Essays in Computational Management Science

Marcelo Salhab Brogliato

EBAPE / FGV

June 14, 2018

Introduction

Modern management and high technology interact in multiple, profound, ways. A whole new ecosystem seems to have emerged within computing and business.

The corporate biography of Tonny Martins, President of IBM Brasil, mentions his successes with blockchain, AI, and cognitive technology *as an executive*, not as a research scientist or specialized engineer.

Professor Andrew Ng tells students at Stanford's Graduate School of Business that "AI is the new electricity", as his hyperbolic way to emphasize the potential transformational power of the technology.

Outline

I Hathor: An alternative towards a scalable cryptocurrency

II An invitation to Sparse Distributed Memory: from the theoretical model to the system dynamics

III Diffusion and dismissal of innovation: forecasting the number of Facebook's active users

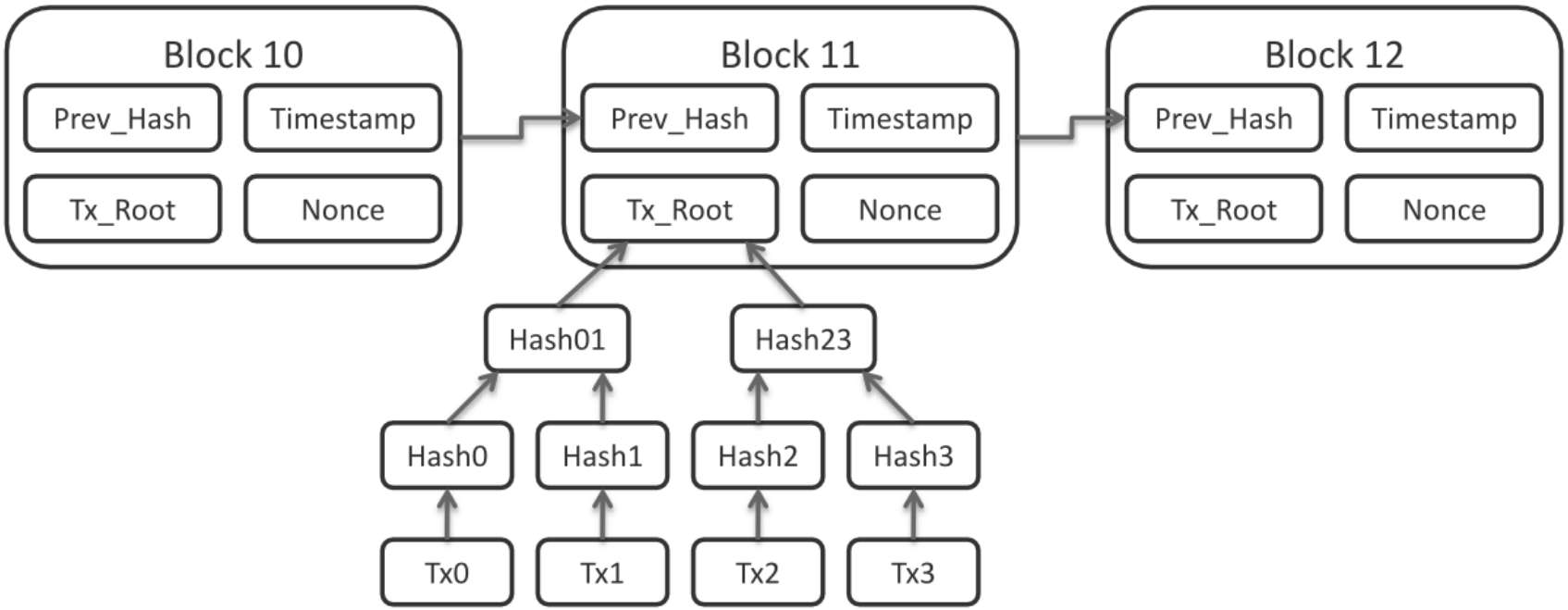
Hathor: An alternative towards a scalable cryptocurrency

The primary problem for creating digital money is how to prevent double spending.

As the money is digital, and copies can be made *ad nauseam*, what can prevent counterfeiting? What would prevent users from sending copies of the same money to two (or more) people?

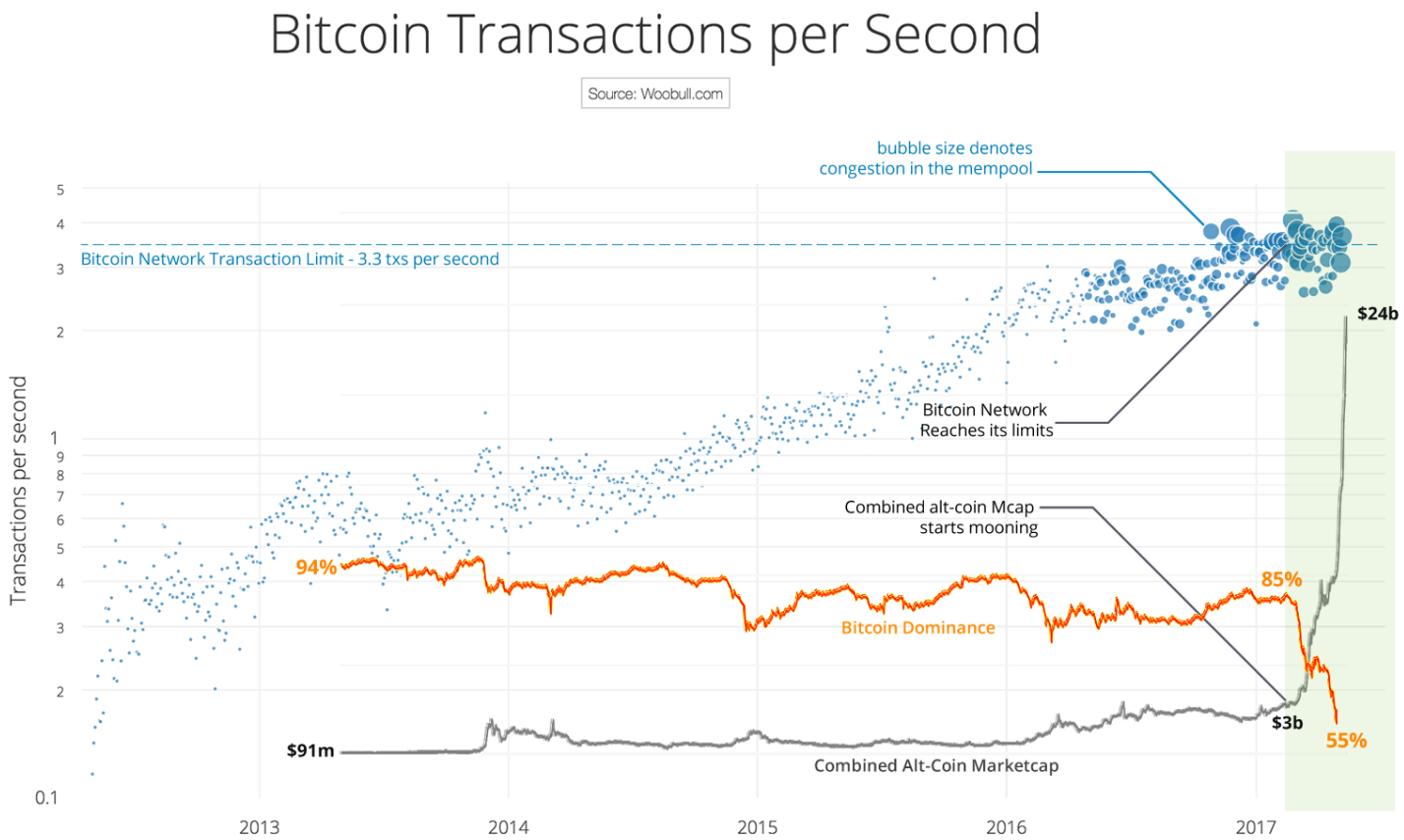
The no central point of trust and predictable money supply together with a clever solution to the double-spending problem is what separates Bitcoin from the 30-year literature on e-cash.

Bitcoin


Block 10
Prev_Hash
Timestamp
Tx_Root
Nonce
Block 11
Prev_Hash
Timestamp
Tx_Root
Nonce
Block 12
Prev_Hash
Timestamp
Tx_Root
Nonce
Hash01
Hash23
Hash0
Hash1
Hash2
Hash3
Tx0
Tx1
Tx2
Tx3

Problems of Bitcoin

Lack of scalability (up to 3.7 tx/s for 1MB per block)? Spam attacks?



Iota

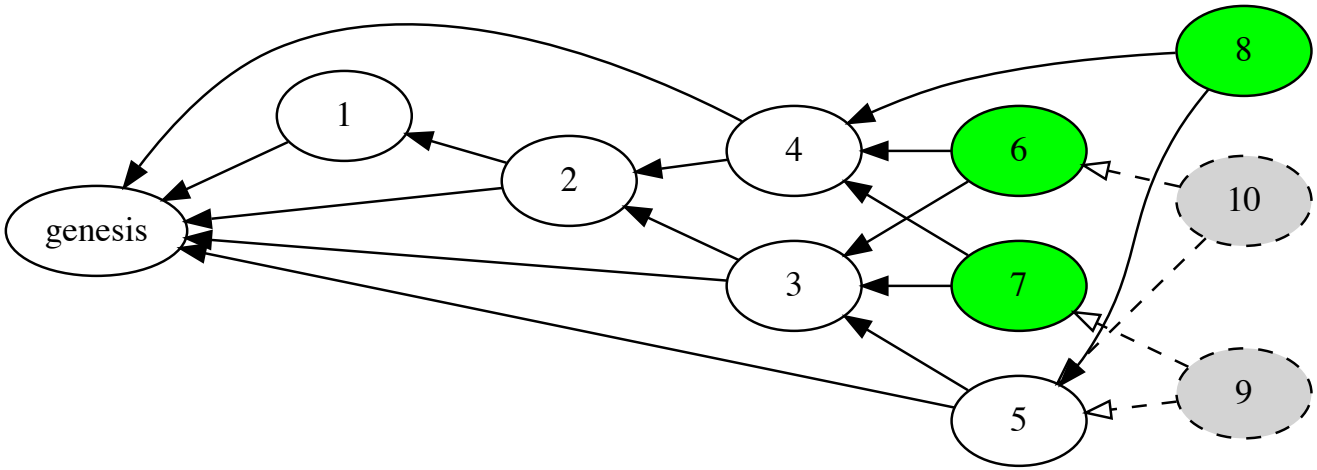


Figure: White nodes represent transactions that have been confirmed at least once. Green circles represent unconfirmed transactions (tips). Gray and dashed nodes are the transactions currently solving the proof-of-work in order to be propagated.

Problems of Iota

Iota does not work in low load scenarios. It uses a central coordinator to prevent attacks.

They does not know what is the minimum number of transactions per seconds so that they can disable the central coordinator.

Contributions

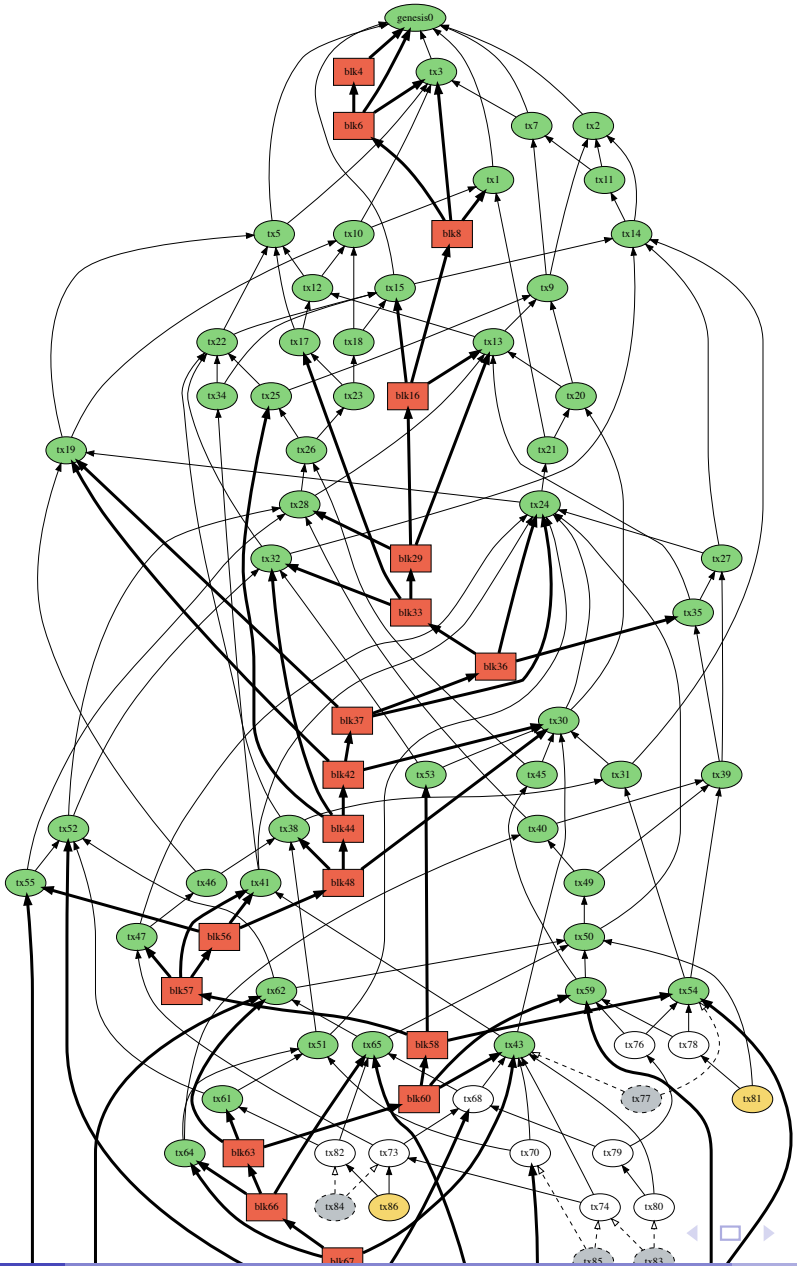
The main contributions are:

i Mathematical analysis of Bitcoin

ii Proposal of Hathor, an architecture which seems to work in both low and high load scenarios.

iii Analysis of Hathor

Hathor's architecture



Mining the next block

Each attempt to find a new block is equivalent to a random sample from a discrete uniform distribution in the interval $[0, 2^{256} - 1]$. A new block is only found when the sample is less than a given number A.

The mining process is a sequence of failed attempts followed by a successful attempt. Let X be the number of attempts until finding a new block. Then, X follows a geometric distribution with probability $p = \frac{A}{2^{256}}$.

Let H be the network hashpower, i.e., the number of attempts per second that the network is able to calculate. Thus, the time to find a new block (or the time between blocks) is $T = X/H$.

As H changes over time, the given number A is adjusted so that the network would find a block every η seconds, which leads to $P(T \leq t) = 1 - \left(1 - \frac{1}{\eta H}\right)^{tH}$.

Mining the next block (2)

Theorem

For H large enough, the time between blocks may be approximated by an exponential distribution, i.e., $P(T \leq t) = 1 - e^{t/\eta}$.

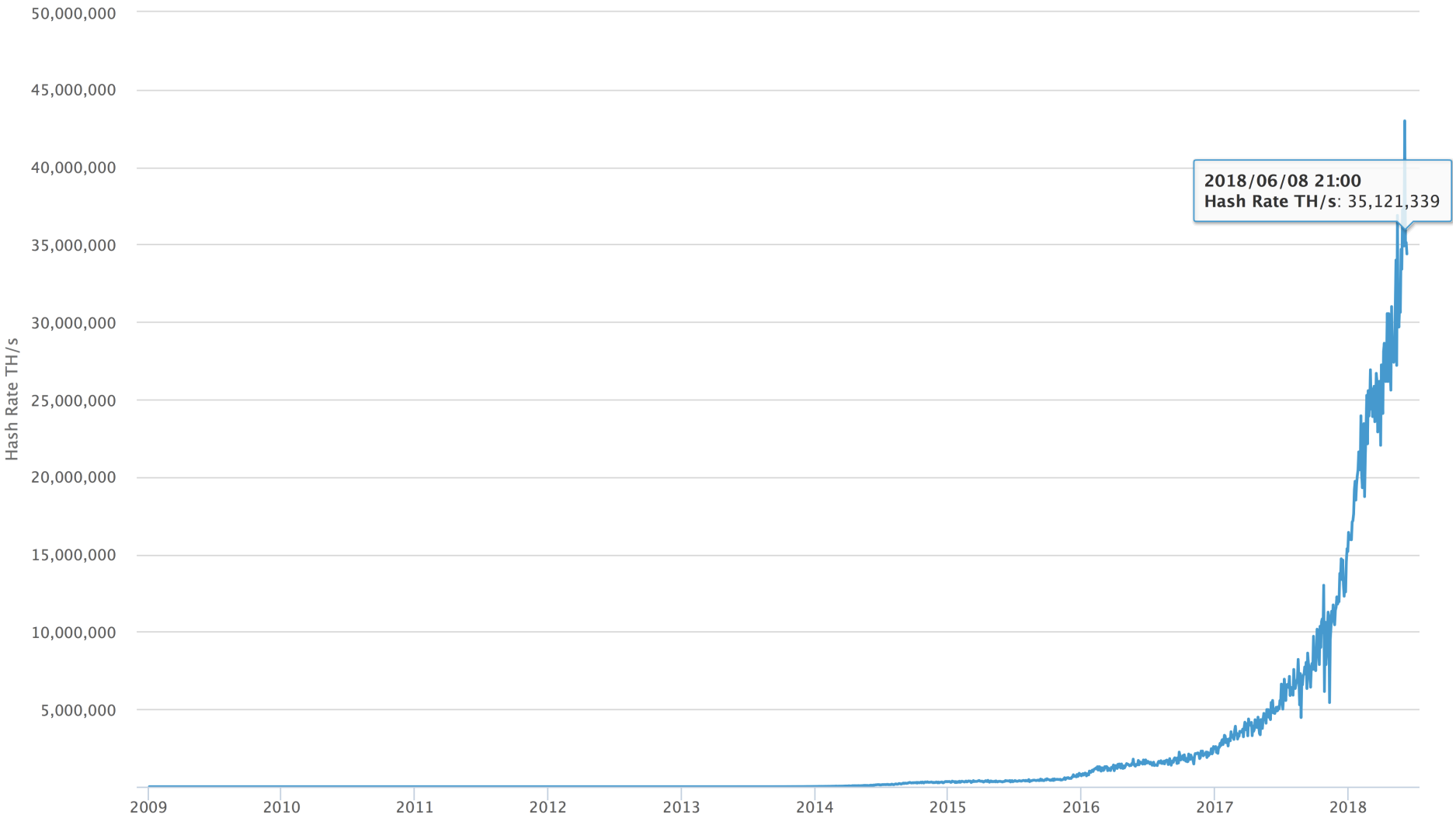
Theorem

The maximum absolute error when approximating the time between blocks by an exponential distribution is e/H.

So, for a maximum absolute error of $0.01\% = 10^{-4}$, the network hashpower H must be at least 26 kH/s.

For comparison, Bitcoin's hashpower is around 35,000,000 TH/s = 35,000,000,000,000,000 kH/s = $35 \cdot 10^{18} H/s$. In this case, the maximum absolute error is $7.7 \cdot 10^{-20} = 0.00000000000000000776$.

Bitcoin's hashpower history



Mining consecutive blocks

Let $Y_n = \sum_{i=1}^{n} T_i$ be the total time to find n consecutive blocks. From probability literature, Y_n follows an Erlang distribution.

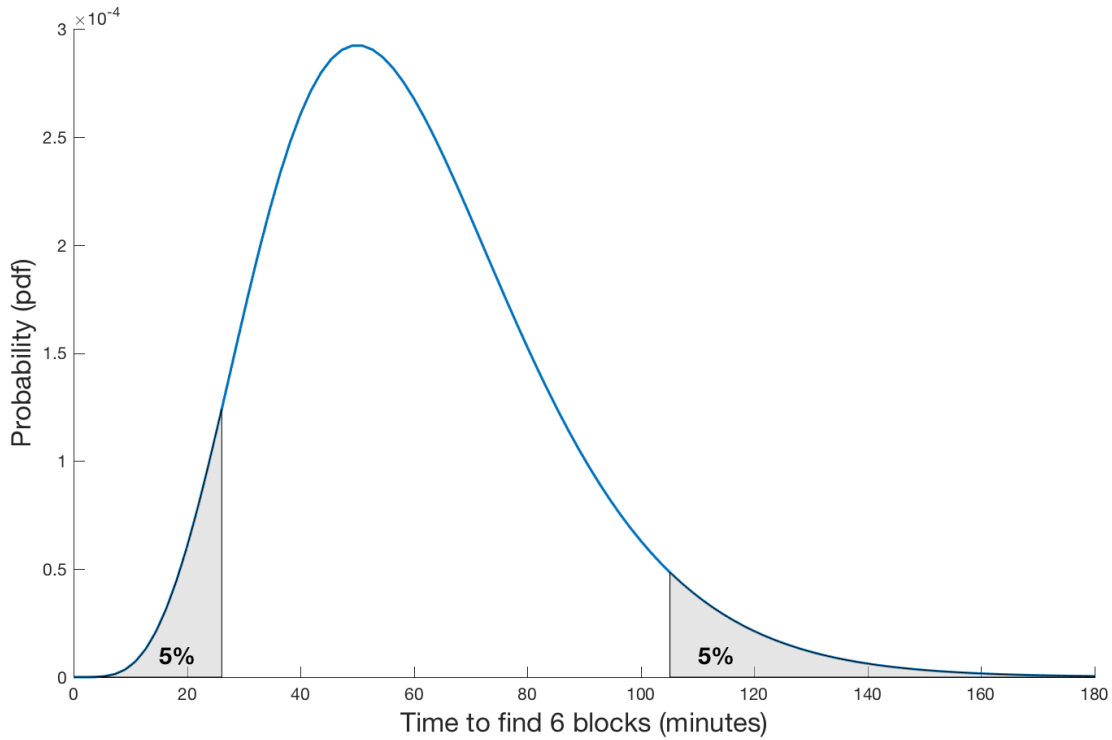


Figure: PDF of Y_6, i.e., probability of finding 6 consecutive blocks.

The double-spending attack

In the double spending attack, the attacker's send some funds to the victim, let's say a merchant.

They wait for k confirmations of the transaction, and the victim delivers the good or the service to the attacker.

Then, the attacker mine enough blocks with a conflicting transaction, double spending the funds which was sent to the victim.

If the attacker is successful, the original transaction will be *erased* and the victim will be left with no funds at all.

Attack in the Bitcoin network

Theorem

Let β be the percentage of the hashpower controlled by the attacker, and γ the percentage of the hashpower without the attacker. Let $p = \frac{\gamma}{\beta+\gamma}$. Then,

$$\mathbf{P}(\textit{successful attack}) = \begin{cases} 1 - \sum_{s=0}^{k-1} \binom{k+s-1}{s} \left((1-p)^s p^k - (1-p)^k p^s\right), & p \geq \\ 1, & \gamma < \beta. \end{cases}$$

For $k = 6$, $p = 0.9$, $\mathbf{P}(\text{successful attack}) = 0.00059141216000002 66$.

For $k = 6$, $p = 0.7$, $\mathbf{P}(\text{successful attack}) = 0.15644958192000014$.

Attack in the Bitcoin network

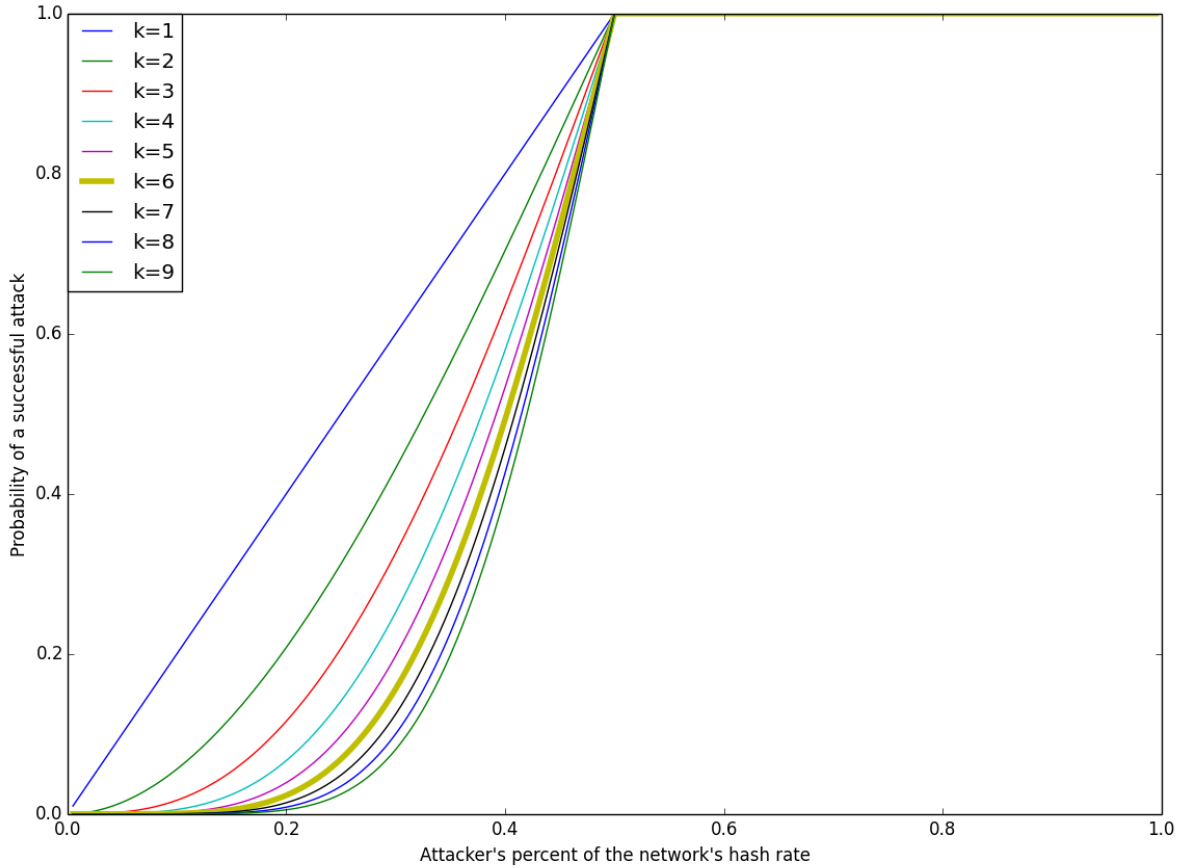
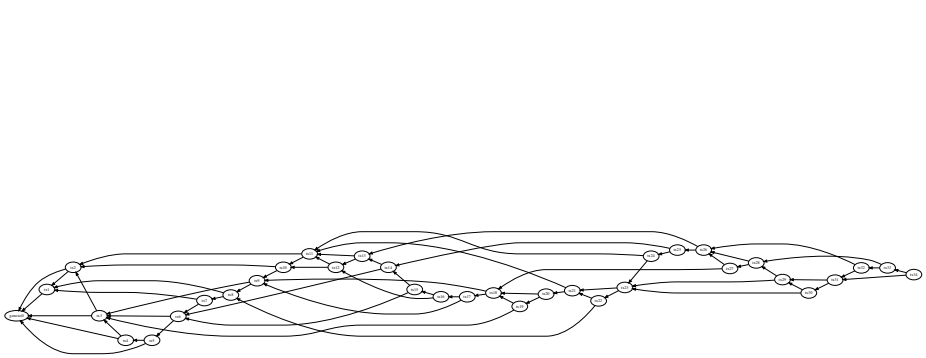
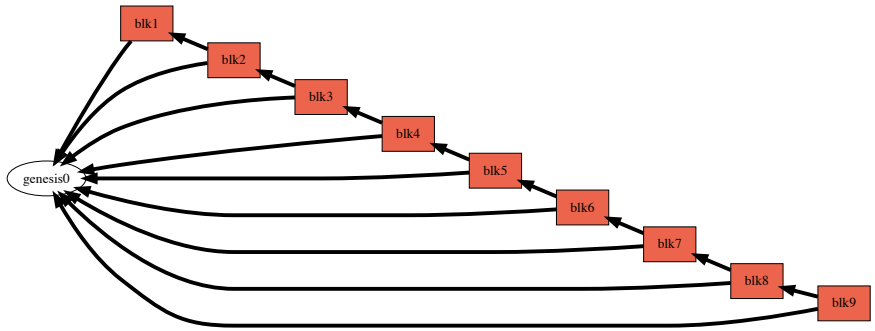


Figure: Probability of a successful attack according to the network's hash rate of the attacker (β).

Analysis of Hathor: boundary cases



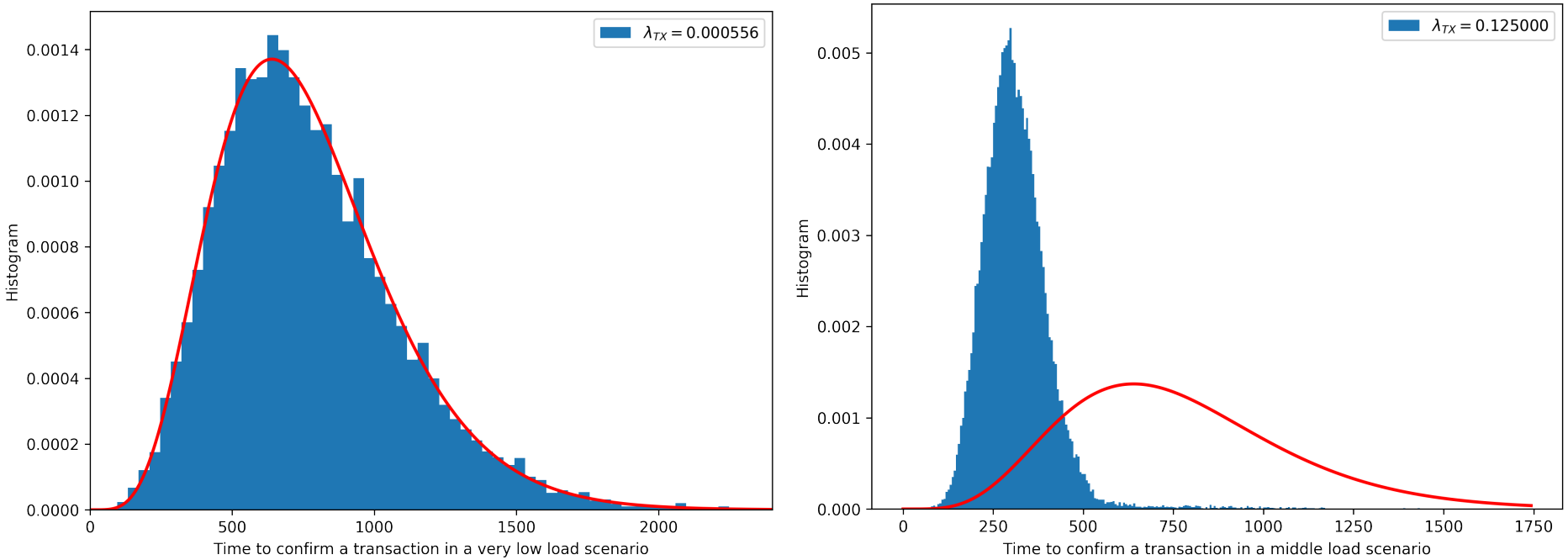
(a) No miners



(b) No transactions

Figure: Visualization of a Hathor's graph in two particular cases: (a) no miners, (b) no transactions.

Analysis of Hathor: confirmation time



(a) Low load

(b) Mid load

Figure: Confirmation time in two scenarios: (a) low load, (b) mid load. The red curve is the distribuion of the time to find six blocks in Bitcoin (which follows an Erlang distribution).

Analysis of Hathor: confirmation time (2)

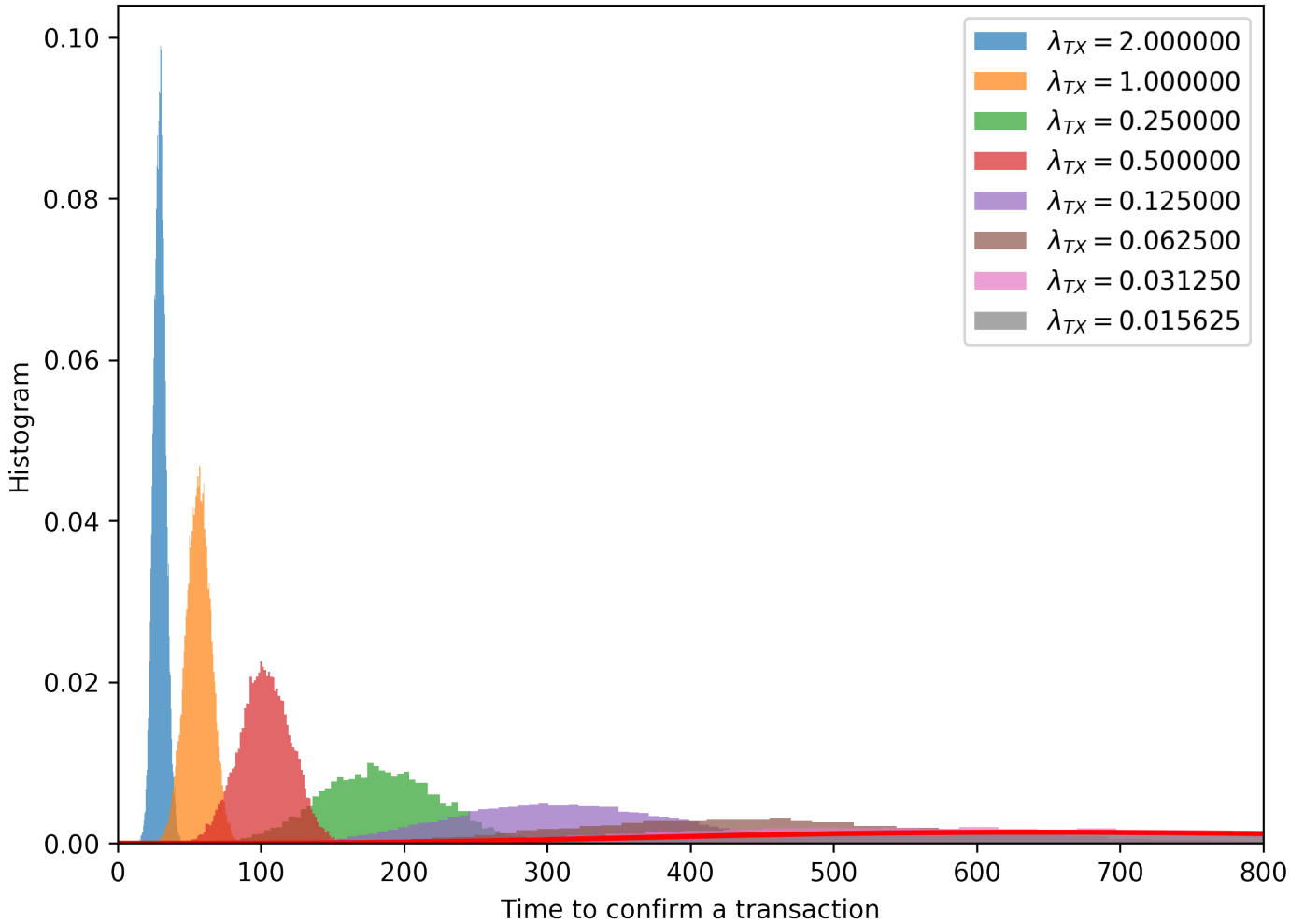


Figure: Confirmation time in many scenarios, moving from a low load ($\lambda_{TX} = 0.015625$) to a high load ($\lambda_{TX} = 2$).

Analysis of Hathor: load changing over time

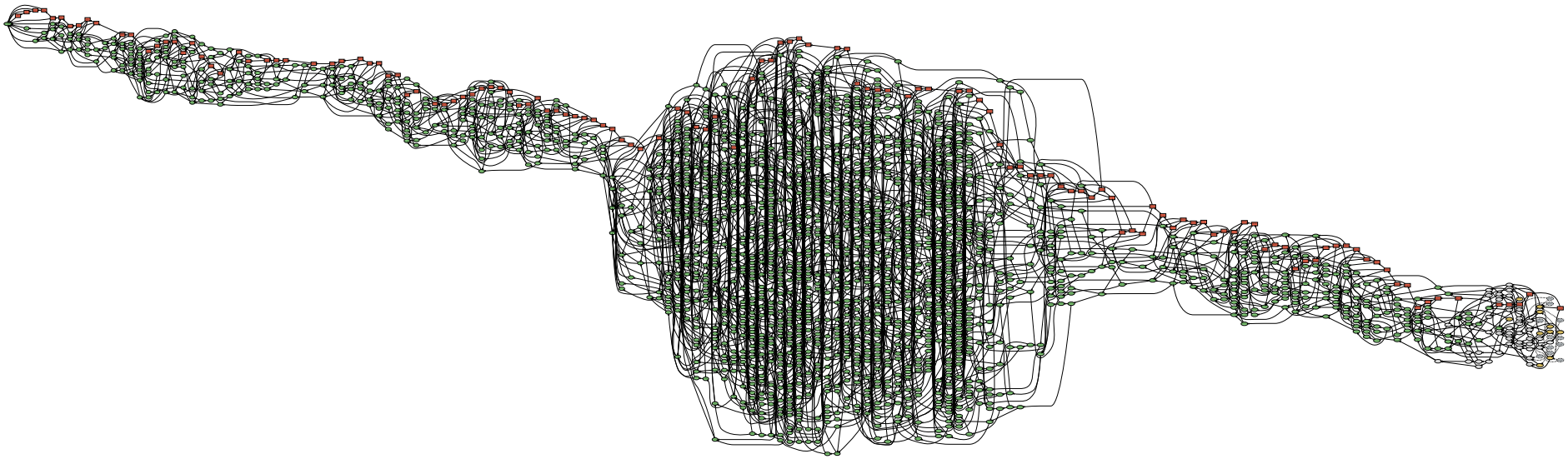


Figure: DAG visualization when the loading is changed over time.

Analysis of Hathor: number of unconfirmed transactions

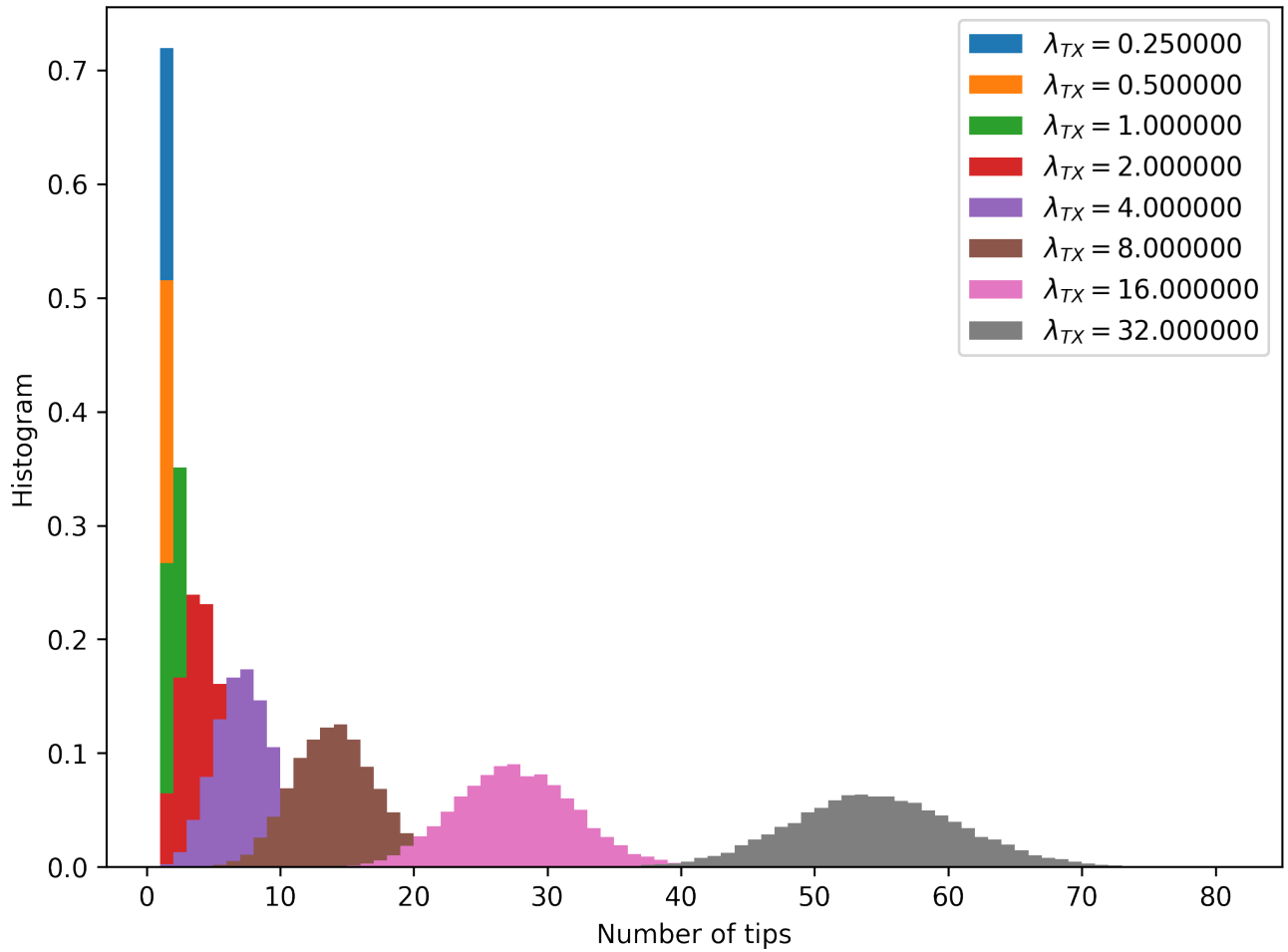


Figure: Histogram of the number of tips for different load scenarios.

An invitation to Sparse Distributed Memory: from the theoretical model to the system dynamics

SDM is a mathematical model of long-term memory that has a number of neuroscientific and psychologically plausible dynamics.

This model is used in all sort of applications due to its ability to closely reflect the human capacity to remember past experiences.

This flexibility into mapping one situation in another is an essential human feature which is hard to replicate in computers.

SDM: How does it work?

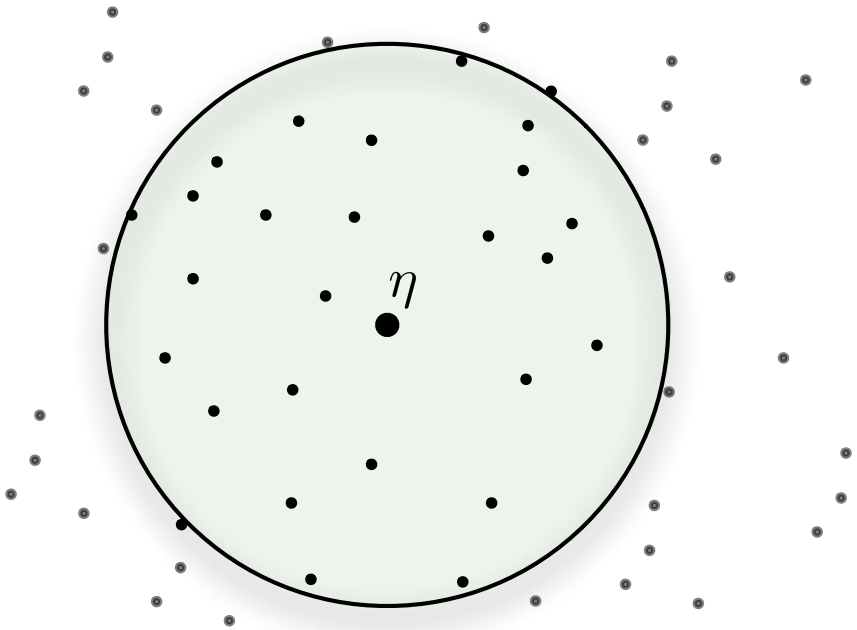


Figure: Activated addresses inside access radius r around the center address.

SDM: How does it work?

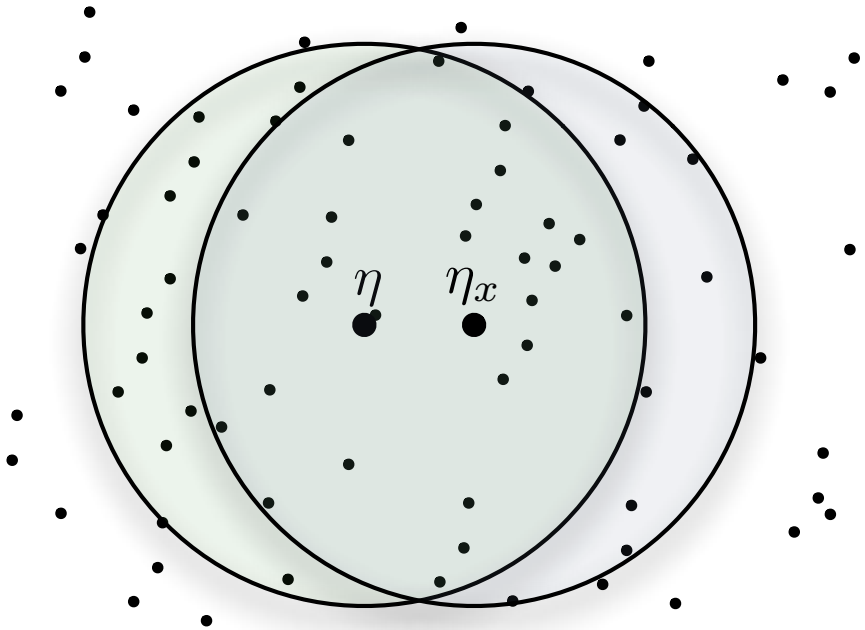


Figure: Shared addresses between the target datum η and the cue η_x.

SDM: How does it work?

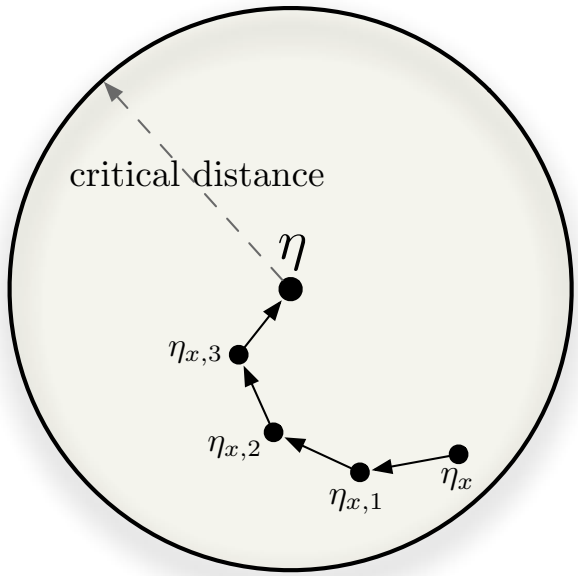


Figure: In this example, four iterative readings were required to converge from η_x to η.

SDM: How does it work?

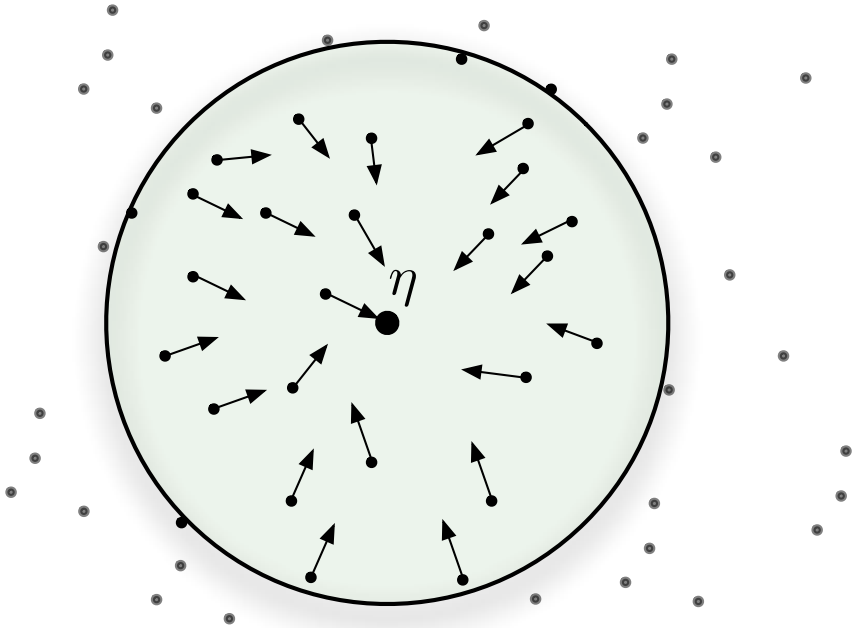


Figure: Hard-locations pointing, approximately, to the target bitstring.

Contributions

The main contributions are:

i. A validated, open-source, cross-platform framework in which researchers are able to replicate someone else's results
ii. Identification and modeling of the autocorrelation in the counters
iii. Loss of neurons resilience
iv. Noise filtering application
v. Classification application
vi. Reinforcement learning application

Loss of neurons resilience

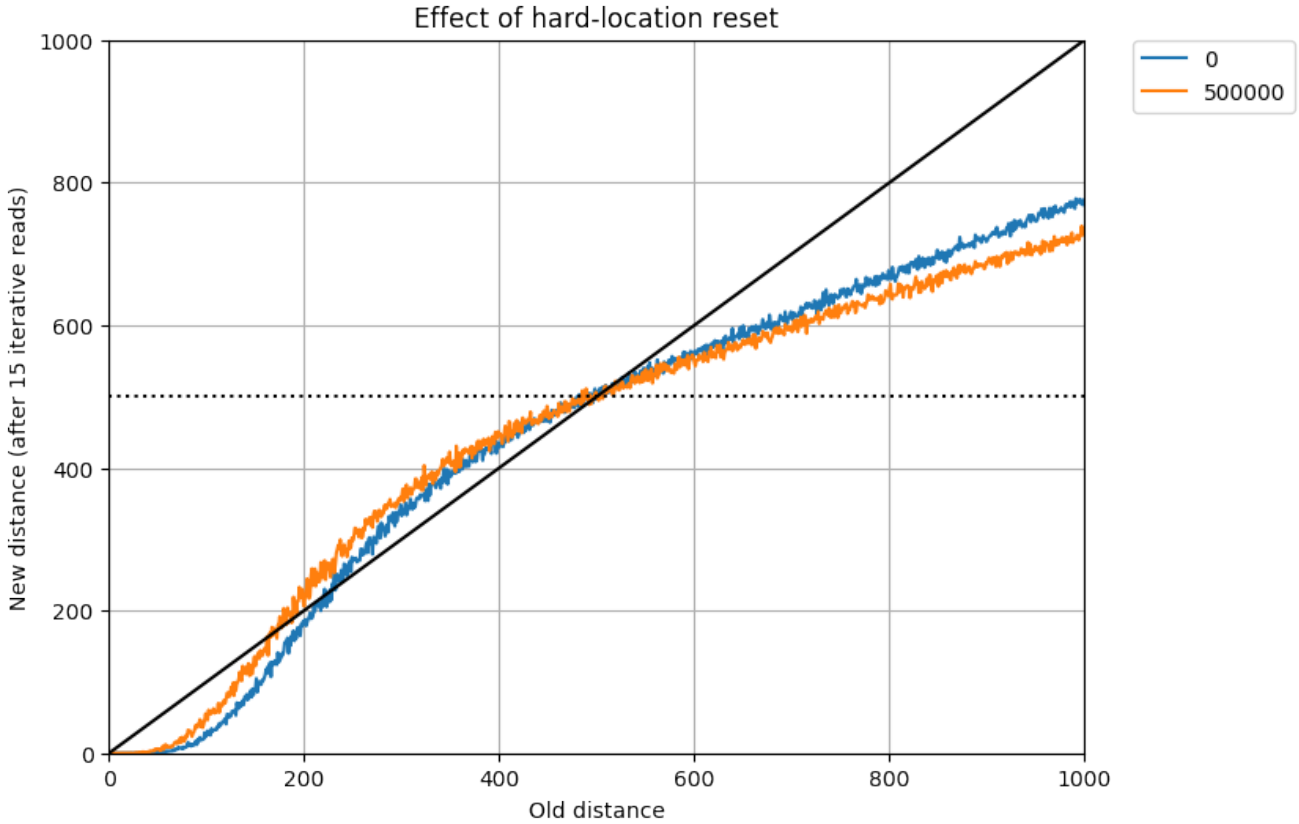


Figure: This graph shows the SDM's robustness against loss of neurons in a SDM with $n = 1,000$ and $H = 1,000,000$. Even when 50% of neurons are dead, SDM recall is barely affected, which is an impressive result and matches with some clinical results of children submitted to hemispherectomy.

Loss of neurons resilience

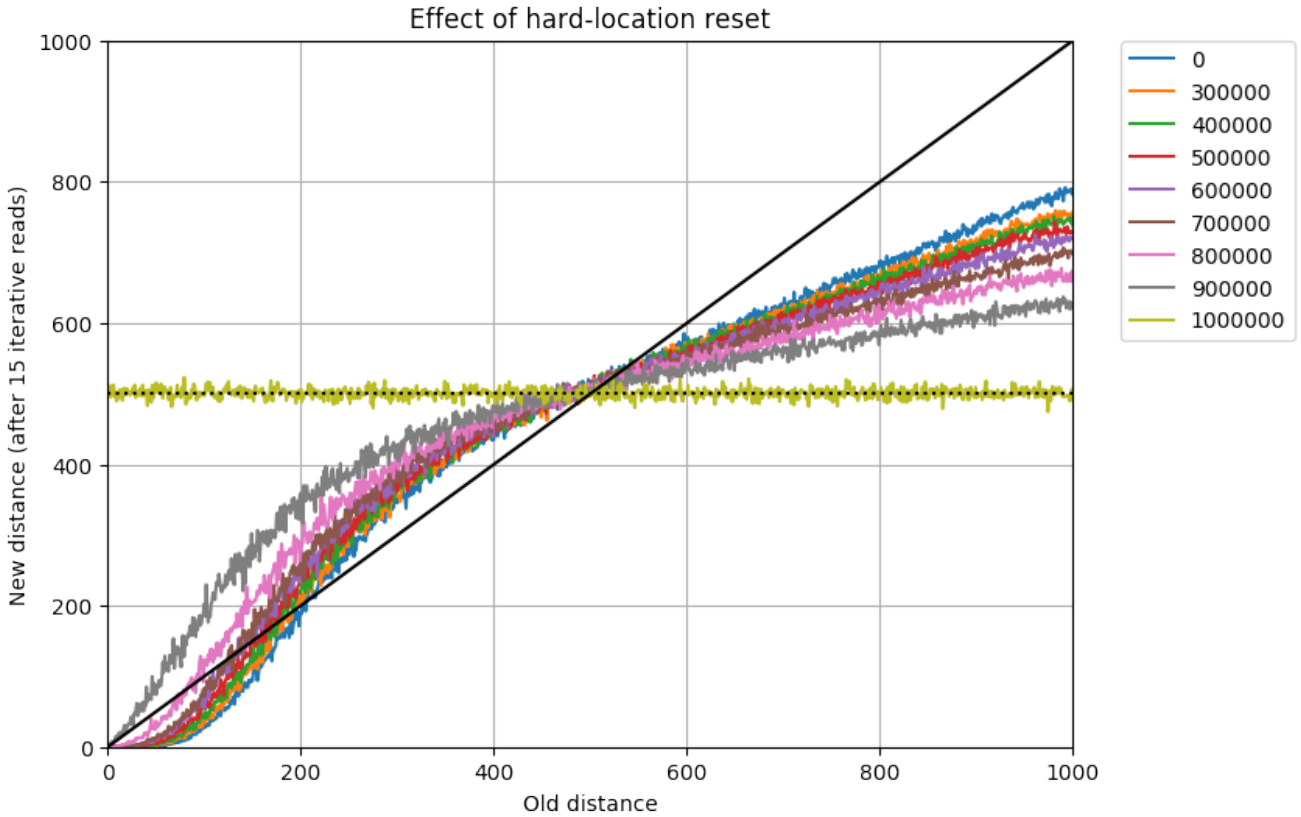


Figure: This graph shows the SDM's robustness against loss of neurons in a SDM with $n = 1,000$ and $H = 1,000,000$. The more neurons are lost, the smaller the critical distance, i.e., the worse the SDM recall.

Critical distance & Autocorrelation

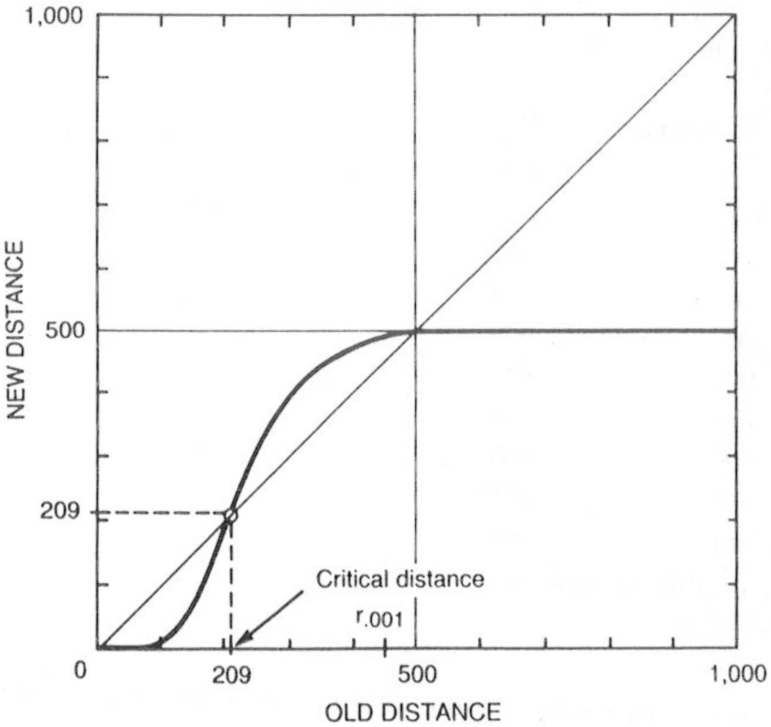
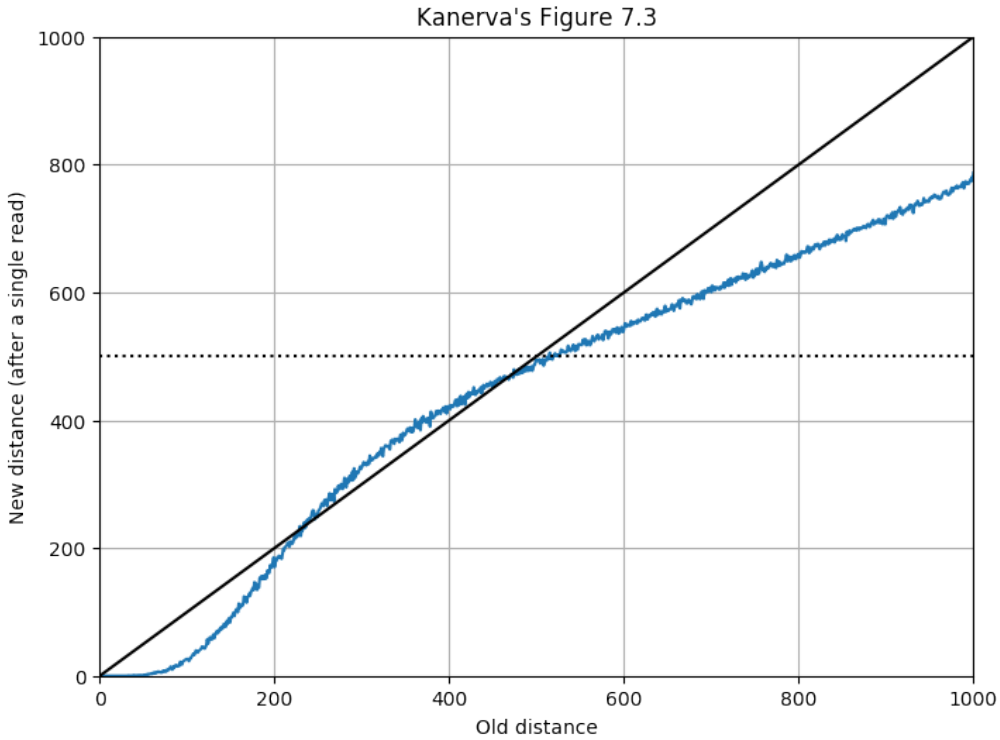


Figure 7.3
New distance to target as a function of old distance.



Critical distance & Autocorrelation

Theorem

Each dimension has a small pull bias, which can be measured by

$$P(x_i = y_i | d(x,y) \leq r) = \frac{\sum_{k=0}^{r} \binom{n-1}{k}}{\sum_{k=0}^{r} \binom{n}{k}}$$

Theorem

The autocorrelation vanishes when $n \to \infty$, i.e.,
$\lim_{n\to\infty} P(x_i = y_i | d(x,y) \leq r) = 1/2$.

When $n = 1,000$ and $r = 451$, $P(x_i = y_i | d(x,y) \leq r) = 0.552905498137$.

Critical distance & Autocorrelation

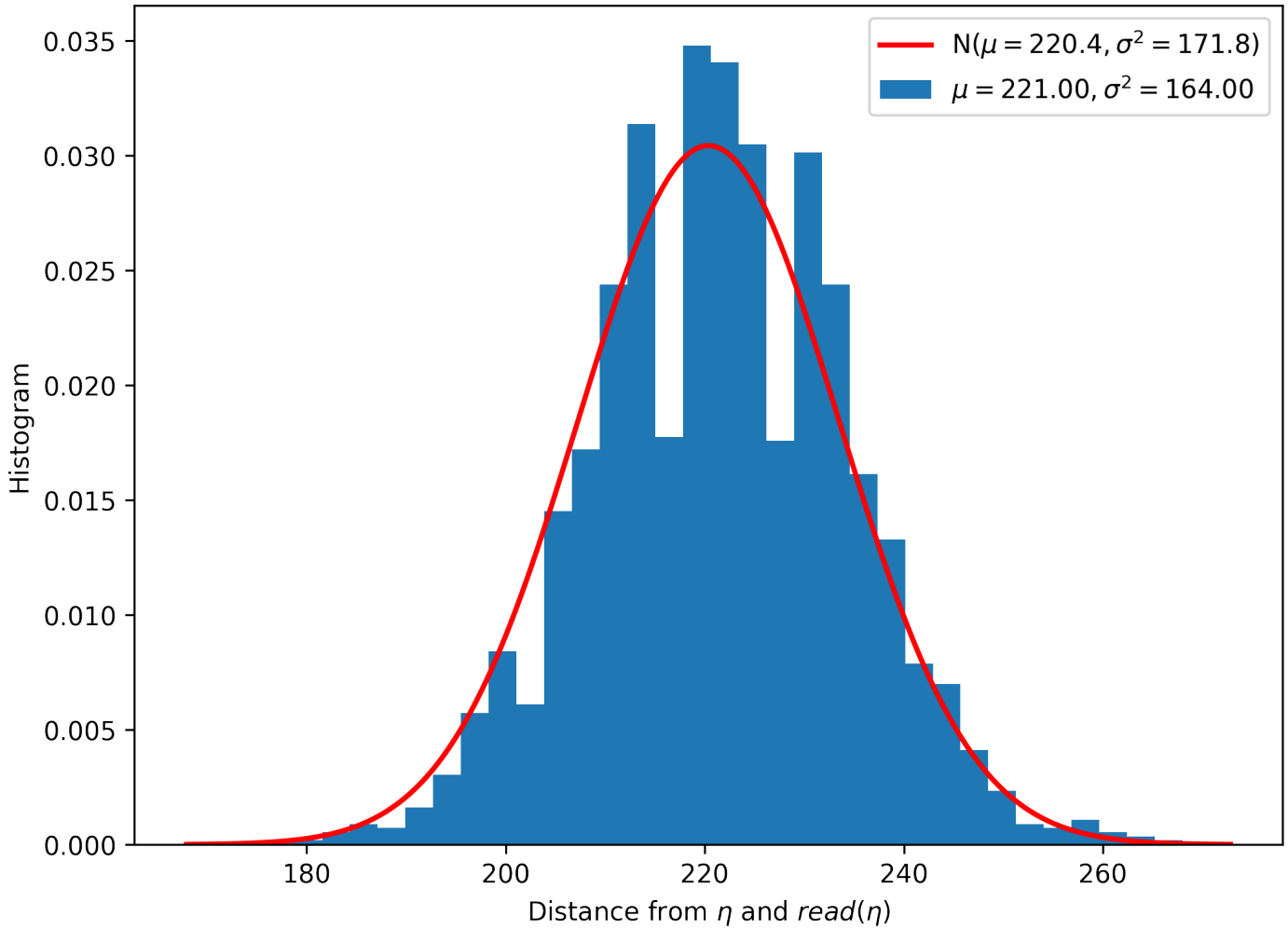


Figure: The histogram was obtained through simulation. The red curve is the theoretical normal distribution.

Critical distance & Autocorrelation

The analytical model which does not consider autocorrelation predicts a critical distance of 209 bits.

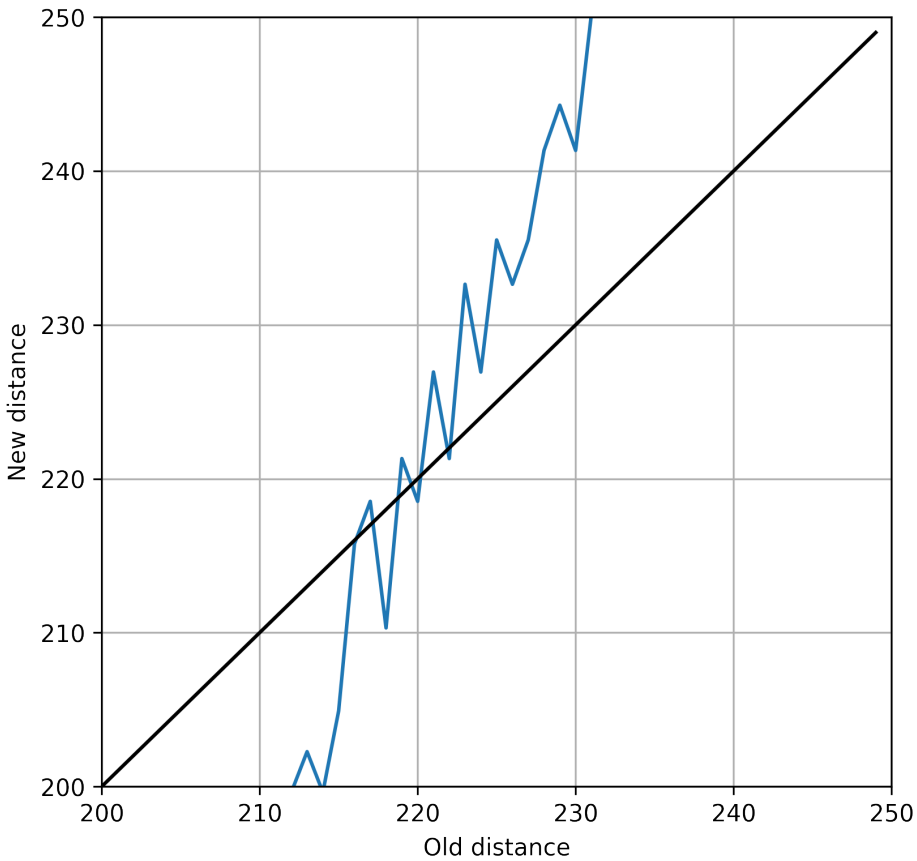


Figure: Zoom-in around $d = 209$.

Classification application: training

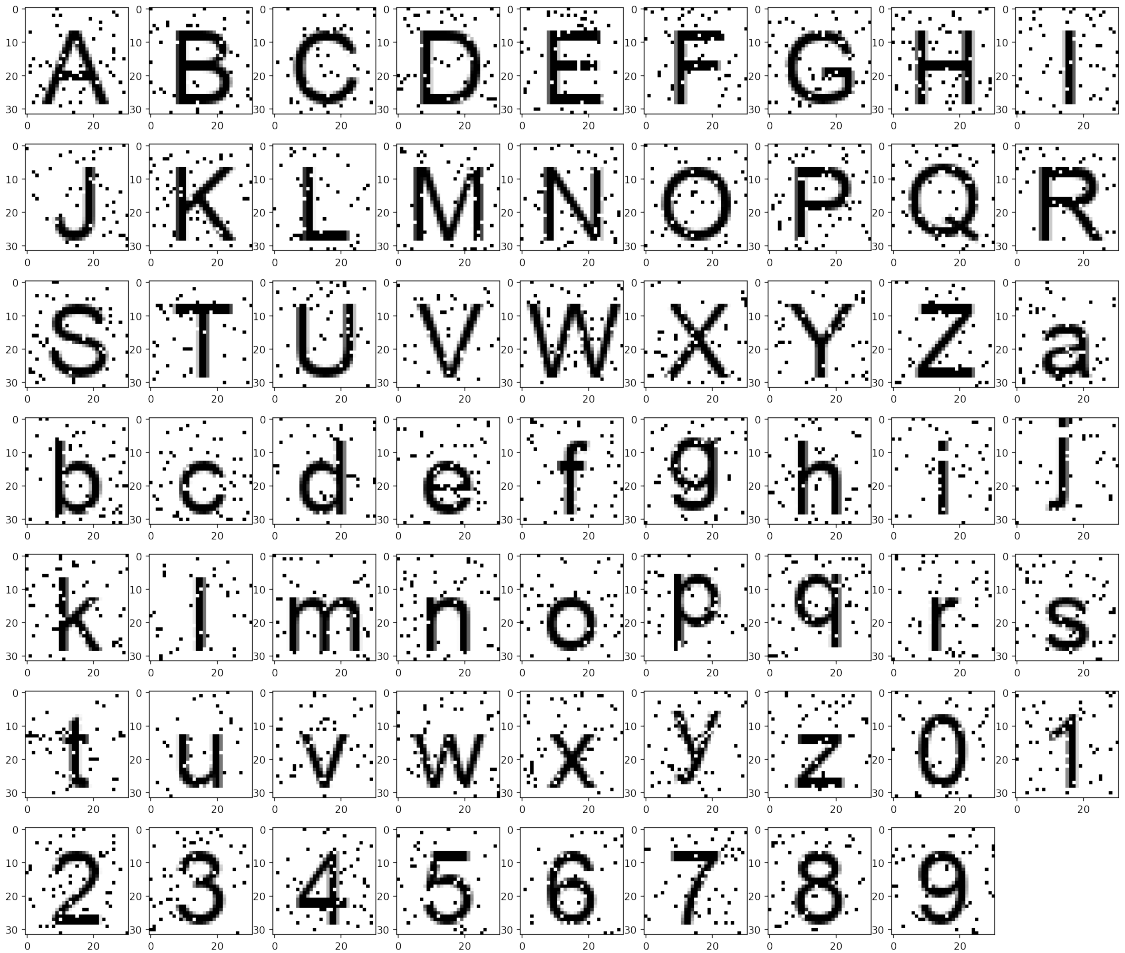
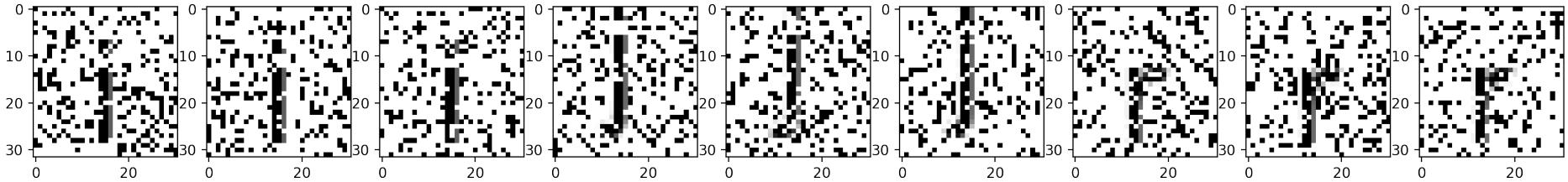
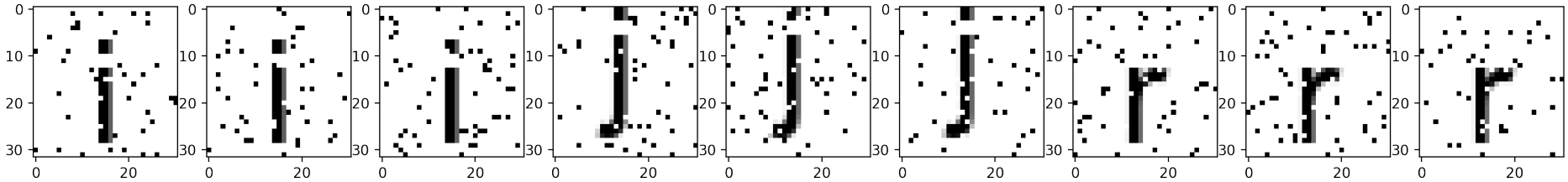


Figure: One noisy image for each of the 62 classification groups.

Classification application: confusing cases

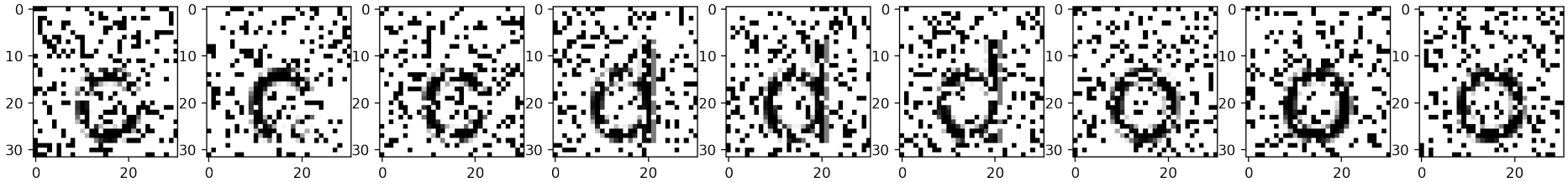


(a) “i”, “l”, and “r” with 20% noise.

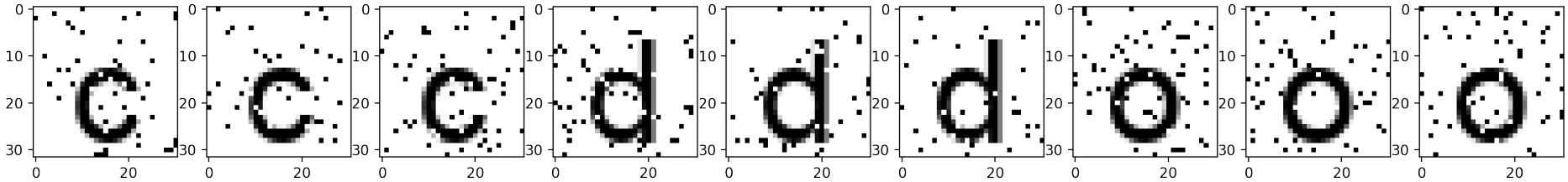


(b) “i”, “l”, and “r” with 5% noise.

Classification application: confusing cases

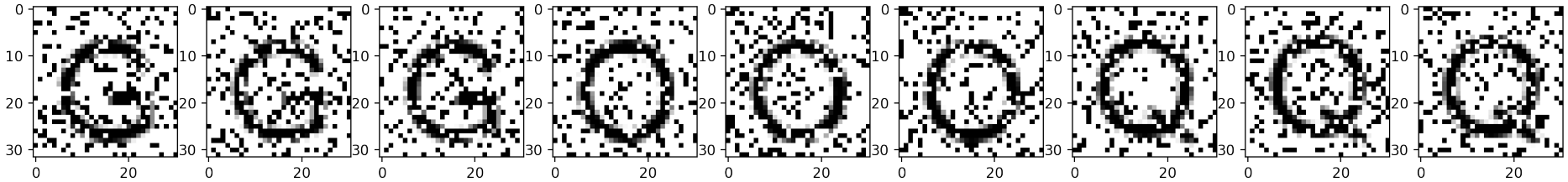


(c) “c”, “d”, and “o” with 20% noise.

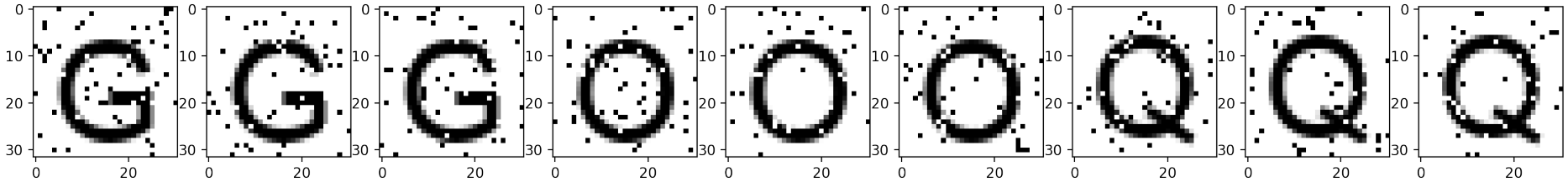


(d) “c”, “d”, and “o” with 5% noise.

Classification application: confusing cases



(e) "G", "O", and "Q" with 20% noise.



(f) "G", "O", and "Q" with 5% noise.

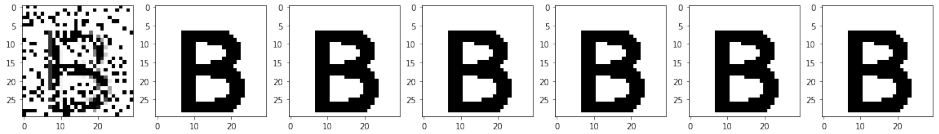
Classification application: results

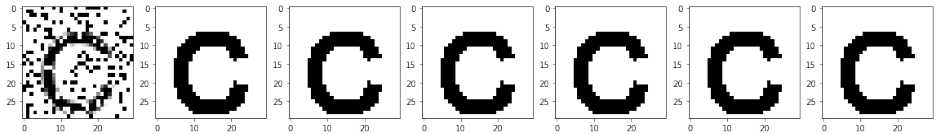
Generally speaking, the classification algorithm made some mistakes in the 20% noise scenario, and just a few mistakes in the 5% noise scenario.

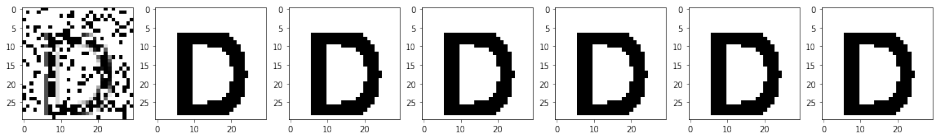
When the training set has 1,000 samples instead of 100 samples of each letter, the algorithm correctly classify almost all letters in both 20% and 5% noise scenarios, except for letters "e" and "o".

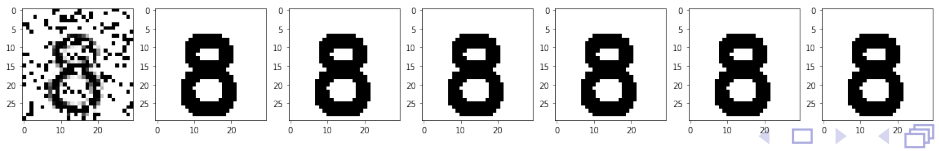
It could not differentiate letters "e" and "o" because their differences are smaller than the critical distance. It would have to be trained more and more times in order to learn to differentiate them (because the critical distance decreases when more items are written into the memory).

Noise filtering: results

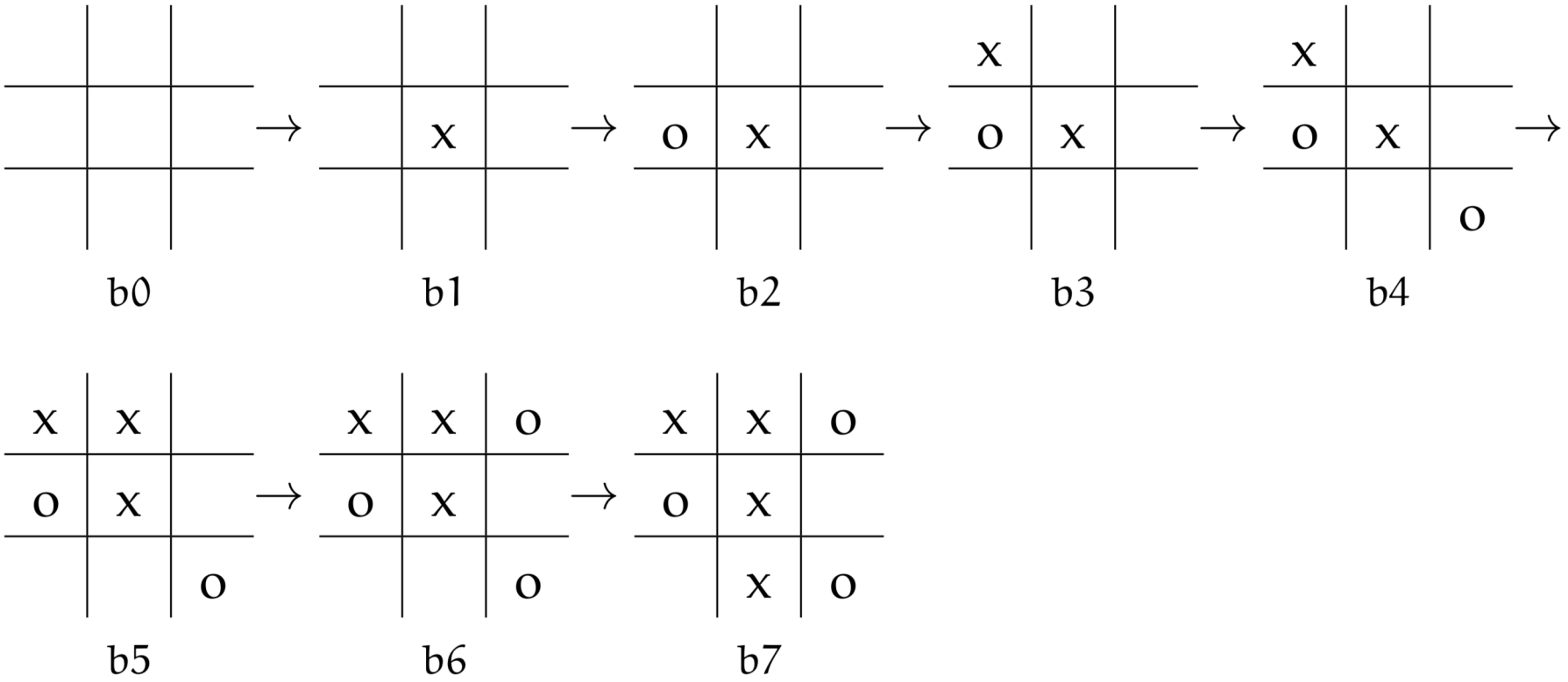








Reinforcement learning: TicTacToe



Reinforcement learning: Results

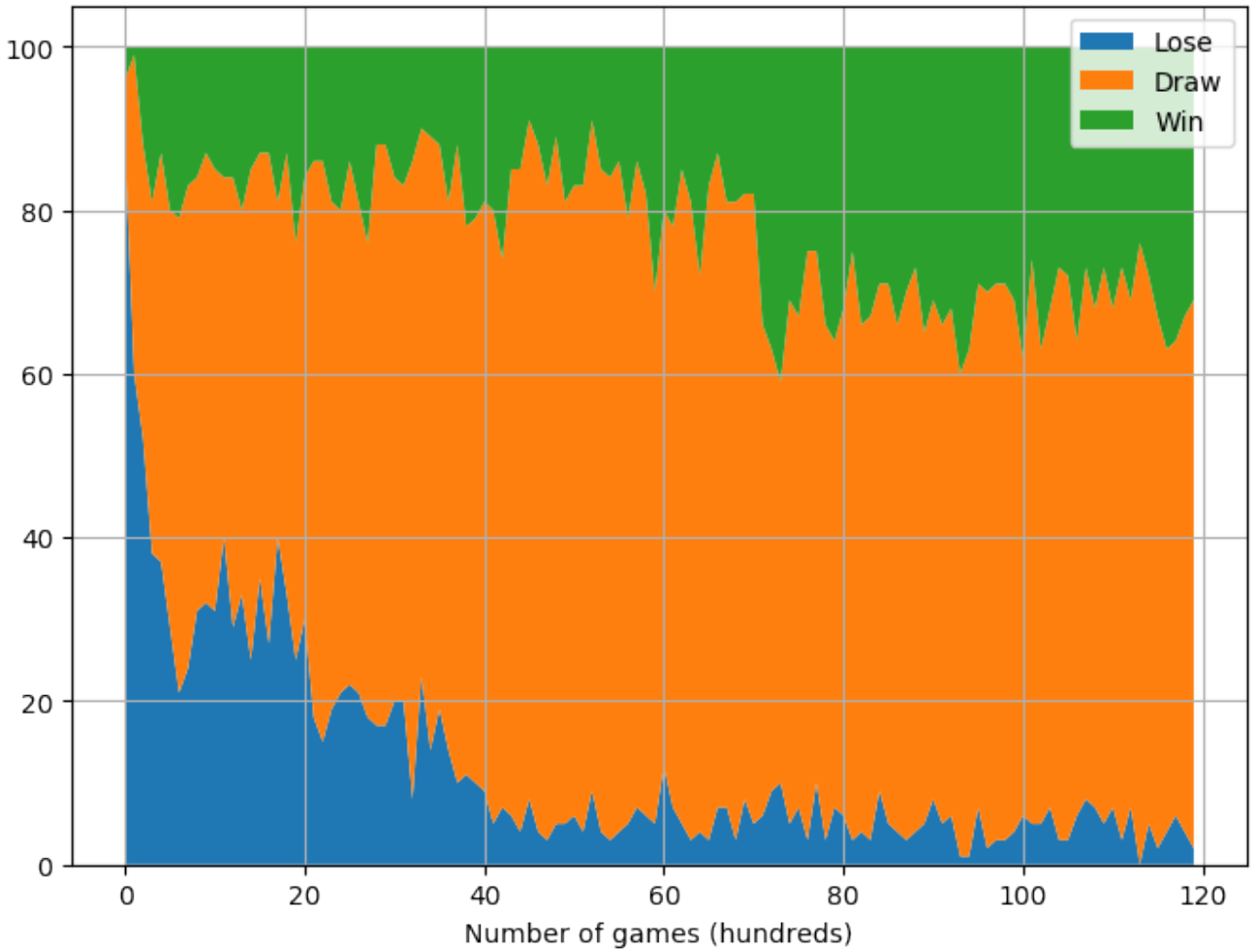


Figure: Results playing against the smart player. Each cycle was made of 100 games for training, and then 100 games for measuring statistics.

Diffusion and dismissal of innovation: forecasting the number of Facebook's active users

The main contributions are:

i Add a rejection parameter to the model

ii Estimate the parameters using only the number of active users

iii The possibility of reducing the number of active users (instead of only increase)