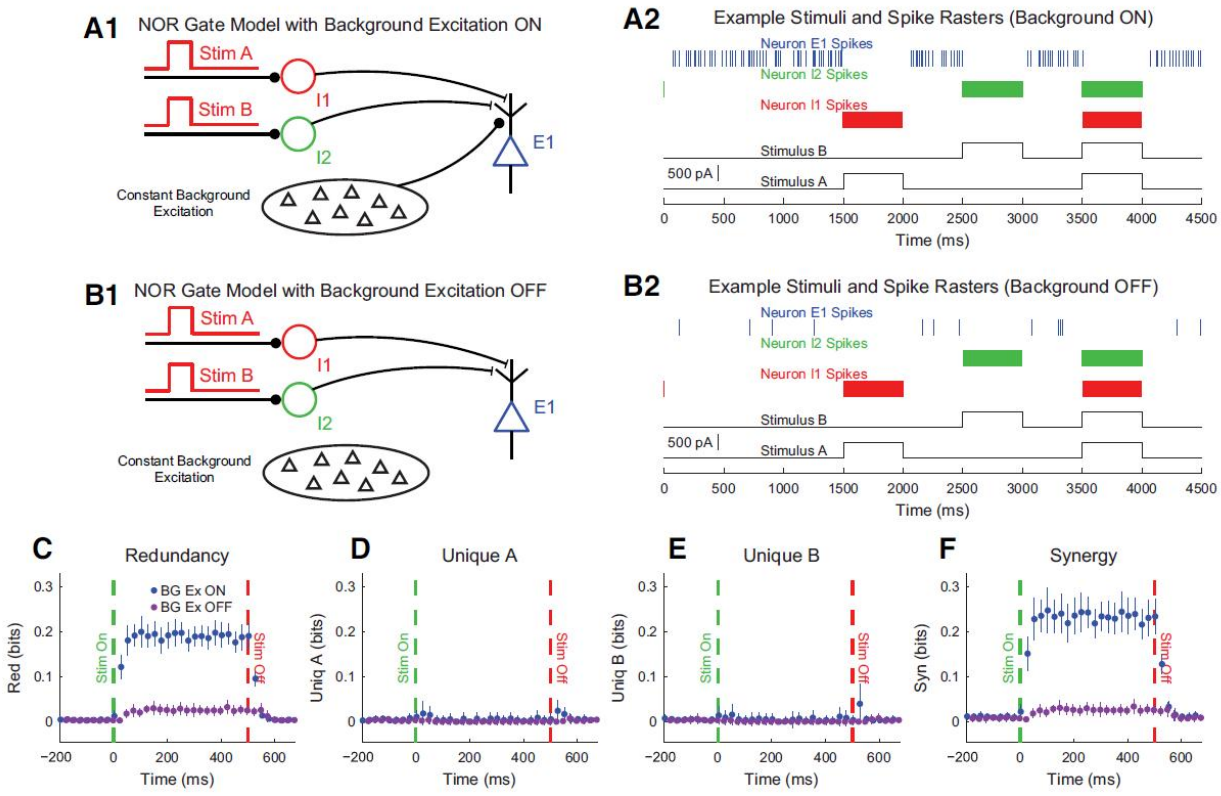



fig 4.7

# Synergy, redundancy, and unique information in a large network

**NeuroDemo 5:**

Part 1: Data Processing

1. Removing Empty Bins (Data Cleaning)

1. The code iterates through two data structures: PIDRadVals which stores binned data and PIDRadCounts which likely keeps track of the number of data points in each bin.
2. For each bin (iBin), it checks the corresponding value in PIDRadCounts. If the count is 1 (meaning only one data point), the entire row for that bin is removed from PIDRadVals. This ensures only bins with sufficient data are used for further analysis.
3. Similarly, the corresponding bin edge value from sEdges (which stores the boundaries of each bin) is also removed to maintain alignment with the cleaned data.

2. Calculating Summary Statistics per Bin:

1. The code loops through each bin (iBin) and each measurement (iMeas).
2. For the current bin and measurement, it extracts the corresponding data (temp = PIDRadVals(:,iBin,iMeas)).
3. It removes any Not a Number (NaN) values from the data (temp(~isfinite(temp)) = []).
4. Using the cleaned data (temp), it calculates the mean (mean(temp)), standard deviation (std(temp)), and confidence intervals for the mean (mean(temp) ± std(temp)/sqrt(length(temp))). These statistics summarize the distribution of data points within that bin.
5. The calculated statistics are then stored in the data structure DP14.PIDCurves.

3. Calculating Bin Center Locations:

This step calculates the center point of each bin by adding half the bin width (0.5*sBins) to the corresponding edge value (sEdges) and storing the results in DP14.dist. This provides a reference point for interpreting the binned data.

4. Extracting Spikes for Specific Stimulations:

1. The code first finds the file containing data for the first run of subtype 5 (iSubType = 5; iModel = 1;) using file naming conventions. It then loads the data from that file.
2. It identifies specific stimulations of interest (specStims) based on pre-defined criteria (e.g., first, middle, last stimulation of a type).
3. The code iterates through these chosen stimulations (iCase) and loops through the neurons (iNeuron).
4. For each stimulation and neuron combination, it extracts the spike times (temp = data.spkTimes{iNeuron}).
5. These spike times are then filtered to include only those occurring within the stimulation window and a short window before and after the stimulation. This ensures the extracted spikes are relevant to the specific stimulation being analyzed.
6. Finally, the filtered spikes are stored in the data structure DP12.spikes.

5. Calculating Error Bars for MI Values:

1. The code first determines the total number of time bins (nT).

2. It then iterates through each time bin (iT) and measurement (iMeas).
3. For the current time bin and measurement, it extracts the corresponding MI values (temp = MIValues(:,iT,iMeas)) for all models.
4. Similar to step 2, it calculates the mean (mean(temp)), standard deviation (std(temp)), and confidence intervals for the mean (mean(temp) ± std(temp)/sqrt(length(temp))) of the MI values across different models. This provides a measure of variability in MI across models for a specific time bin and measurement.
5. The calculated error bars (mean and confidence intervals) are stored in DP13.errorBarVals.

6. Saving the Data:
After all the processing is complete, the code saves all the processed data structures (DP1 to DP15) in a single file named ModelType5PlotData.mat. This file serves as the input for the figure generation part of the code (Part 2).

Part 2: Figure Generation

1. Setting Up Plots:
This section defines various parameters that control the visual appearance of the plots. Examples include setting margins between elements, spacing between subplots, font sizes for labels and titles, line widths for traces, and colors used for different data series. These parameters ensure a consistent and visually appealing presentation of the data.

2. Center-Out Task Figure:
1. This figure focuses on analyzing the response of the model to stimuli presented in different directions.
2. Direction Diagram: The code plots a circle with wedges representing different directions. The wedges are likely color-coded based on some data values in DP1.points. This provides a visual reference for the different directions used in

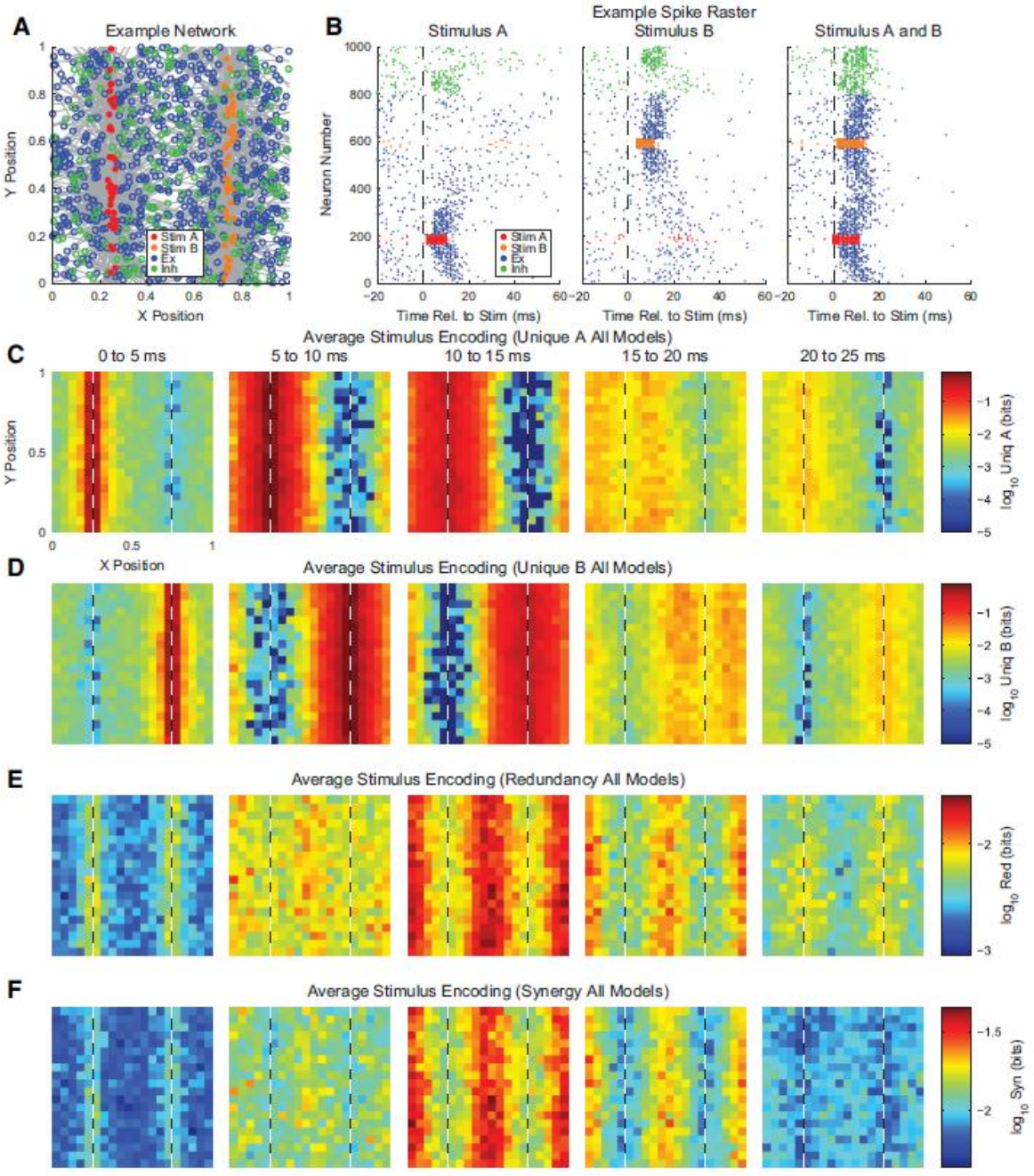
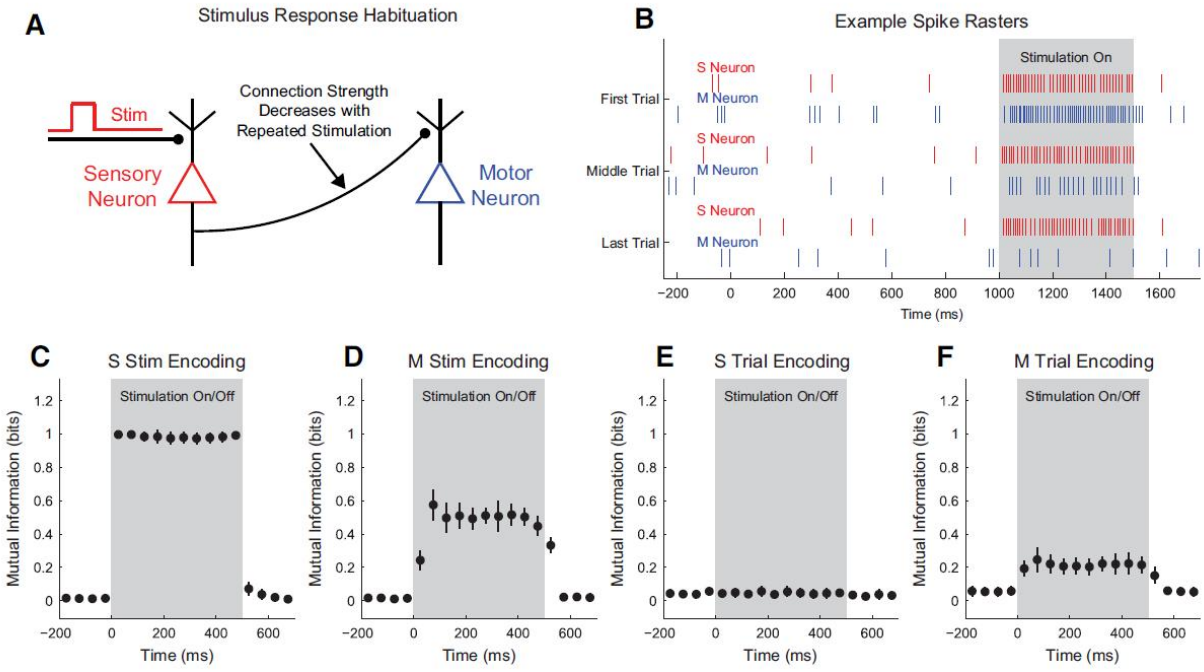

A
Example Network
Y Position
X Position
Stim A
Stim B
Ex
Inh
B
Example Spike Raster
Stimulus A
Stimulus B
Stimulus A and B
Neuron Number
Time Rel. to Stim (ms)
C
Average Stimulus Encoding (Unique A All Models)
0 to 5 ms
5 to 10 ms
10 to 15 ms
15 to 20 ms
20 to 25 ms
log10 Uniq A (bits)
D
Average Stimulus Encoding (Unique B All Models)
log10 Uniq B (bits)
E
Average Stimulus Encoding (Redundancy All Models)
log10 Red (bits)
F
Average Stimulus Encoding (Synergy All Models)
log10 Syn (bits)

fig 4.9

**Aplysia stimulus response habituation**

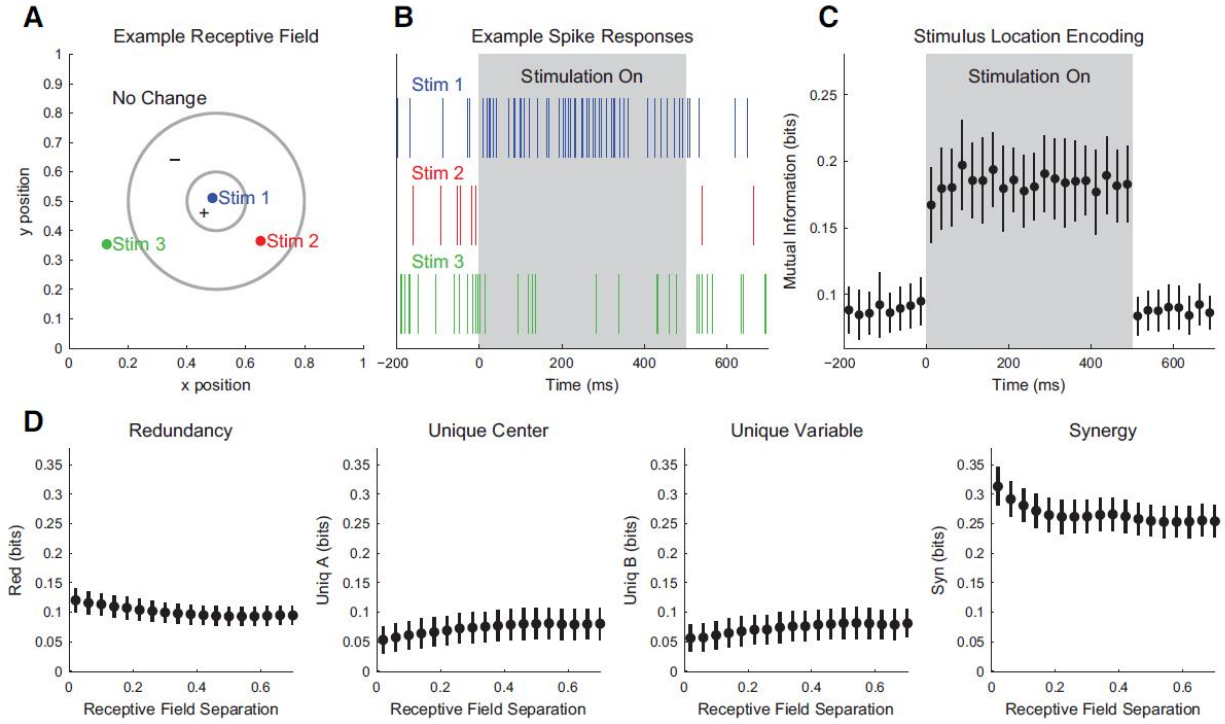
The Aplysia sea slug has been a valuable model organism in neuroscience for studying basic principles of learning and memory, including habituation, due to its relatively simple nervous system. Here's how habituation works in the context of Aplysia's stimulus-response system:

1. Stimulus.
2. Response.
3. Habituation.
4. Neural Mechanisms.
5. Long-Term Habituation.



## Center-surround retinal ganglion cells

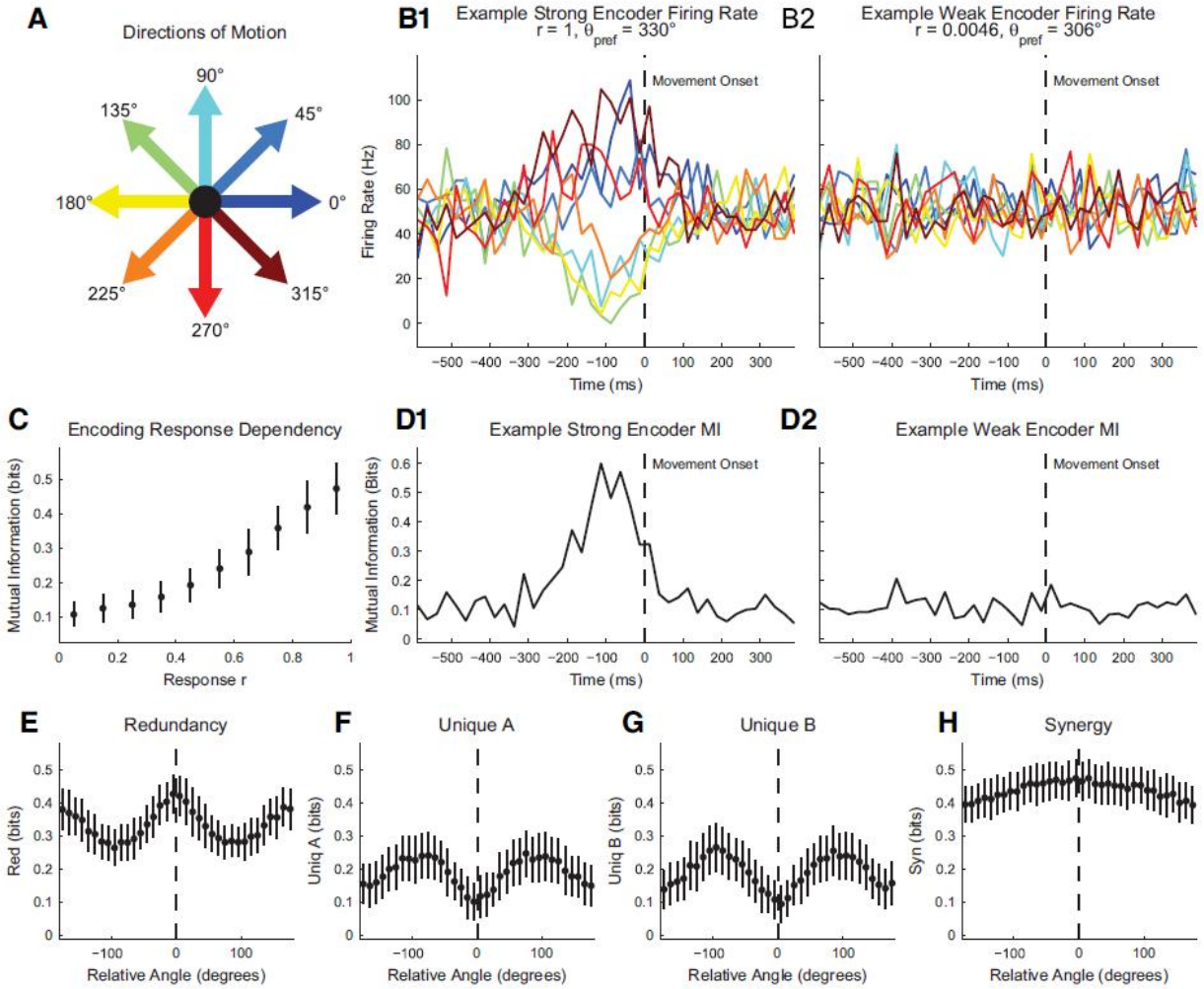
Center-surround retinal ganglion cells are a type of retinal ganglion cell (RGC) found in the retina of the vertebrate eye, including mammals like humans. These cells play a crucial role in processing visual information before it is transmitted to the brain via the optic nerve.



**Movement direction and motor cortex neurons**

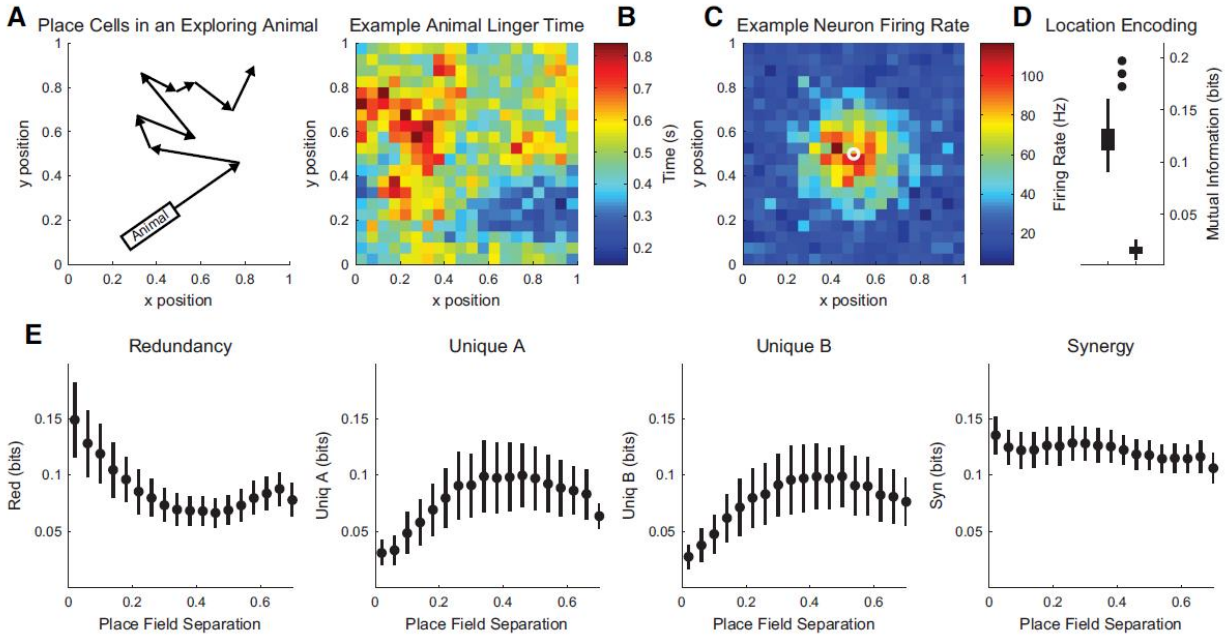
Motor Cortex Neurons:

1. Neurons in the motor cortex, especially in M1, are involved in planning, initiating, and executing voluntary movements. These neurons exhibit a variety of properties, including directional selectivity, meaning they are preferentially activated by movements in specific directions.
2. Motor cortex neurons are organized somatotopically, meaning different parts of M1 control movements of specific body parts. For example, neurons in one region of M1 may control movements of the hand, while neurons in another region may control movements of the arm or face.



## Place cells

Place cells are neurons found in the hippocampus that are selectively active when an animal is in a specific location within its environment. They were first discovered by John O'Keefe and Jonathan Dostrovsky in 1971 while studying the neural activity of rats as they navigated through different environments.



# 5. FUTURE ENHANCEMENT AND CONCLUSION

## 5.1 Conclusion

In summary, this project offers a holistic strategy to meet the imperative requirement for meticulous validation of research outcomes in the realm of information theory and neuroscience. Through systematic parameter alterations and the integration of experimental and computational approaches, we have evaluated the resilience and replicability of the initial study's findings. Our results underscore the necessity of rigorous validation protocols in advancing understanding in this domain, pinpointing avenues for future inquiry and stressing the pivotal role of replication endeavors in safeguarding the credibility of scientific progress.

## 5.2 Future Enhancements

While the demonstrations we employed throughout this project were focused on neural spiking, clearly many other types of data are used widely in neuroscience. We chose to focus on neural spiking data due to our expertise with it. However, we wish to emphasize that nearly identical analyses could easily be performed with BOLD signal data from fMRI studies, fluorescence data from calcium imaging studies, or voltage signals from extracellular, EEG, or MEG studies. Certainly, sampling constraints (e.g., slower time resolution in fMRI and calcium imaging) or other pre-processing steps (e.g., initial power spectrum decomposition in EEG) would alter the results of the analyses or the precise details of how they were applied, but the distribution of voltage values, BOLD signals, or fluorescence signals across trials or time can just as easily be discretized as spike counts using similar methods.

Indeed, even other types of data that are conceptually different from the various measures of neural activity discussed above can be easily treated with these information theory tools. Information theory is currently used in genetics, but studies could be performed linking genetics and neuroscience.

# 6. REFERENCES


1. **https://www.physionet.org/content/eegmmidb/1.0.0/**

2. **https://bilalzonjy.github.io/EDFViewer/EDFViewer.html**

3. **https://www.renesas.com/us/en/support/engineer-school/noise-estimating-calculators#downloads**

4. Information Theory and Machine Learning | MDPI Books