Reimplementing the Interactive Activation and Competition Model: Representing Human Memory

Simone Rittenhouse sgr344@nyu.edu
Mariam Abdullah ma3259@nyu.edu
Zirui Liu zl5162@nyu.edu
Haitong Zhu hz3916@nyu.edu

Center for Data Science, New York University

Abstract

The Interactive Activation and Competition (IAC) model, introduced by McClelland (1981), simulates memory processes through dynamic network interactions. In this paper, we present a Python-based re-implementation of the IAC model, replicating results from McClelland's original Jets vs. Sharks dataset and extending the model to U.S. Census data. Our results demonstrate the model's ability to replicate key cognitive properties and explore complex relationships between demographic and income attributes while updating the IAC model to Python code, making it more accessible and versatile for modern research applications.

Introduction

Memory — the process through which humans store, retrieve, and apply information — is fundamental to cognition and learning. Understanding the mechanics of memory is not only essential for advancing cognitive science but also for building computational models that simulate human cognition. One such model, the Interactive Activation and Competition (IAC) network (McClelland, 1981) offers a robust approach to simulate memory as a dynamic network of interconnected units. These units interact based on input patterns, effectively capturing key cognitive properties such as content addressability, pattern completion, and spontaneous generalization. This implementation, originally performed on a West Side Story inspired Jets vs. Sharks dataset, demonstrates how the IAC model could resolve ambiguous stimuli.

While McClelland's 1981 IAC model offers important insights, it has remained limited in its implementation. Many implementations are limited to the McClelland's original Jets vs. Sharks dataset, leaving the broader potential of the model underexplored. Additionally, previous implementations of the IAC model are written in Java and MATLAB, offering performance advantages at the cost of steep learning curves and difficult integration, particularly as the data science community increasingly shifts to Python and R.

Accordingly, our study seeks to re-implement the model in Python using McClelland's original equations and explore its application to a novel dataset. Our key contributions include a Jupyter Notebook compatible interactive visualization tool inspired by Axel Cleeremans' Java-based IAC runtime (Cleeremans, n.d.), replication results using McClelland's Jets vs. Sharks dataset, and an evaluation of the model's ability to simulate core memory properties within a novel domain.

We intend to open-source the code for easier adaptation in future research. Our public GitHub repository can be found here: IACModel_CCM.

Related Work

The IAC model demonstrates how network units process information through excitatory and inhibitory interactions. Grossberg (1978) proposed that these connections help organisms manage fluctuating input patterns without becoming overwhelmed or losing the patterns to distortion through noise. The interactive systems outlined in his work offer explanations for several phenomena in sensory processing, including memory.

Within the framework of memory, Grossberg describes network units as hierarchically organized feature detectors, each responsive to particular patterns. When fed an input pattern, activated detectors compete with one another through inhibitory connections. The most successful dectectors then store their activities in short-term memory. Detectors also respond to derived patterns which have not directly observed, supporting memory properties like default assignment, where humans infer plausible but unseen information by linking new stimuli to similar past experiences.

Expanding upon the work of Grossberg, McClelland and Rumelhart (1981) first used an interactive activation and competition mechanism to explore letter detection. Their model features hierarchical processing levels, including visual, letter, and word detectors. Activating one word suppresses competing words while reinforcing relevant letter representations through feedback. Their findings correspond with human cognitive studies, underscoring the IAC model's broad applicability to cognitive phenomena.

Beyond the applications of Grossberg and McClelland, Hofmann and Jacobs (2014) found that IAC models can account for implicit memory effects in behavioral tasks and neuroimaging studies. Their findings demonstrate that computational models like IAC explain cognitive mechanisms while serving as heuristic tools to bridge cognitive and neural levels in cognitive neuroscience.

Background

The architecture and key properties of the IAC model are highlighted below.

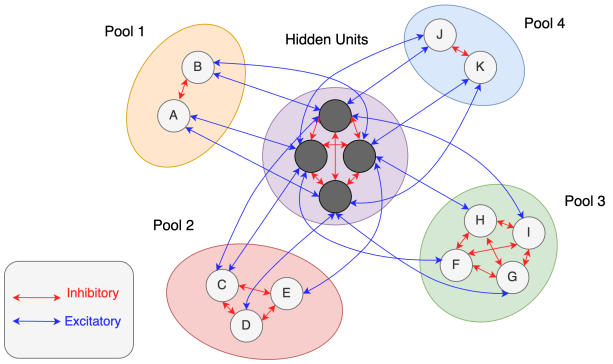

Figure 1: Overview of IAC architecture

Interactive Activation and Competition Models. Fundamentally, an IAC network consists of several units organized into competitive pools, as shown in Figure 1. The connections between pools, known as projections, can be excitatory *or* inhibitory. Each pool has an inhibitory self-projection, meaning units compete with their in-pool neighbors. There may also be excitatory projections between pools, which are assumed to be bidirectional, making these responses "interactive" as changes in one pool are both influenced by and influence connected pools. The inhibitory self-projections cause competition – i.e., strongly activated units will more strongly drive down the activations of their in-pool neighbors.

There is an additional pool, the bias pool, that contains a single bias unit. This unit is always active and can take different connection weights with other units in the network, essentially adding a constant term to the inputs for all other units. In McClelland's original network (McClelland, 1981), the bias is assumed to be zero. We adopt the same approach for simplicity and usability, avoiding the need for manual user input for unit-specific bias terms.

The activation of units within the network evolves continuously over time, influenced by the following: the unit's current activation and the unit's net input. To formalize the change in the activation of a unit, we must first define the following network parameters:

1. $\boldsymbol{max}$ – The maximum possible activation value of a unit.
2. $\boldsymbol{min}$ – The minimum possible activation value of a unit.
3. $\boldsymbol{rest}$ – Units' resting activation. The activations of units settle to this value without external input.
4. $\boldsymbol{decay}$ – The rate at which units return to their resting activation.
5. $\boldsymbol{estr}$ – The weight of external input to network units.
6. $\boldsymbol{\alpha}$ – The weight of internal excitatory input to network units.
7. $\boldsymbol{\gamma}$ – The weight of internal inhibitory input to network units.

With these parameters defined, the change in a unit's activation, a_i, can be expressed as

$$\Delta a_i = \begin{cases} (max - a_i)\text{net}_i - decay(a_i - rest), & \text{net}_i > 0 \\ (a_i - min)\text{net}_i - decay(a_i - rest), & \text{otherwise} \end{cases}$$

Where net_i is the total input for unit i. This is the sum of the input to the unit from neighboring (i.e. connected) units and any external input from outside of the network. The net input to unit i can be formalized as

$$\text{net}_i = \Sigma_j w_{ij} output_j + extinput_i$$

Where w_{ij} is the strength of the connection between unit i and unit j, $output_j$ is the output of unit j, and $extinput_i$ is the external input into unit i. Here, $output_j$ is defined by

$$output_j = \begin{cases} a_j, & a_j > 0 \\ 0, & \text{otherwise} \end{cases}$$

Where a_j is the activation of unit j. In McClelland and Rumelhart (1988), net_i may also include a Gaussian noise term to implement some of the inherent variability of human data; however, it is excluded here for the sake of simplicity.

The sum of inputs from neighboring units, $\sum_j w_{ij} output_j$, can further be broken down into excitatory and inhibitory connections. In this formulation, $inhib_i$ is the sum of all $w_{ij} output_j$ such that j shares an inhibitory connection with i. Similarly, $excit_i$ is the sum of all $w_{ij} output_j$ such that j shares an excitatory connection with i. The net input then becomes:

$$\text{net}_i = \boldsymbol{\alpha}(excit_i) + \boldsymbol{\gamma}(inhib_i) + \boldsymbol{estr}(ext_i)$$

Where $extinput_i = \boldsymbol{estr}(ext_i)$.

Within an IAC model, there are two kinds of units, visible and hidden (Figure 1). Users can provide external input to visible units, but by default, it is assumed that users cannot provide such input to hidden units. This means that $extinput_i$ is set to zero for any hidden units i, and their net input relies solely on inputs from neighboring units.

Key Properties. When modeling memory, the IAC model demonstrates several key properties that align with characteristics of human cognition, which we define as:

1. **Retrieval by Name:** The model recalls specific details when queried with a direct label, similar to remembering details about a person when hearing their name.
2. **Content Addressability:** The model retrieves all linked information (i.e., pool units connected through the same hidden unit) even when given only partial input, similar to our ability to recall someone's name when given related features.
3. **Graceful Degradation:** The model's ability to retrieve linked information decreases slowly as input becomes more noisy (e.g., as more unrelated input is given), akin to how human memory weakens progressively in the presence of noise but rarely fails completely.

4. **Default Assignment:** The model infers missing details by associating similar features from related units, filling in gaps based on known similarities.

5. **Spontaneous Generalization:** The model identifies common features across multiple examples, forming generalized category representations, much like human conceptual learning.

Methodology

Our implementation uses a modular Python architecture, with two core files: `IAC_plotting` and `IAC_example`.

The `IAC_plotting` script houses the core plotting logic. The function `plot` takes in a DataFrame and a column name representing the network's hidden units. It then runs a Dash app for the visualization, implementing three key tasks: initialization, simulation, and rendering.

This script is importable to a Jupyter Notebook for execution. `IAC_example` contains an example of this as well as our application of the model to novel domains.

Visualization Design

The system begins by mapping unique DataFrame entries to a corresponding unit. Columns define cognitive categories, organizing units into pools. Nodes are then created and stored in an undirected graph, with edges established between hidden and visible units based on the rows of the DataFrame. Inhibitory edges exist within the same pool, fostering competitive dynamics, while inter-pool edges are excitatory. A spatial arrangement algorithm taken from Cleeremans (n.d.)'s implementation organizes nodes into concentric rings, ensuring minimal edge overlap and highlighting inter-pool relationships.

Using the same model parameters outlined by McClelland and Rumelhart (1988), we define the following:

- **Activation Bounds:** $max = 1.0, min = -0.2$,
- **Resting Activation:** $rest = -0.1$
- **Decay Rate:** $decay = 0.1$
- **Connection Weights:** $estr = 0.4, \alpha = 0.1, \gamma = 0.1$

Simulations on the model are performed after a user selects one or more units, specifies a number of update cycles, and clicks "Run Simulation". This changes units' activations following the update rule (Δa_i) outlined previously. The detailed simulation algorithm is provided in the Appendix (Algorithm 1).

The corresponding network visualization is built using Plotly and Dash. Node color indicates activation magnitude and pool membership, with pool-based colors for units near resting activation, blue shades for high excitatory activations, and red shades for high inhibitory activations. Color saturation reflects the unit's deviation from its resting state. Edge color denotes connection type: blue for excitatory and red for inhibitory links.

The interface updates the network in real time based on user inputs. Clicking a node selects it, applying an external input of 1.0 and changing the fill color to bright green. A control panel enables simulation adjustments, resets, and new runs. Hovering reveals details such as activation level, net input, and connections. The default visualization for McClelland's West Side Story dataset appears in the Appendix (Figure 5).

To recap, the below function trace offers a succinct view into how the model works. All of these steps are contained within the function call `plot(df, hidden_state=None)`, which runs the visualization:

- **Initialization:**
  - `init_graph(df, hidden_state=None)`: Returns dictionary `pools` mapping units to pools, dictionary `pos` mapping units to positions, and graph object `G` with nodes and edges. Argument `hidden_state` is a string value of the column to use as hidden units within the network. If this argument is not passed, the first column is used.
    - `position_nodes(pools)`: Creates `pos` dictionary mapping units to (x,y) coordinates in concentric rings.
- **Simulation:**
  - `run_simulation(G, pools, unit_info, clicked_nodes, num_cycles)`: Updates unit activations and net inputs stored in `unit_info` as well as edge weights in `G` by running `num_cycles` network updates.
    - `get_netinput(node, G, pools, unit_info, extinput)`: Calculates net input of unit `node`.
- **Visualization:**
  - `fig_update(hoverData, clickData, n_clicks_reset, n_clicks_sim, num_cycles, selected_data, figure, graph_data, unit_info)`: Updates the figure based on user interaction.
  - `create_plot(pos, pools, G, unit_info, hover_node=None)`: Builds a Plotly figure representing the current state of the network.
    - `activation_to_color(act, default_colors, maxact=MAX, minact=MIN, restact=REST)`: Maps unit to fill colors depending on the strength and direction of their activation. If $|(act_i - restact)| < 0.05$, the unit keeps its default, pool-based color.

Data

To re-implement McClelland's original model, we used the Jets and Sharks dataset (McClelland, 1981). This dataset consists of 27 character names which compose 27 hidden units in the IAC model. Six visible pools correspond to these

hidden units: the character's name, gang affiliation, age group, highest level of education, marital status, and occupational title.

For the novel application component, we sourced a machine learning dataset that uses U.S. Census data to predict whether an individual earns over $50,000 (*Adult Census Income*, n.d.). The dataset contains 32,561 observations across 15 features.

To ensure each hidden unit connects to only a single visible unit within each pool, categorical features were sorted by unique values, with education level (15 categories) selected as the hidden layer. A random sample ensured unique education-level entries per row. After dropping several columns, eight visible pools remained: age, work class, education level, marital status, occupational title, sex, hours worked per week, and income. Pre-processing involved bucketing age and hours worked, as well as abbreviating labels (e.g., `Self-emp-not-inc` to `Self-emp`) to enhance visualization clarity and reduce clutter.

Results

Replication Results. To evaluate the quality of our visualization against Cleeremans (n.d.), we ran two comparative network updates. Importantly, Cleeremans' network parameters remain unknown, making exact replication difficult.

The first experiment selects the name "Ken" to retrieve associated attributes after 150 update cycles.

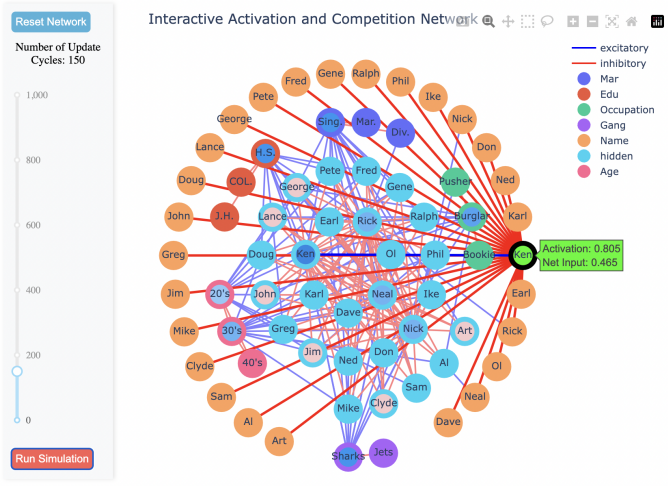

Figure 2: IAC Model at 150 cycles for external input "Ken".

The result of this update is shown in Figure 3. We find similar patterns of activations between the two implementations, with related attributes such as "Burglar" and "Sharks" receiving higher activation. Note that in Table 1, we observe a stronger activation for the age group "30s" than "20s", despite Ken belonging to the latter group. This can be attributed to the activations of the hidden units "Nick", "Neal", and "Rick", which all belong to the "30s" age group and thus pass excitatory connections to that unit. We also observe that these hidden units are more strongly activated than with the network

Table 1: Activated units for external input "Ken" at 150 cycles.

Unit	Current Model	Cleeremans' Model
	Act. / Net Input	Act. / Net Input
Name: Ken	0.805 / 0.465	0.886 / 0.435
Hidden: Ken	0.646 / 0.208	0.692 / 0.128
High School	0.584 / 0.166	0.523 / 0.066
Sharks	0.584 / 0.166	0.495 / 0.060
Single	0.540 / 0.140	0.495 / 0.060
Burglar	0.419 / 0.091	0.395 / 0.040
30s	0.378 / 0.084	-0.080 / 0.002
Hidden: Nick	0.378 / 0.079	0.194 / 0.020
Hidden: Neal	0.378 / 0.079	0.194 / 0.020
Hidden: Rick	0.269 / 0.056	0.058 / 0.008
20s	0.170 / 0.027	0.395 / 0.040

parameters used by Axel Cleereman, explaining the much lower activation of the "30s" unit within his implementation.

Our second comparative experiment selects "Sharks" and, as before, runs both models for 150 update cycles. The resulting network is shown below:

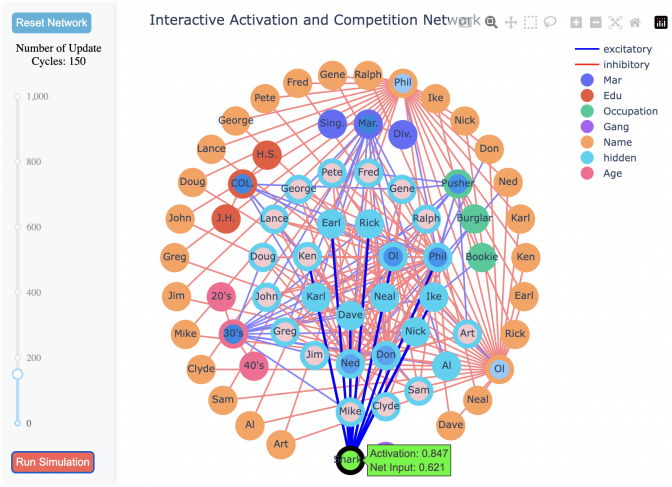

Figure 3: IAC Model at 150 cycles for external input "Sharks".

We can see in Table 2 that, although the two implementations differ in their activations and net inputs, they follow the same trends and result in the same activated units. This result also serves as an example of spontaneous generalization as the network creates a representation of the category "Shark" by activating typically associated units such as being approximately 30 years old, college educated, and married.

New Domain Results. Here, we aim to analyze the relationships between education levels and other features in a U.S. Census dataset, such as income, age, marital status, and occupation, as visualized through the IAC model. The goal is to understand how categorical features interact within the network and contribute to income prediction.

As shown in Table 3 and Figure 4, we analyze income

Table 2: Activated units for external input "Sharks" at 150 cycles.

Unit	Current Model	Cleeremans' Model
	Act. / Net Input	Act. / Net Input
Sharks	0.847 / 0.621	0.902 / 0.512
30s	0.657 / 0.221	0.660 / 0.112
College	0.657 / 0.221	0.660 / 0.112
Married	0.657 / 0.221	0.660 / 0.112
Hidden: Phil	0.630 / 0.198	0.636 / 0.101
Hidden: Ol	0.630 / 0.198	0.636 / 0.101
Pusher	0.513 / 0.126	0.516 / 0.064
Hidden: Ned	0.472 / 0.108	0.483 / 0.056
Hidden: Don	0.472 / 0.108	0.483 / 0.056
Name: Phil	0.220 / 0.041	0.222 / 0.021
Name: Ol	0.220 / 0.041	0.222 / 0.021

Table 3: Activated units for external input ">50k" at 500 cycles.

Unit	Activation / Net Input
>50k	0.809 / 0.476
Private	0.700 / 0.266
Married	0.700 / 0.266
Male	0.700 / 0.266
Hidden: Assoc.	0.695 / 0.261
Hours per week: [40, 50)	0.660 / 0.107
Hidden: 5th-6th	0.625 / 0.193
Hidden: 1st-4th	0.502 / 0.121
Age: [40, 50)	0.470 / 0.107
Hidden: Bachelors	0.400 / 0.092
Hidden: Masters	0.413 / 0.087
≤50k	0.365 / 0.073
Education: Assoc.	0.270 / 0.051
IT	0.270 / 0.051
Age: [50, 60)	0.246 / 0.046
Transport	0.188 / 0.035
Education: 5th-6th	0.188 / 0.035

above $50,000 by examining the activations and net inputs of related units. For example, we find high activation in "Male" (activation: 0.700), suggesting a connection between being male and having a high income. Units relating to occupation like "Private" and "IT" also show moderate to high activations. This reflects their positive influence on income predictions, given the role of stable and skilled employment in determining income levels. This conclusion is further demonstrated by the activation of working 40 to 50 hours per week, indicating higher income for those with full-time employment. Older age groups are also activated (40-50 years, activation: 0.470; 50-60 years, activation: 0.246) suggesting that those with longer careers are higher earners.

To explore the impact of age on income more thoroughly, we ran simulations selecting >50k and one of the following age groups: 30-40 years, 40-50 years, and 50-60 years. The resulting network states are shown in the Appendix (6, 7, 8). When age groups are activated alongside the income level ">50k," a few significant patterns emerge. In the age group [30-40), the occupations "Cleaner" and "Other" showed noticeable activation, pointing to their connection with this income level. For the [40-50) age group, "Transport" and "IT" roles were activated, suggesting that these jobs play a role in earning higher incomes. Lastly, the [50-60) group had activations in "Craft" and "Manager", highlighting the importance of skilled trades and managerial positions for this age range.

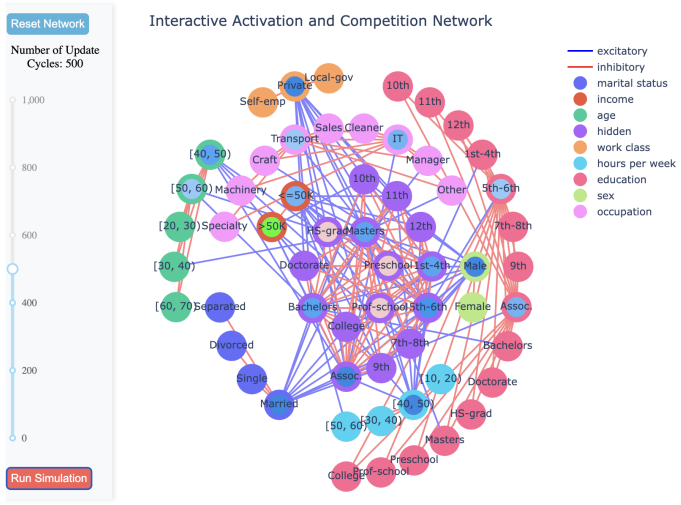

Figure 4: IAC Model at 500 cycles for external input ">50k".

Discussion

Our re-implementation of the IAC model highlights its adaptability to modern datasets and research paradigms. By translating the original IAC model into Python and applying it to U.S. Census data, we validate its ability to replicate core cognitive phenomena such as content addressability, pattern completion, and spontaneous generalization. Our findings demonstrate the model's potential to extend beyond its original scope, bridging cognitive modeling with contemporary data science.

A key outcome is the IAC model's capacity to identify and quantify interactions between categorical variables, including gender, marital status, occupation, and income. Our results align with established socio-economic research, illustrating how dynamic, interconnected systems can yield interpretable insights from multi-dimensional datasets. The model's ability to handle ambiguous or conflicting stimuli, reinforces its

strength in resolving uncertainties within datasets, whether by overlapping or partial information. By providing interactive visual representations, our work supports a deeper understanding of the model's activation flow and competition mechanisms, making it an effective teaching tool for cognitive science.

Future Work

Despite these strengths, several limitations warrant future exploration. The IAC model's behavior is highly sensitive to parameter settings, currently set manually. This approach lacks systematic optimization, potentially restricting adaptability to new datasets. Automated parameter selection would enhance both applicability and theoretical clarity.

Additionally, the model's static connection weights hinder its ability to adapt dynamically to evolving environments, limiting its suitability for tasks requiring real-time learning or multi-modal integration. Support for multi-modal data types, such as text and images, could extend the model's applicability to more complex cognitive and real-world problems.

Its reliance on fully connected architectures also increases computational overhead, reducing scalability for large datasets. While the model's interpretability is a key strength in cognitive modeling, applying it to datasets with thousands of variables introduces challenges related to high-dimensional features and data pre-processing. Optimizations such as sparse matrix representations or GPU acceleration could significantly improve scalability.

Lastly, additional visualization features such as allowing custom input for network parameters and showing intermediate simulation cycle updates could enhance user experience by allowing for more clarity into the network's update behavior and parameter effects.

By addressing these limitations, the IAC model could become a more robust tool for both teaching and research, bridging theoretical principles and practical applications.

In summary, this project affirms the relevance of the IAC model in contemporary research while modernizing its implementation to meet current needs. By validating its cognitive characteristics and extending its application to new domains, the current study lays the groundwork for further innovations.

References

Adult census income. (n.d.). https://www.kaggle.com/datasets/uciml/adult-census-income/data.

Cleeremans, A. (n.d.). *Interactive activation application.* https://axc.ulb.be/pages/interactive-activation-application.

Grossberg, S. (1978). A theory of visual coding, memory, and development. In E. L. J. Leeuwenberg & H. F. J. M. Buffart (Eds.), *Formal theories of visual perception.* New York: John Wiley & Sons.

Hofmann, M. J., & Jacobs, A. M. (2014). Interactive activation and competition models and semantic context: From behavioral to brain data. *Neuroscience Biobehavioral Reviews*, *46*, 85-104. Retrieved from https://www.sciencedirect.com/science/article/pii/S0149763414001493 (Micro- and Macro-Perspectives on Cognitive Conflict Control) doi: https://doi.org/10.1016/j.neubiorev.2014.06.011

McClelland, J. L. (1981). *Retrieving general and specific information from stored knowledge of specifics.*

McClelland, J. L., & Rumelhart, D. E. (1981). Interactive activation and competition: A model for context effects in letter perception: Part 1. an account of basic findings. *Psychological Review*, *88*(5), 375–407.

McClelland, J. L., & Rumelhart, D. E. (1988). Explorations in parallel distributed processing: A handbook of models, programs, and exercises. In *Computational models of cognition and perception.* Cambridge, MA, US: The MIT Press.

Appendix

Algorithm 1 IAC Model Simulation Update

for $i \leftarrow 1$ to num_cycles **do**
  for node $n \in$ Graph **do** ▷ Update unit net inputs
    if n is selected **then**
      $extinput \leftarrow 1.0$
    else
      $extinput \leftarrow 0.0$
    end if
    inhibitory $\leftarrow 0.0$, excitatory $\leftarrow 0.0$
    for $j \in$ node neighbors **do**
      if $a_j > 0$ **then**
        if $pools[n] = pools[j]$ **then**
          inhibitory $\leftarrow$ inhibitory $+(-1.0 \cdot a_j)$
        else if $pools[n] \neq pools[j]$ **then**
          excitatory $\leftarrow$ excitatory $+(1.0 \cdot a_j)$
        end if
      end if
    end for
    $\text{net}_n \leftarrow \boldsymbol{\gamma}(\text{inhibitory}) + \boldsymbol{\alpha}(\text{excitatory}) + \boldsymbol{estr}(extinput)$
  end for
  for node $n \in$ Graph **do** ▷ Update unit activations
    if $\text{net}_n > 0$ **then**
      $\Delta a_n = (\boldsymbol{max} - \text{net}_n)\text{net}_n - \boldsymbol{decay}(\text{net}_n - \boldsymbol{rest})$
    else
      $\Delta a_n = (\text{net}_n - \boldsymbol{min})\text{net}_n - \boldsymbol{decay}(\text{net}_n - \boldsymbol{rest})$
    end if
    $a_n \leftarrow a_n + \Delta a_n$
  end for
end for

Default Visualization

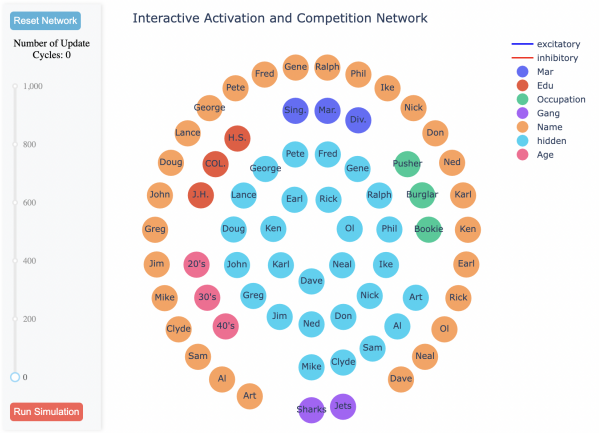

Figure 5: Default IAC Model for Jets and Sharks data.

Census Age Visualizations

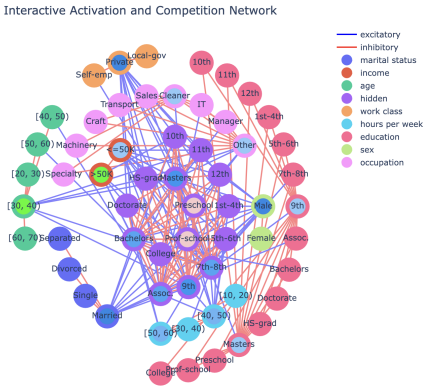

Figure 6: IAC Model at 500 cycles for external input ">50k" and age "[30, 40)".

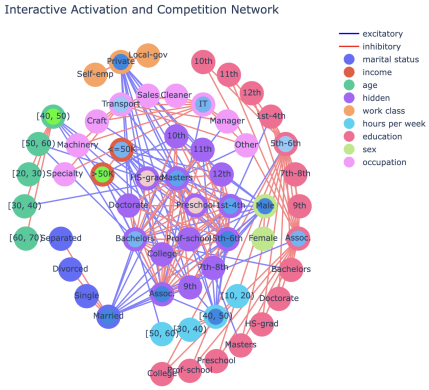

Figure 7: IAC Model at 500 cycles for external input ">50k" and age "[40, 50)".

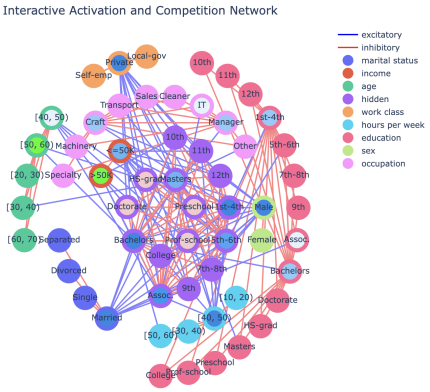

Figure 8: IAC Model at 500 cycles for external input ">50k" and age "[50, 60)".