COEN 6331
Neural Networks

Assignment 4

Evaluating ART Network Performance in Pattern Recognition

Benjamin Vauchel
Student ID: 40280381

April 3rd, 2024

Table of Contents

Table of Figures

Table of Tables

Introduction

This report introduces Adaptive Resonance Theory (ART) networks, a family of neural networks designed to tackle the challenges of pattern recognition. ART networks are renowned for their ability to solve the ‘plasticity/stability’ problem, that is, adapt to new information without forgetting previously learned data, and categorize input patterns based on their similarities and differences.

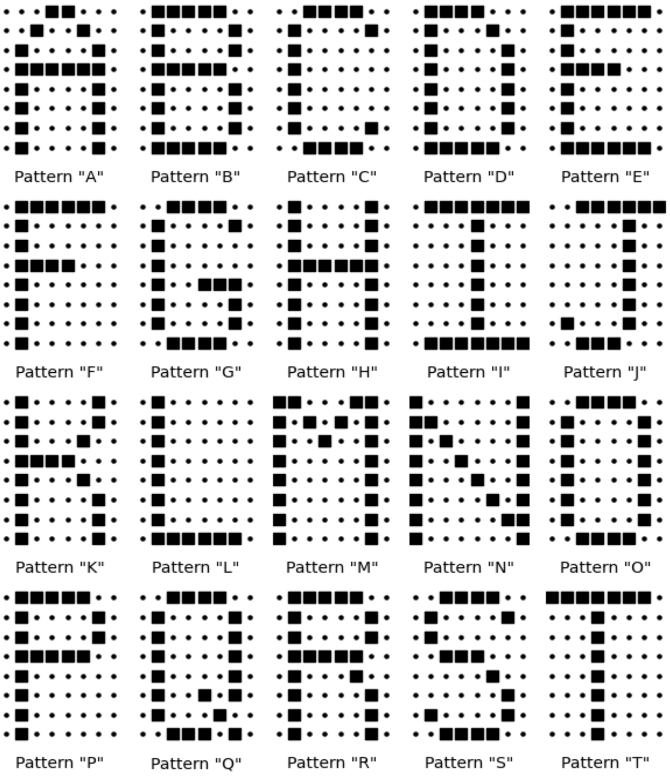
The problem addressed in this report is the identification and classification of binary patterns representing letters of the alphabets A to T. We explore the performance of an ART network in recognizing these patterns, which are represented as binary matrices. This involves setting up a pattern set, analyzing the network's efficiency through various metrics, and evaluating its ability to handle both clean and noisy data.

In particular, we examine the initialization of weights, the critical learning process, and the significance of parameters such as the vigilance criterion in the categorization process. Additionally, we introduce modifications to improve the network's accuracy and adaptability, highlighting the impact of a penalty rate on the similarity calculation between input patterns and learned categories.

Analysis

Setting the Pattern Set

The pattern set is made up of twenty patterns representing the alphabets A to T of size 8x8, as shown in **Figure 1**. Each pattern is a binary matrix of size (8, 8), representing a letter form, and therefore comprising a total of 64 pixels. The value of a pixel is either 0 (represented by a small circle or a white pixel) or 1 (represented by a solid square or a black pixel).



Pattern "A" Pattern "B" Pattern "C" Pattern "D" Pattern "E"

Pattern "F" Pattern "G" Pattern "H" Pattern "I" Pattern "J"

Pattern "K" Pattern "L" Pattern "M" Pattern "N" Pattern "O"

Pattern "P" Pattern "Q" Pattern "R" Pattern "S" Pattern "T"

Figure 1. Pattern set plotted as pixels

The patterns are flattened into a 1-D array (of 64 pixels) before being put into the ART Network, to match the network architecture.

Evaluation Metrics

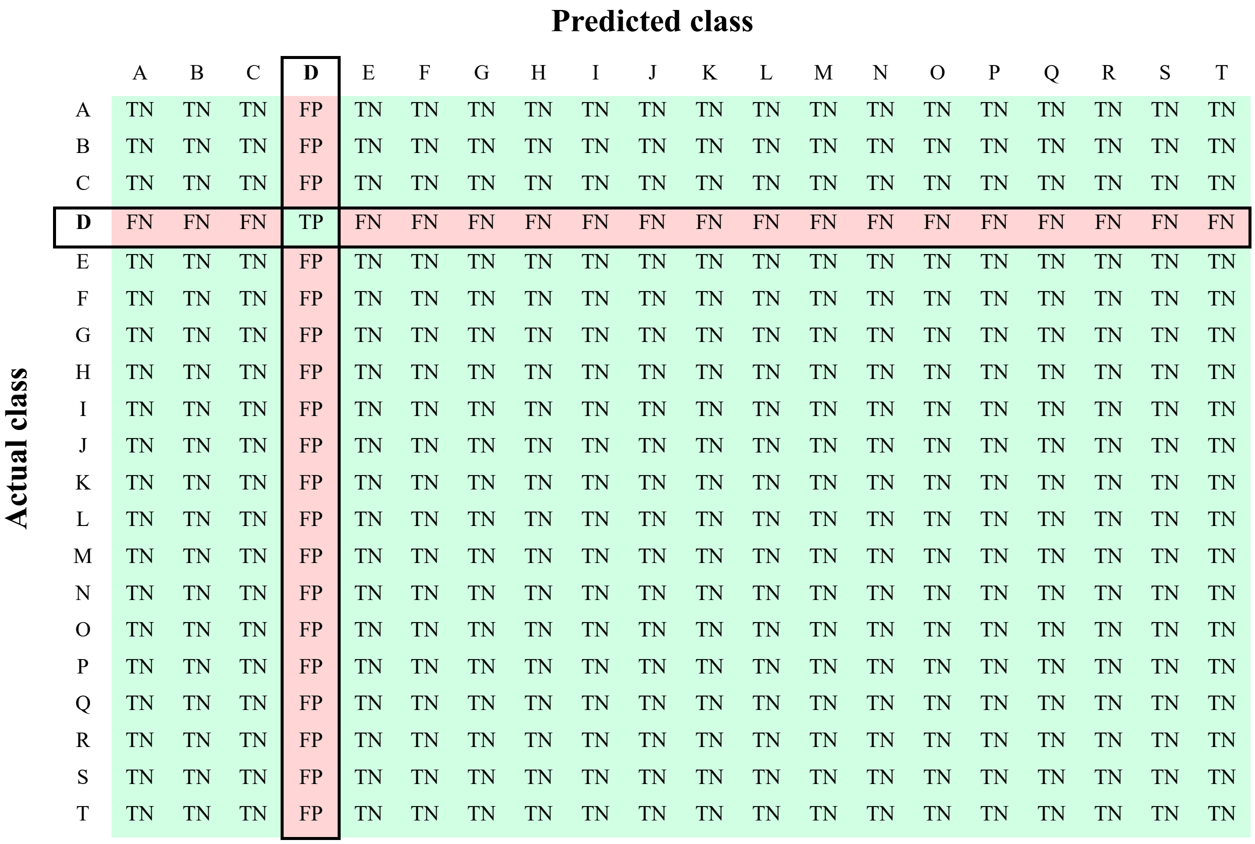
The most important evaluation metric to get an understanding of the overall performance of the network is the overall accuracy, which is defined as:

$$\text{Overall Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{\mathrm{tr}(\text{Confusion Matrix})}{\mathrm{sum}(\text{Confusion Matrix})}$$

To visualize more accurately the performance of the model, we can calculate the Confusion Matrix which shows the true class labels against the predicted class labels, allowing us to evaluate how well the model performs for each individual class.

From the confusion matrix, we can calculate four interesting numbers:

- **True Positives (TP)** for a class are instances correctly predicted as belonging to that class.
- **False Positives (FP)** are instances wrongly predicted as belonging to that class (but actually belong to a different class).
- **False Negatives (FN)** are instances of that class incorrectly predicted as belonging to a different class.
- **True Negatives (TN)** for a class are all the instances that are correctly identified as not belonging to that class.



Predicted class

Actual class

	A	B	C	**D**	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
A	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
B	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
C	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
D	FN	FN	FN	TP	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN	FN
E	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
F	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
G	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
H	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
I	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
J	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
K	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
L	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
M	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
N	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
O	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
P	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
Q	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
R	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
S	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN
T	TN	TN	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN	TN

Figure 2. Confusion Matrix example highlighting the TP, TN, FP and FN for class “D”

The goal is to minimize false predictions, that is, FN and FP, and maximize true predictions, TP and TN.

When TP, TN, FP and FN are determined, the accuracy for one specific class can be calculated:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

However, this metric ignores the specific types of errors the model makes. To evaluate how well the model deals with identifying and predicting True Positives, we should measure precision and recall instead.

Precision measures how often the positive predictions are correct:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Recall measures how often the model correctly identifies positive instances (true positives) from all the actual positive samples in the dataset:

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Finally, the F_1 score is the harmonic mean of precision and recall:

$$\mathrm{F}_1 = 2\frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

It allows us to represent both precision and recall in one metric.

Building the ART Network

The ART1 network comprises two main layers in addition to the input layer:

- **Comparison Layer $\boldsymbol{F_1}$:** It compares the input pattern with the prototype patterns the network has learned so far.
- **Recognition Layer $\boldsymbol{F_2}$:** It consists of neurons that represent categories or classes of the input patterns.

The global structure of an ART network is shown in **Figure 3**.

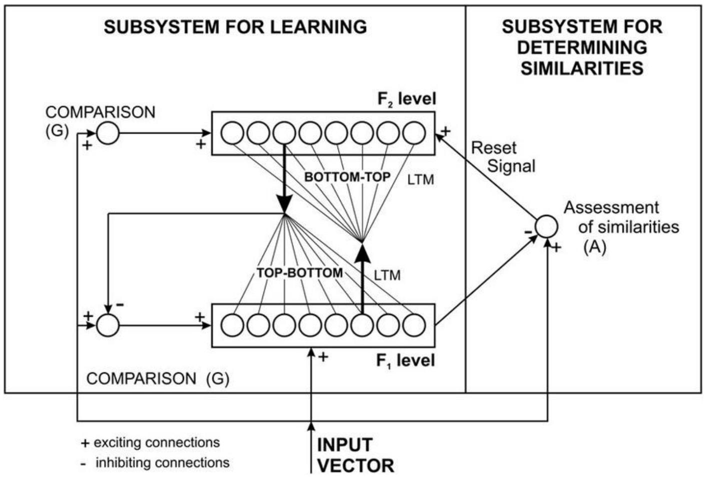


Figure 3. Global structure of an ART network[1]

Weight Matrices

There are two sets of weights: bottom-up weights and top-down weights, each serving a different function in the network's operation.

The bottom-up weights W_{bu} connect the input layer to the recognition layer. These weights are used in the initial categorization layer, proposing possible categories based on the current input. The initial values of bottom-up weights are set to

$$\frac{1}{1 + n_{\text{features}}}$$

where n_{features} is the number of features in the input pattern. This initialization ensures that no single feature dominates the initial learning phase.

The update rule for bottom-up weights after a category is selected (and assuming the vigilance criterion is met, see below) is as follows:

$$W_{\text{bu}}^{\text{new}}(c, i) = \frac{W_{\text{td}}^{\text{new}}(c, i)}{0.5 + \sum_j W_{\text{td}}^{\text{new}}(c, j)}$$

[1] *Source:* Automatic feature recognition using artificial neural networks to integrate design and manufacturing: Review of automatic feature recognition systems - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-structure-of-the-adaptive-resonance-theory-using-the-binary-vector-input-form-ART-1_fig13_220306695 [accessed 3 Apr, 2024]

where $W_{\text{td}}^{\text{new}}(c, j)$ represents the updated top-down weight for category c and feature i, and the denominator normalizes the weights, ensuring they remain within suitable bounds.

The top-down weights W_{td} connect the recognition layer back to the input layer. They are used in the recognition phase, determining how closely the current input pattern matches the stored pattern of each category. Thus, these weights represent the model's memory of each category. Initially, top-down weights are set to 1, indicating a neutral state where any input can potentially match any category.

When a category is selected (and ensuring it meets the vigilance criterion), top-down weights are updated according to the learning rule:

$$W_{\text{td}}^{\text{new}}(c, i) = \lambda \times \min\left(X_i, W_{\text{td}}^{\text{old}}(c, i)\right) + (1 - \lambda) W_{\text{td}}^{\text{old}}(c, i)$$

where λ is the learning rate, X_i is the i-th element of the input pattern, and $W_{\text{td}}^{\text{old}}(c, i)$ is the old weight for category c and feature i. This rule adjusts the category's prototype to better match the input pattern, incorporating the new information while retaining previously learned characteristics based on the learning rate, solving the ‘plasticity/stability’ problem.

Learning Process

The learning process in an ART1 network involves the following steps:

1. **Initialization:** The weights are initialized as explained above, and parameters such as the vigilance parameter $\delta \in]0,1[$ are set.
2. **Input Presentation:** A binary input pattern I is presented to the F_1 layer of the network.
3. **Competition:** The recognition layer's neurons compete to represent the input pattern, and the neuron that has the highest similarity wins. This process is performed with a bottom-up weight vector (input to recognition layer) and a top-down weight vector (recognition layer to input). Matching scores are computed as

$$\forall m \in [1, M], \qquad y_m^0 = \sum_{i=1}^{n} w_{\mathrm{bu},im} x_i = \sum_{i=1}^{n} \min(w_{\mathrm{bu},im}, x_i)$$

The last equality is due to the nature of binary representation: a bitwise product is similar to the minimum function when dealing with 0 and 1.

The best matching category j is then selected with

$$y_j^0 = \max_m (y_m^0)$$

This score does not involve the same conservative criteria as the vigilance test but instead serves to guide the initial category choice before the similarity check.

4. **Resonance Check:** The network checks if the similarity between the input pattern and the winning neuron's prototype meets a predefined threshold determined by the vigilance parameter. If the similarity is sufficient, the network updates the weights to incorporate the new pattern into the winning category. The similarity is computed as

   $$\sigma = \frac{1}{\|x\|} \sum_{i=1}^{n} w_{td,ij} x_i = \frac{1}{\|x\|} \sum_{i=1}^{n} \min(w_{td,ij}, x_i)$$

   This similarity is compared to the vigilance δ: if $\sigma > \delta$ then the weights are updated as explained above, otherwise the node j is deactivated by the reset mechanism mentioned below. The vigilance parameter is crucial in the training phase: A higher vigilance promotes stability by preventing the network from easily altering established categories, while a lower vigilance allows for more plasticity.

5. **Reset and Search:** If the similarity between X and c does not meet the threshold set by the vigilance, a reset mechanism is activated, temporarily disabling the currently active neuron in the recognition layer and allowing another neuron to compete for the input pattern. This process repeats until a suitable category is found or a new category is created for the input pattern.

However, simulations show that using only the top-down weights for the similarity check leads to better performance than involving bottom-up weights for the match score. Indeed, as mentioned above, bottom-up weights are part of the initial pattern recognition process, helping to identify which category an input pattern most closely matches at first glance. This step is more about recognizing patterns than the decision-making process of updating categories.

Using top-down weights for similarity prevents the model from being overly influenced by fleeting or noisy patterns that might not truly represent a category. Furthermore, a refined similarity measure ensures categories are accurately chosen for their actual resemblance to input patterns while maintaining consistent criteria. These criteria account for both the presence and absence of features and consider the overall size of the input pattern. This approach directly enforces the vigilance threshold, leading to the precise formation of categories.

Classifying patterns

The classification is performed during the training as well as the recognition of patterns. When a pattern is put into the trained ART1 network for recognition, the following steps are performed:

1. **Pattern Presentation**: The input pattern is presented to the network.
2. **Similarity Calculation**: The network calculates the similarity for each category using the top-down weights, to determine which category's template is most similar to the input pattern.
3. **Best Match Selection**: Out of all the categories, the one with the highest similarity score is initially selected as the best match.
4. **Vigilance Check**: The chosen category's similarity score is compared against the vigilance threshold. If the score is higher than or equal to the vigilance parameter, the input pattern is sufficiently similar to the category, and the classification is successful.
5. **Reset and Search**: If the similarity score does not meet the vigilance threshold, the neuron is temporarily deactivated, and the network searches for another category by recalculating similarities, excluding the previous best match.
6. **Iteration or End**: This process continues until either a category that meets the vigilance criterion is found, or all categories have been considered and none are suitable. If no suitable

category is found, we conclude that the input pattern cannot be classified based on the current categories and vigilance setting.

Simulations

In all simulations other than the individual tests, the results are based on 1,000 simulations, to provide an accurate analysis of network performance.

Training the model

The training of the model is always performed with indicating a maximum of 20 categories. Different values of vigilance are tested by plotting the stored patterns (top-down weights) sorted alphabetically. In addition, the pattern set is shuffled so that the patterns are put into the model in a different order from one training to another. **Figure 4** shows stored patterns in a network trained with a vigilance parameter $\delta = 0.5$. The network failed to store patterns ‘E’ and ‘O’ because it considered they are part of an existing category. To understand which category these patterns are mistaken with, we can plot the reshaped top-down weights as the training goes. **Figure 5** shows this evolution with an unshuffled pattern set and $\delta = 0.5$.

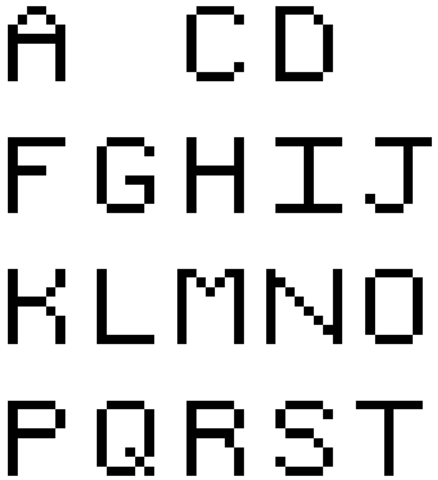


Figure 4. Stored patterns in top-down weights from a trained model with $\delta = 0.5$

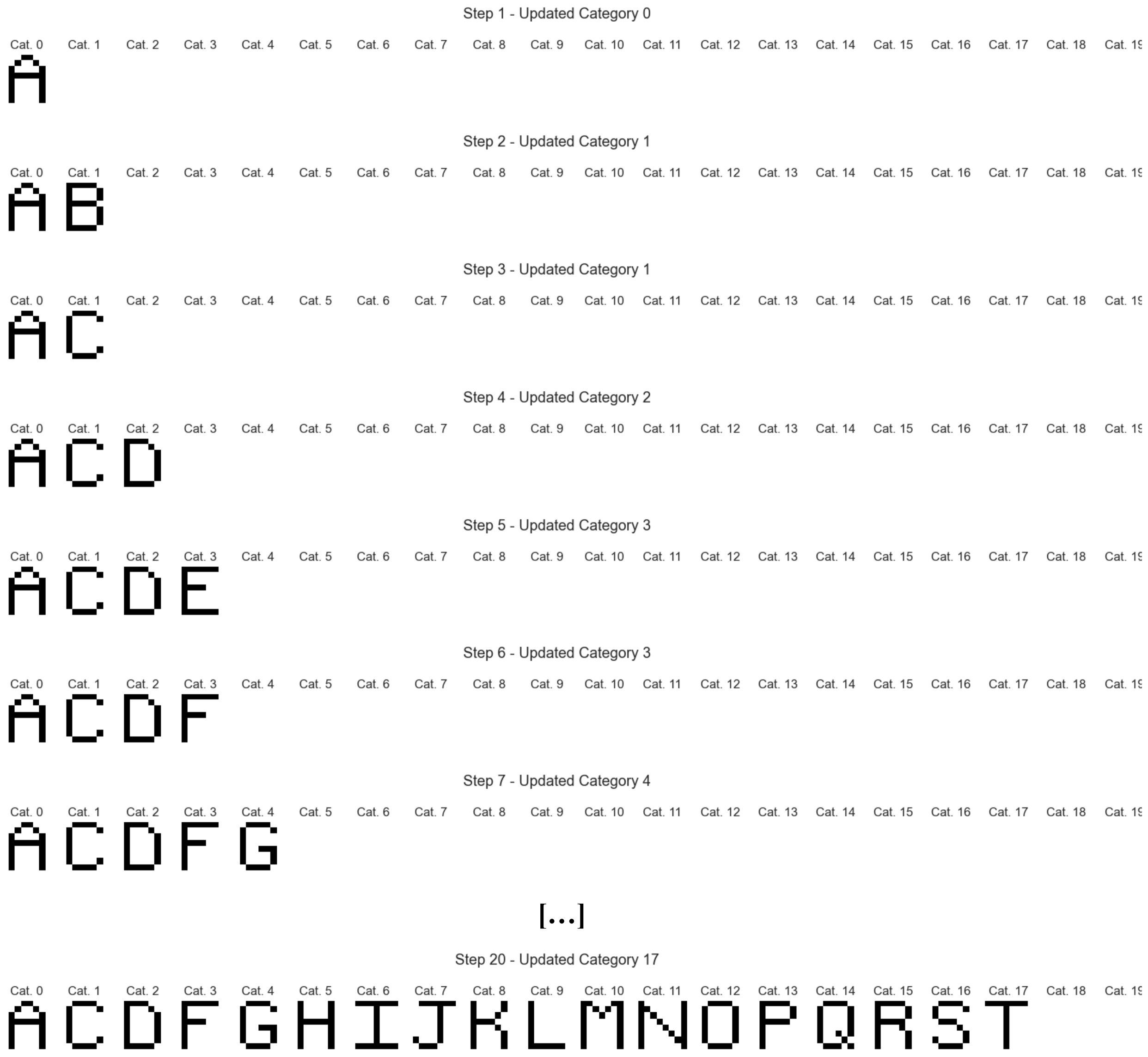

Step 1 - Updated Category 0
Step 2 - Updated Category 1
Step 3 - Updated Category 1
Step 4 - Updated Category 2
Step 5 - Updated Category 3
Step 6 - Updated Category 3
Step 7 - Updated Category 4
[…]
Step 20 - Updated Category 17
Cat. 0
Cat. 1
Cat. 2
Cat. 3
Cat. 4
Cat. 5
Cat. 6
Cat. 7
Cat. 8
Cat. 9
Cat. 10
Cat. 11
Cat. 12
Cat. 13
Cat. 14
Cat. 15
Cat. 16
Cat. 17
Cat. 18
Cat. 1Ç

Figure 5. Evolution of the stored patterns during the training phase ($\delta = 0.5$)

The 'C' was classified in the category where the 'B' was classified because the pattern 'C' is contained in the pattern 'B' in terms of pixels. In the same way, the pattern 'F' was classified in the category where the pattern 'E' was classified, overwriting the 'E' with an 'F'.

To address this issue, the training can be repeated: when the input pattern 'E' will go again through the network, it won't be recognized in any category, and therefore a new category will be created to store it. However, a problem will emerge from this fix: when a 'F' will be presented to the network, it might be recognized as a 'E' because the 'E' matches well the 'F' as it contains all the pixels. **Figure 6** shows a part of the evolution of the stored categories during the second epoch of the training phase ($\delta = 0.5$).

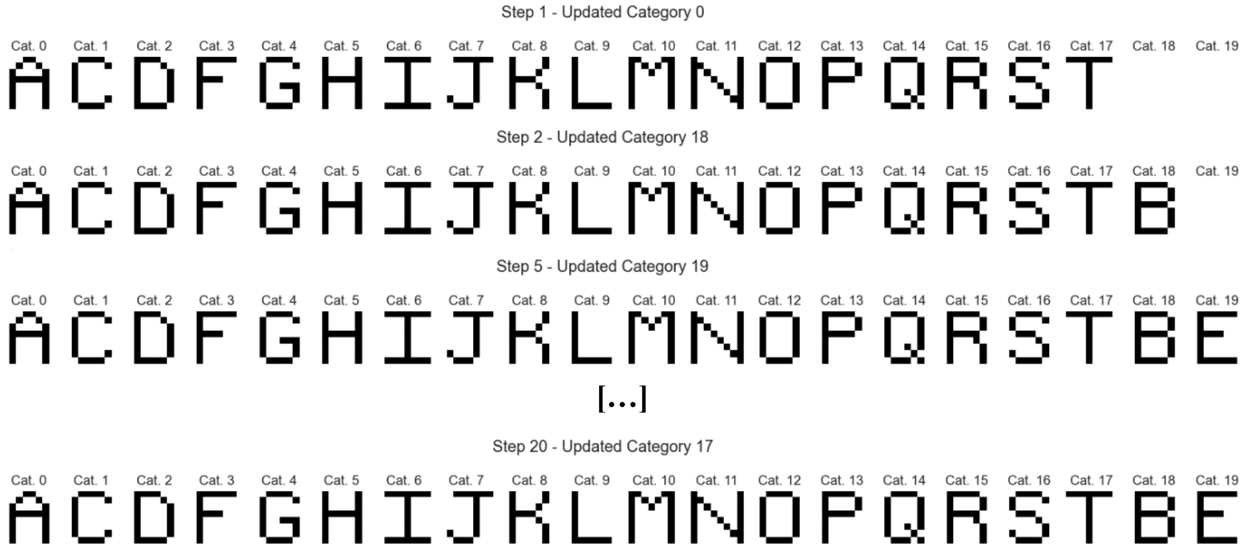

Step 1 - Updated Category 0
Cat. 0 Cat. 1 Cat. 2 Cat. 3 Cat. 4 Cat. 5 Cat. 6 Cat. 7 Cat. 8 Cat. 9 Cat. 10 Cat. 11 Cat. 12 Cat. 13 Cat. 14 Cat. 15 Cat. 16 Cat. 17 Cat. 18 Cat. 19
A C D F G H I J K L M N O P Q R S T
Step 2 - Updated Category 18
Cat. 0 Cat. 1 Cat. 2 Cat. 3 Cat. 4 Cat. 5 Cat. 6 Cat. 7 Cat. 8 Cat. 9 Cat. 10 Cat. 11 Cat. 12 Cat. 13 Cat. 14 Cat. 15 Cat. 16 Cat. 17 Cat. 18 Cat. 19
A C D F G H I J K L M N O P Q R S T B
Step 5 - Updated Category 19
Cat. 0 Cat. 1 Cat. 2 Cat. 3 Cat. 4 Cat. 5 Cat. 6 Cat. 7 Cat. 8 Cat. 9 Cat. 10 Cat. 11 Cat. 12 Cat. 13 Cat. 14 Cat. 15 Cat. 16 Cat. 17 Cat. 18 Cat. 19
A C D F G H I J K L M N O P Q R S T B E
[…]
Step 20 - Updated Category 17
Cat. 0 Cat. 1 Cat. 2 Cat. 3 Cat. 4 Cat. 5 Cat. 6 Cat. 7 Cat. 8 Cat. 9 Cat. 10 Cat. 11 Cat. 12 Cat. 13 Cat. 14 Cat. 15 Cat. 16 Cat. 17 Cat. 18 Cat. 19
A C D F G H I J K L M N O P Q R S T B E

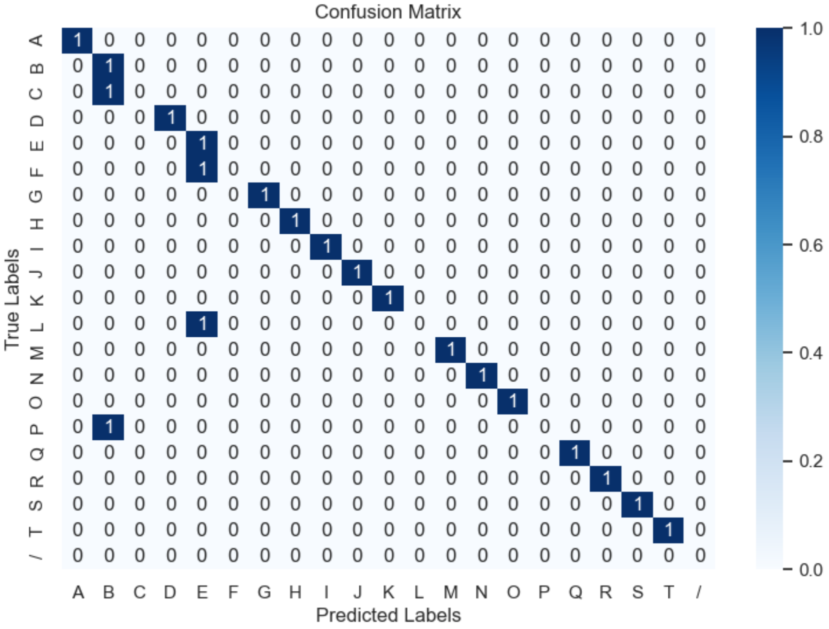
Figure 6. Evolution of the stored patterns during the second epoch of training ($\delta = 0.5$)

As expected, the missing categories are stored in the network during this second epoch.

Recognizing clean patterns

Recognizing clean patterns allows to test the basic network performance. The original pattern set is put into the trained network and each pattern is categorized. The resulting confusion matrix is shown in **Figure 7**. We observed that while most patterns are correctly clustered, the 'C' and 'P' were recognized as 'B', as the pattern 'B' includes both patterns 'C' and 'P'. Furthermore, 'L' and 'F' were recognized as 'E'. The overall accuracy of the network tested with the clean patterns is therefore 80%.

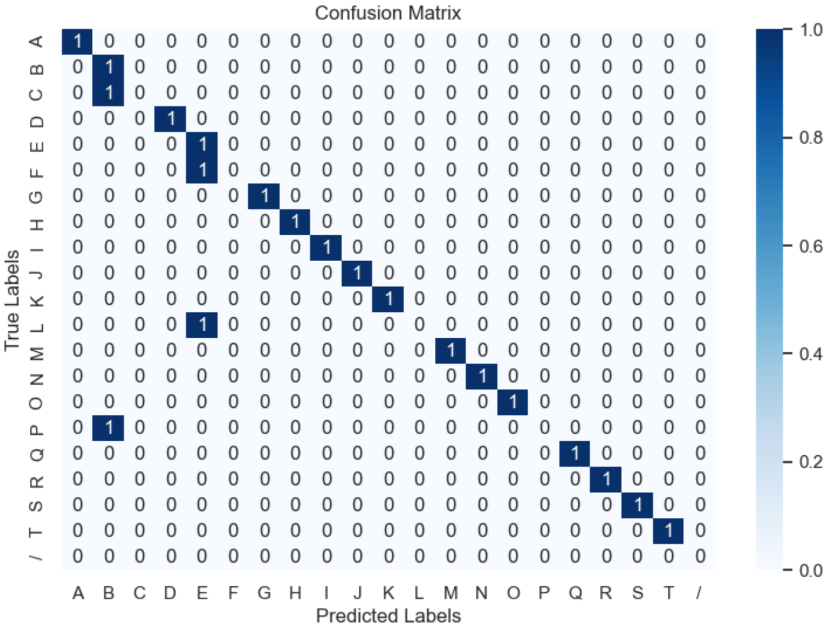
From this analysis, we can conclude that the value of the vigilance δ is too low to recognize perfectly clean patterns and is therefore not strict enough. Indeed, by adjusting the vigilance parameter, we can control how strictly the network classifies new input patterns, either maintaining strict category integrity (high values) or allowing more generalization (low values).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	/
A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7. Confusion Matrix resulting from 1 simulation of classifying clean patterns ($\delta = 0.5$)

Another simulation was performed with a vigilance $\delta = 0.95$, and the resulting confusion matrix is shown in **Figure 8**. The issue encountered is that this value of vigilance yields to the same result as before.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	/
A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8. Confusion Matrix resulting from 1 simulation of classifying clean patterns ($\delta = 0.95$)

We can take different approaches to improve the model: improving the weights update, the weights initialization, or the similarity calculation. We focused on the similarity calculation, adding a new hyperparameter called penalty rate p. The idea is to decrease the similarity score when the input pattern has a 0 (absence of a feature) where the category's top-down weights have a 1 (presence of a feature). This penalty is calculated by counting the number of such mismatches, multiplying by the penalty rate, and then subtracting this value from the intersection (the count of matching features). It helps penalizing the categories that include many patterns. For example, the 'E' contains the 'F' and the 'L'; with the new model, the pixels that are not in common are penalizing the similarity score.

The confusion matrix resulting from a simulation of the new model with $\delta = 0.5$ and $p = 0.001$ is shown in **Figure 9**. The overall accuracy of the model is 100%, thus the penalization of uncommon pixels is effective.

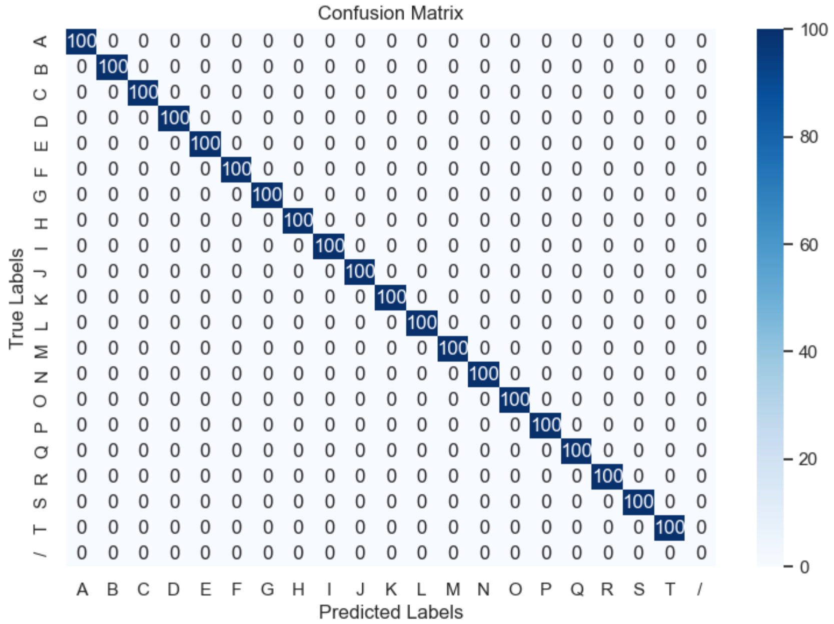


Figure 9. Confusion Matrix resulting from 100 simulation of classifying clean patterns ($\delta = 0.5$, $p = 0.001$)

Recognizing noisy patterns

The model is trained using clean patterns, but the testing phase uses noisy patterns to assess the capacity of generalization and the true performance of the model.

Noise is introduced into each pattern according to a noise ratio that defines how many pixels are randomly flipped. Flipping a pixel means performing a bitwise NOT operation. **Figure 10** shows a set of noisy patterns with a noise ratio of 25%.

First, the performance was evaluated without the penalty in the similarity calculation with $\delta = 0.5$, and with a 10% noise ratio, as illustrated in **Figure 11**. We observe that several patterns are recognized as a 'B', which leads to a bad precision for this category: among the positive predictions, few of them are correct. Besides, some patterns are not recognized at all, such as 'C', 'F', 'L' and 'P', because they are "included" in patterns that contain them.

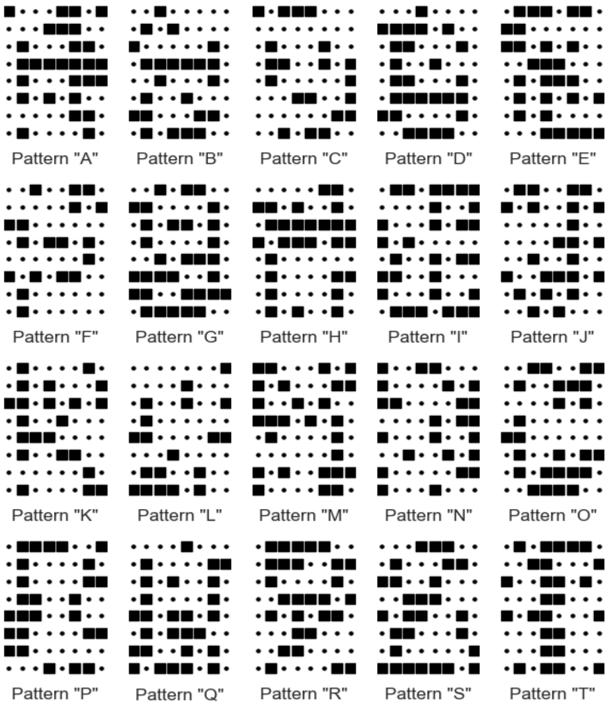
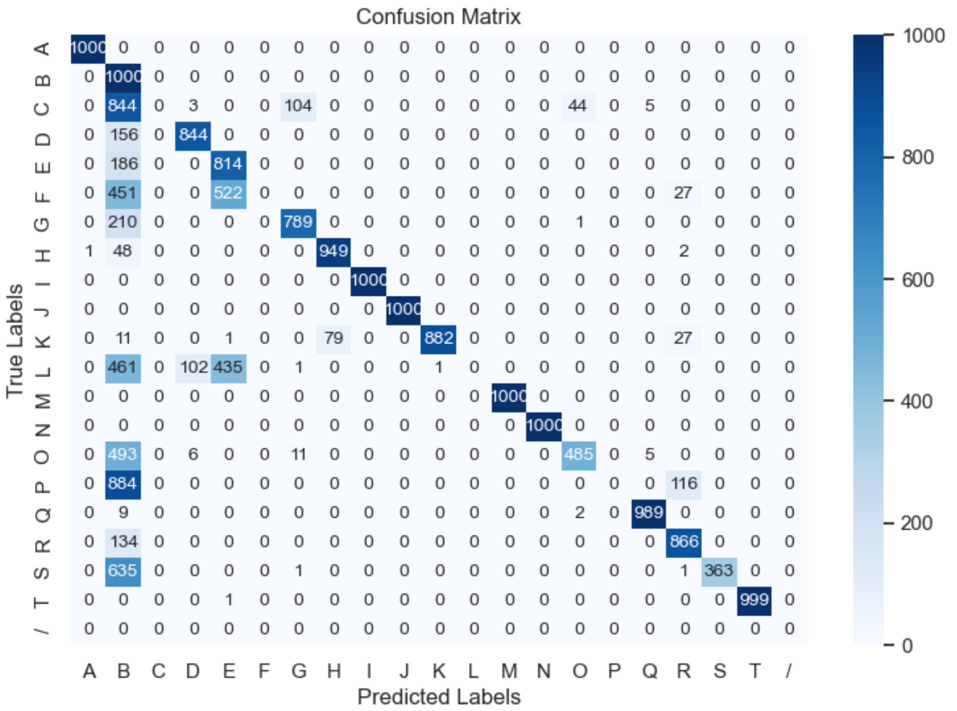


Figure 10. Noisy pattern set (25% noise ratio)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	/
A	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	844	0	3	0	0	104	0	0	0	0	0	0	0	44	0	5	0	0	0	0
D	0	156	0	844	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	186	0	0	814	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	451	0	0	522	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0
G	0	210	0	0	0	0	789	0	0	0	0	0	0	0	1	0	0	0	0	0	0
H	1	48	0	0	0	0	0	949	0	0	0	0	0	0	0	0	0	2	0	0	0
I	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0
K	0	11	0	0	1	0	0	79	0	0	882	0	0	0	0	0	0	27	0	0	0
L	0	461	0	102	435	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0
O	0	493	0	6	0	0	11	0	0	0	0	0	0	0	485	0	5	0	0	0	0
P	0	884	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	116	0	0	0
Q	0	9	0	0	0	0	0	0	0	0	0	0	0	0	2	0	989	0	0	0	0
R	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	866	0	0	0
S	0	635	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	363	0	0
T	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	999	0
/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

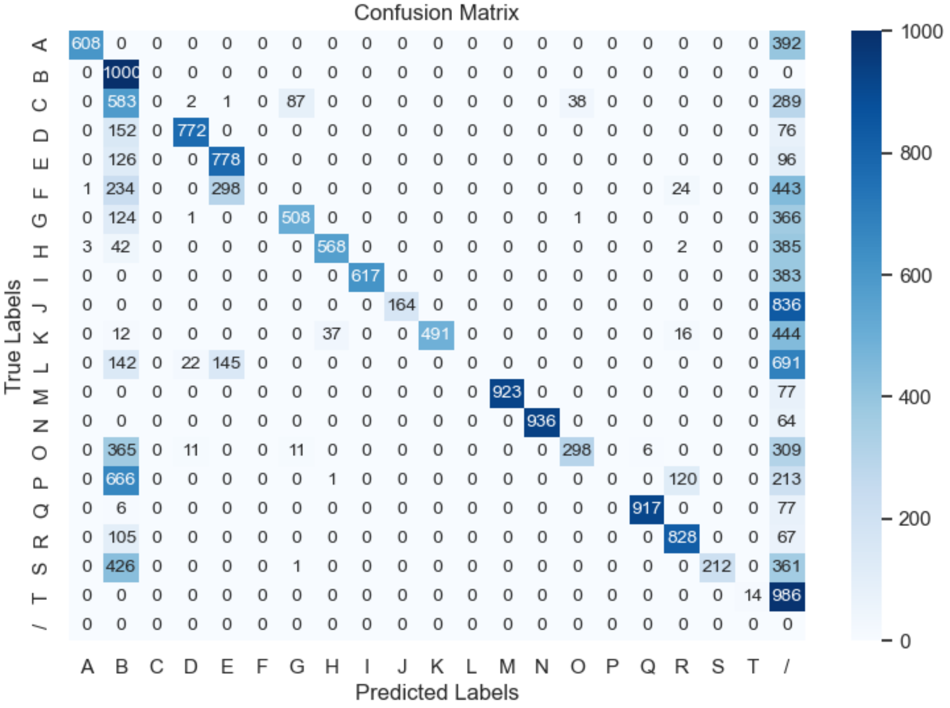
Figure 11. Confusion Matrix resulting from 1,000 simulations of recognizing noisy patterns (10% noise) ($\delta = 0.5$, $p = 0$)

A comparison of different metrics for patterns 'B' and 'L' are shown in **Table 1**. The recall of class 'B' is perfect, because 100% of patterns 'B' were recognized as such, but the precision is low (18.11%), as many non-'B' patterns were recognized as 'B', which leads to a low F_1 score. Regarding the class 'L', the accuracy is 95.00% since true negatives are numerous: The more classes there are in a classification problem, the less relevant the accuracy since true positives are no longer important compared with true negatives. However, no 'L' pattern was recognized, resulting in a recall of 0%, and the precision cannot be calculated (or can be considered zero) since there are no positives.

Table 1. Metrics for patterns 'B' and 'L' resulting from 1,000 simulations of recognizing noisy patterns (10% noise) ($\delta = 0.5$, $p = 0$)

	Pattern 'B'	Pattern 'L'
Accuracy	77.39%	95.00%
Precision	18.11%	N/A
Recall	100.00%	0.00%
$\mathbf{F_1}$	30.67%	0.00%

A solution to avoid recognizing patterns that contain other patterns (such as ‘B’) would be to increase the vigilance. **Figure 12** shows the confusion matrix resulting from 1,000 simulations on noisy patterns (10% noise) with $\delta = 0.8$.

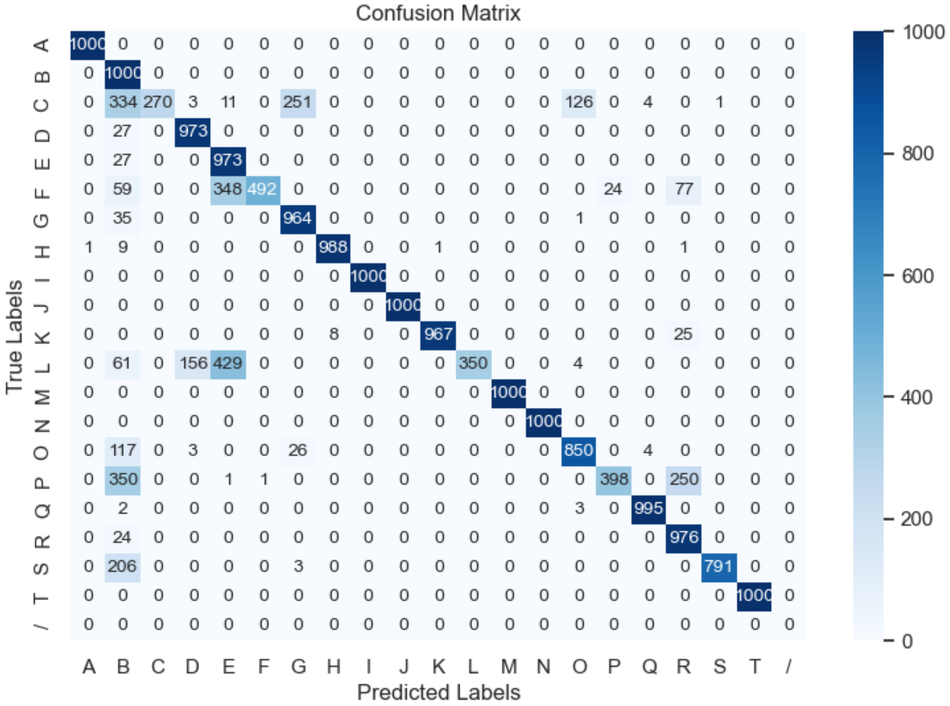


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	/
A	608	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	392
B	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	583	0	2	1	0	87	0	0	0	0	0	0	0	38	0	0	0	0	0	289
D	0	152	0	772	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	76
E	0	126	0	0	778	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96
F	1	234	0	0	298	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	443
G	0	124	0	1	0	0	508	0	0	0	0	0	0	0	1	0	0	0	0	0	366
H	3	42	0	0	0	0	0	568	0	0	0	0	0	0	0	0	0	2	0	0	385
I	0	0	0	0	0	0	0	0	617	0	0	0	0	0	0	0	0	0	0	0	383
J	0	0	0	0	0	0	0	0	0	164	0	0	0	0	0	0	0	0	0	0	836
K	0	12	0	0	0	0	0	37	0	0	491	0	0	0	0	0	0	16	0	0	444
L	0	142	0	22	145	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	691
M	0	0	0	0	0	0	0	0	0	0	0	0	923	0	0	0	0	0	0	0	77
N	0	0	0	0	0	0	0	0	0	0	0	0	0	936	0	0	0	0	0	0	64
O	0	365	0	11	0	0	11	0	0	0	0	0	0	0	298	0	6	0	0	0	309
P	0	666	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	120	0	0	213
Q	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	917	0	0	0	77
R	0	105	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	828	0	0	67
S	0	426	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	212	0	361
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	986
/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12. Confusion Matrix resulting from 1,000 simulations of recognizing noisy patterns (10% noise) ($\delta = 0.8$, $p = 0$)

Although some patterns are no longer recognized as ‘B’, they are still not correctly recognized, and the classification leads to them being out-of-category, i.e. it can't find a category that satisfies the vigilance criterion for classifying the pattern. This is highlighted in the confusion matrix in the last column. This problem further justifies the improvement we have made to the similarity calculation, which penalizes patterns that contain others (in terms of pixels). Note that the precision for pattern ‘B’ is higher with this higher vigilance, at 25.19%, 5 points higher than with a vigilance of 0.5.

By introducing the penalty, the model becomes much more efficient, as shown in **Figure 13**. Indeed, the overall accuracy of the model is better (84.94%), but struggling patterns such as ‘C’, ‘F’, ‘L’, and ‘P’ are still problematic. A comparison of metrics for patterns ‘B’ and ‘L’ is given to evaluate the influence of the penalty over them. The precision of pattern ‘B’ increased, resulting in a better F_1 score (+31%). Moreover, the precision of pattern 'L' is 100.0% because all positives are true, and the recall increased to 35%, which means 35% of patterns 'L' are correctly recognized as such.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	/
A	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	334	270	3	11	0	251	0	0	0	0	0	0	0	126	0	4	0	1	0	0
D	0	27	0	973	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	27	0	0	973	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	59	0	0	348	492	0	0	0	0	0	0	0	0	0	24	0	77	0	0	0
G	0	35	0	0	0	0	964	0	0	0	0	0	0	0	1	0	0	0	0	0	0
H	1	9	0	0	0	0	0	988	0	0	1	0	0	0	0	0	0	1	0	0	0
I	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	8	0	0	967	0	0	0	0	0	0	25	0	0	0
L	0	61	0	156	429	0	0	0	0	0	0	350	0	0	4	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0
O	0	117	0	3	0	0	26	0	0	0	0	0	0	0	850	0	4	0	0	0	0
P	0	350	0	0	1	1	0	0	0	0	0	0	0	0	0	398	0	250	0	0	0
Q	0	2	0	0	0	0	0	0	0	0	0	0	0	0	3	0	995	0	0	0	0
R	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	976	0	0	0
S	0	206	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	791	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0
/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13. Confusion Matrix resulting from 1,000 simulations of recognizing noisy patterns (10% noise) ($\delta = 0.5$, $p = 0.001$)

Table 2. Metrics for patterns 'B' and 'L' resulting from 1,000 simulations of recognizing noisy patterns (10% noise) ($\delta = 0.5$, $p = 0.001$)

	Pattern 'B'	Pattern 'L'
Accuracy	93.75%	96.75%
Precision	44.42%	100.00%
Recall	100.00%	35.00%
F_1	61.52%	51.85%

A first interesting analysis is to study the influence of the vigilance over the accuracy of the model, as the overall accuracy is a first-resort indicator (**Figure 14**). We observe that the accuracy drops when the vigilance exceeds 0.7: the model becomes too strict and cannot categorize any noisy pattern. Note that with low vigilance values, the model does not lose performance, which means that the categories learned are the same for these values of vigilance.

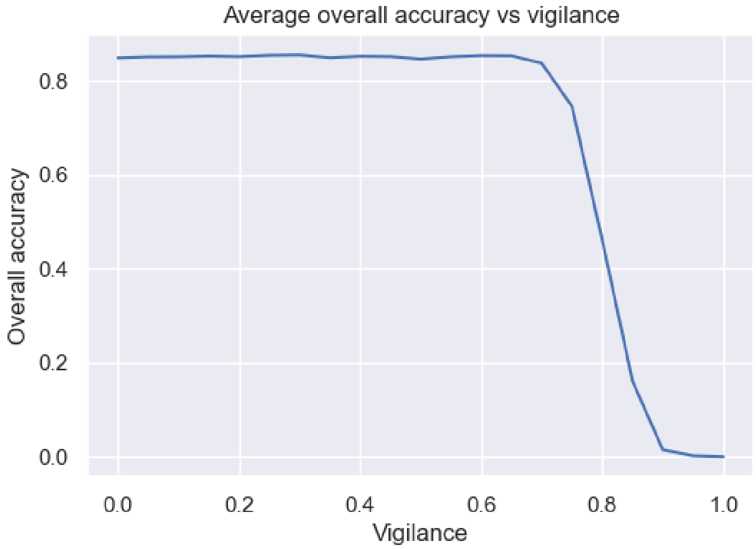


Figure 14. Overall accuracy vs vigilance δ with $p = 0.001$, 10% noise, average of 1,000 simulations

Another analysis of the model is to evaluate its overall performance against the noise ratio, as shown in **Figure 15**.

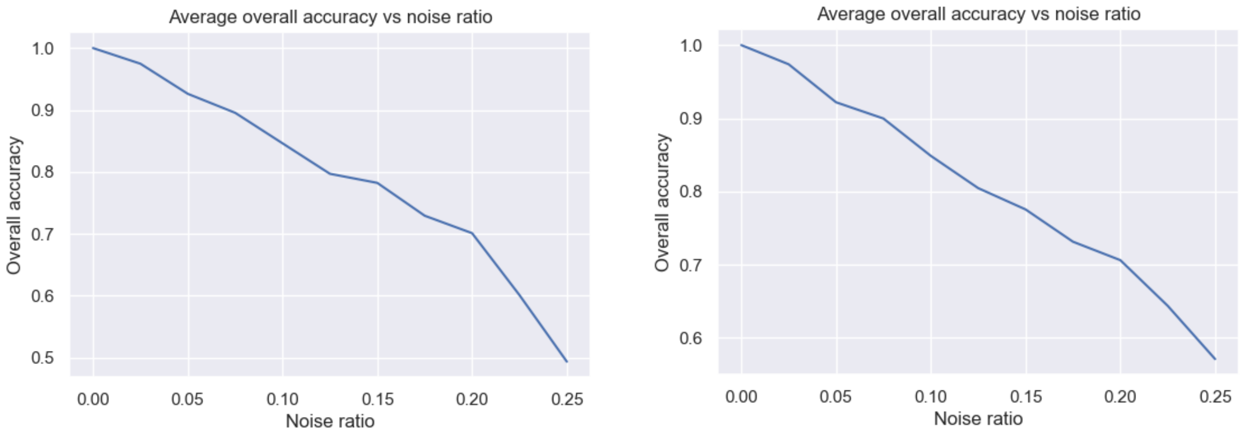


Figure 15. Overall accuracy vs noise ratio with $p = 0.001$, average of 1,000 simulations $\delta = 0.5$ on the left and $\delta = 0.2$ on the right

The overall accuracy decreases as noise increases, and it is interesting to note that a smaller vigilance (meaning a more tolerant network) results in a little better accuracy (57.11% for $\delta = 0.2$ against 49.73% for $\delta = 0.5$).

Finally, we need to study the influence of the penalty rate on the accuracy. Indeed, the value of p has not been taken at random in this study, since the penalty must have sufficient weight to have an influence on the model, but not too great so as not to reduce the similarity too much and fail all the vigilance checks.

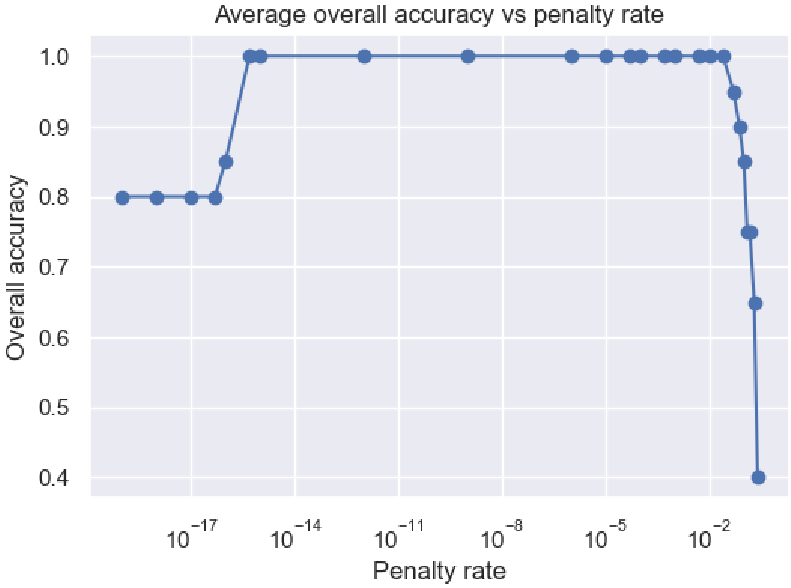


Figure 16. Overall accuracy vs penalty rate resulting from 1,000 simulations with $\delta = 0.5$ and clean patterns

A wide range of penalty rate values gives high performance (accuracy to 100%): we need to choose a penalty rate between 10^{-15} and 10^{-2} to get the best results for clean patterns. When the penalty rate is too small, there is barely no penalty and the default similarity is effective. When the penalty rate is too high, it penalizes most patterns so much that the similarity falls well below the vigilance and the accuracy drops.

Conclusion

In this study, we explored the concept of Adaptive Resonance Theory (ART), focusing on its application in pattern recognition tasks. Our experimental setup involved a set of binary matrix representations for the alphabets A to T, which served as the basis for our investigations into the network's recognition capabilities.

The training phase's efficacy was evaluated based on the network's ability to accurately categorize the patterns, revealing insights into the model's performance. Subsequently, the network's performance was tested under conditions of recognizing clean patterns, where it demonstrated an 80% overall accuracy. This phase highlighted the network's baseline capability in pattern recognition without the interference of noise.

The recognition of noisy patterns introduced a new layer of complexity, where patterns were altered with a specific noise ratio to assess the network's robustness and adaptability. Under these conditions, the model's performance varied, indicating a nuanced response to increasing levels of noise. An important aspect of our analysis focused on the impact of the vigilance parameter and noise levels on the network's performance. Adjusting the vigilance parameter allowed us to explore the network's adaptability to new patterns. The introduction of a penalty rate in the similarity calculation emerged as a significant enhancement, improving the network's discrimination capabilities between similar patterns, resulting in a better accuracy.

Future research could explore possible enhancements for this model, such as tuning the vigilance parameter during the classification if the input pattern is not recognized.

I certify that this submission is my original work and meets the Faculty's Expectations of Originality.

April 3, 2024

Benjamin Vauchel

Vauchel