

# RÉSEAUX DE NEURONES ARTIFICIELS

*pour la détection d'anomalies*



# SOMMAIRE

01

## Apprentissage supervisé des PMC

Implémentation sur R du PMC pour la détection d'anomalies

02

## Présentation des GANs

Generative Adversarial Networks

03

## Implémentation d'un GAN

Implémentation sur Python d'un GAN pour la détection d'anomalies dans un jeu de données industriel

04

## Implémentation d'une carte de Kohonen

Implémentation sur Python pour la détection d'anomalies en cybersécurité

05

## Implémentation d'un DBN

Implémentation sur Python pour la détection d'anomalies en cybersécurité

Questions 1 & 2

Question 5

Question 3

Question 4

Question 6

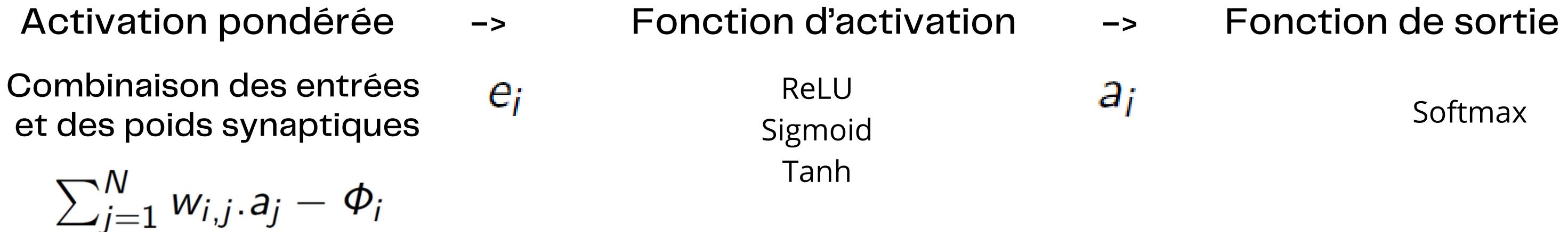
# APPRENTISSAGE SUPERVISÉ DES PMC

Fonctionnement en deux modes :

- **Mode de reconnaissance** : Propagation de l'information pour produire une sortie à partir d'une entrée
- **Mode d'apprentissage** : Ajustement des poids pour adapter le réseau à une tâche donnée

Plusieurs couches : couche d'entrée, cachées et de sortie

Communication via des fonctions : fonction d'entrée  $h$ , fonctions d'activation  $a$ , fonction de sortie



# APPRENTISSAGE SUPERVISÉ DES PMC

- Propagation avant : L'entrée  $x$  est propagée à travers les couches pour produire une sortie. L'activation  $h(i)$  est calculée pour chaque neurone  $i$  puis  $h(i)$  est transformée en une sortie pour chaque neurone  $a_i = F(h(i))$ .
- Calcul de l'erreur : La différence entre la sortie réelle  $a_i$  et la sortie attendue  $d_i$  est mesurée par une fonction d'erreur, souvent l'erreur quadratique :  $E(a, d) = \frac{1}{2} \sum_{i=1}^m (a_i - d_i)^2$ , où  $m$  est le nombre de neurones en sortie.
- Rétropropagation de l'erreur : L'erreur est propagée en sens inverse pour ajuster les poids synaptiques. La backpropagation repose sur la descente du gradient, en utilisant la règle de la chaîne à commencer par les couches de sortie pour remonter vers les couches cachées :  $\Delta w_{i,j} = -\lambda \frac{\delta E}{\delta w_{i,j}}$ , où  $\lambda$  est le pas d'apprentissage.

propagation avant → calcul de l'erreur → rétropropagation  
répété sur plusieurs cycles (epochs) jusqu'à minimisation de l'erreur



# QU'EST CE QUE *la détection d'anomalies*

01

## Apprentissage supervisé des PMC

Implémentation sur R du  
PMC pour la détection  
d'anomalies

Question 1 & 2



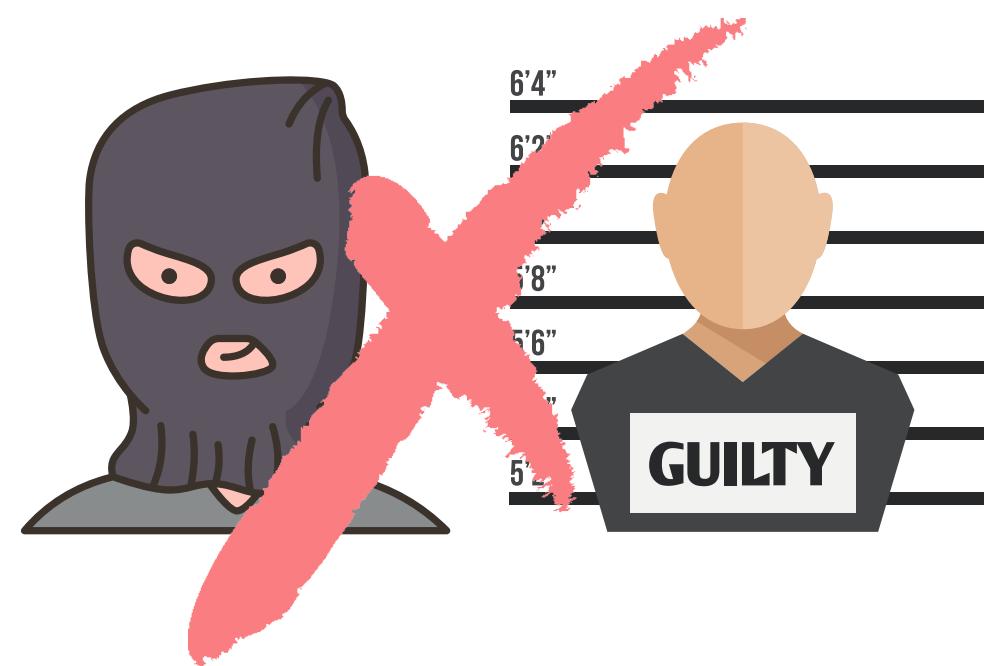
# QU'EST CE QUE *la détection d'anomalies*

01

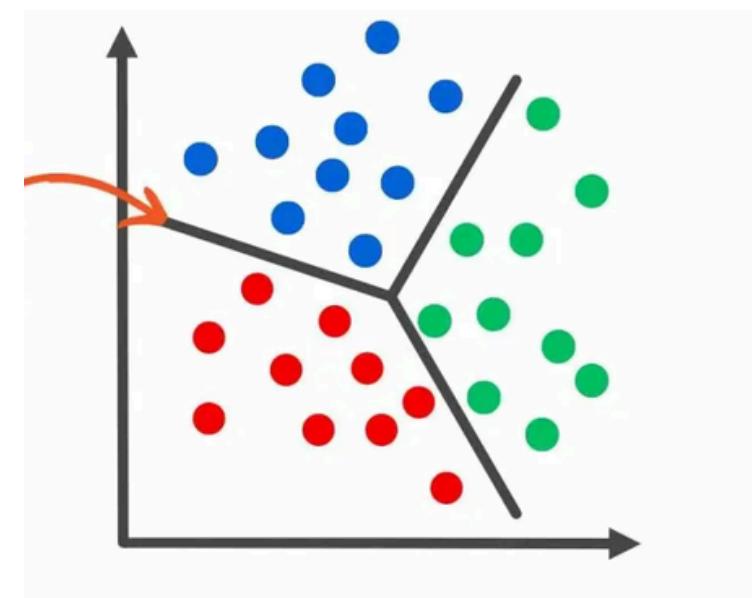
## Apprentissage supervisé des PMC

Implémentation sur R du  
PMC pour la détection  
d'anomalies

Question 1 & 2



En réalité, tout peut être considéré comme anomalie -> **Importance du contexte**

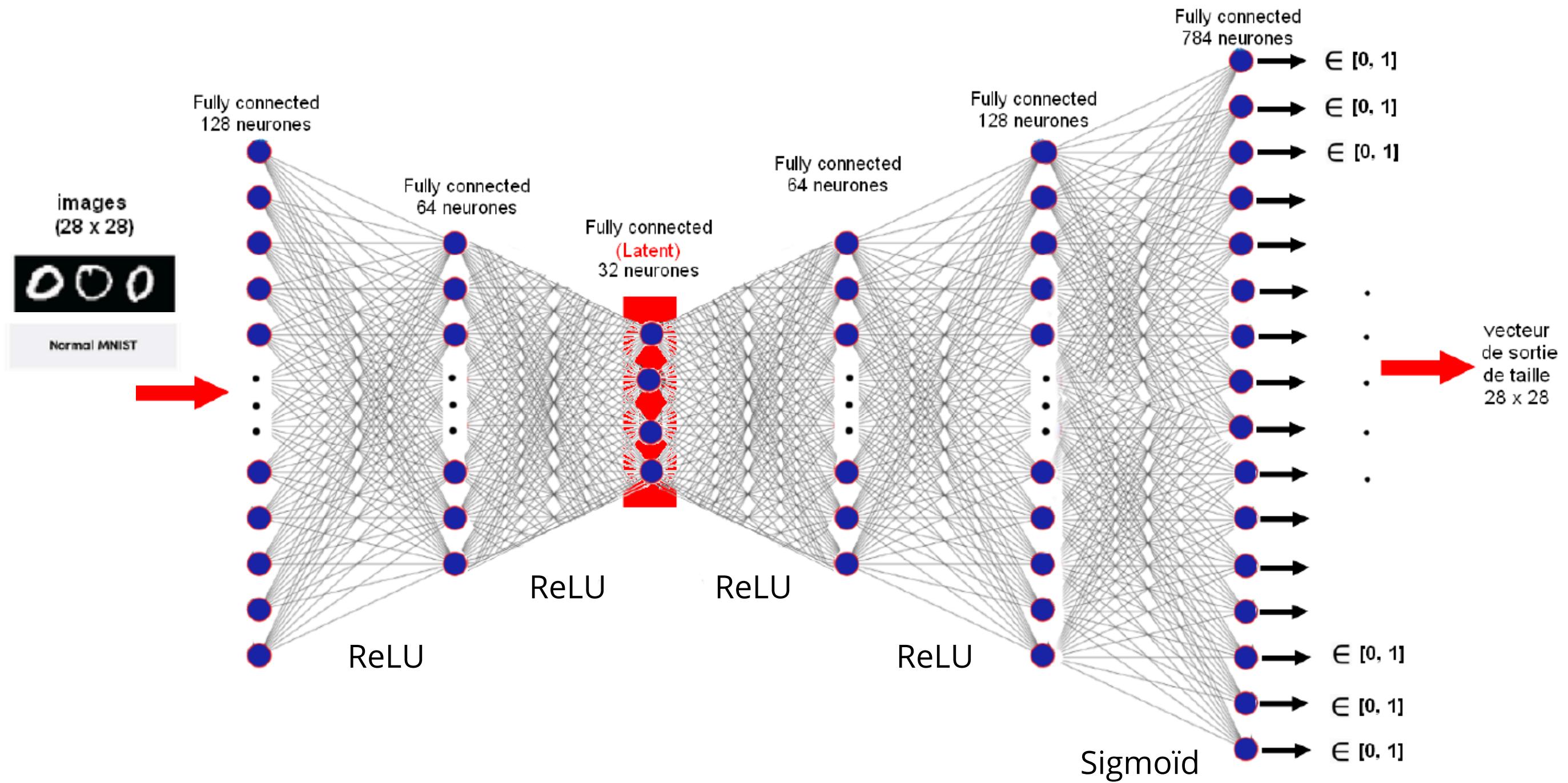


# IMPLÉMENTATION DU PMC

pour la détection d'anomalies



Question 1 & 2



# RÉSULTATS DU PMC

*pour la détection d'anomalies*

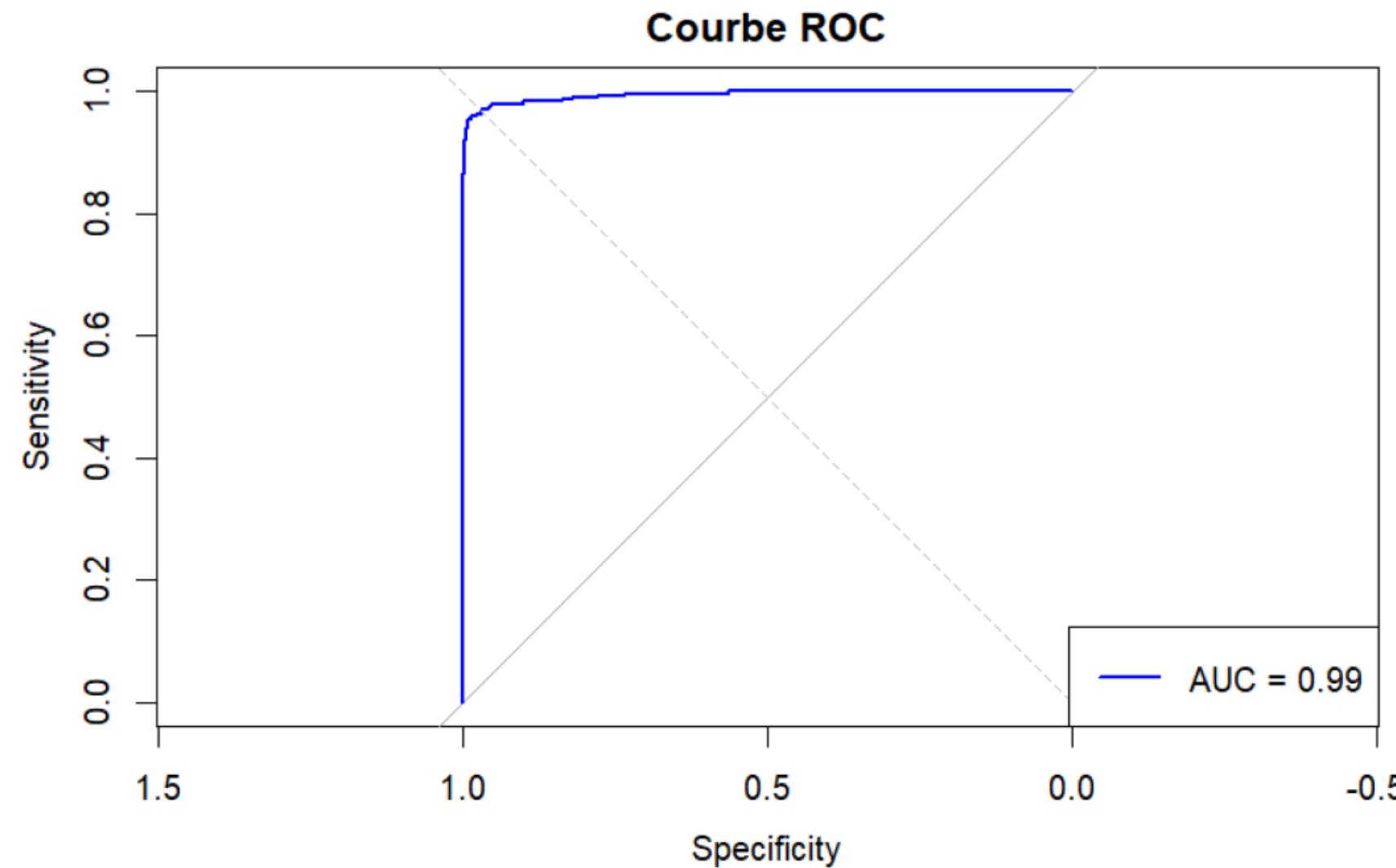
01

## Apprentissage supervisé des PMC

Implémentation sur R du PMC pour la détection d'anomalies

Question 1 & 2

```
31/31 [=====] - 0s 6ms/step
282/282 [=====] - 1s 3ms/step
Seuil d'anomalie: 24.32507
Taux de détection d'anomalies parmi les données normales: 0.05
Taux de détection d'anomalies parmi les anomalies: 0.9900222
```



Confusion Matrix and Statistics

		Reference	
		0	1
Prediction	0	45	10
	1	5	40

Rapport de Classification :  
Précision : 0.82  
Rappel : 0.9  
F1-Score : 0.86

# LA DISTANCE DE MAHALANOBIS

*pour la détection d'anomalies*

01

Comparaison avec  
une méthode  
statistique

La distance de Mahalanobis d'un vecteur à plusieurs variables  $x = (x_1, x_2, x_3, \dots, x_p)^T$  à un vecteur de moyennes  $\mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_p)^T$  et possédant une matrice de covariance  $\Sigma$  est définie comme suit :

$$D_M(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}.$$

**Avantages :**

Question 1 & 2

Elle tient compte des corrélations entre les variables et de la variance des données dans leur espace multidimensionnel.

# COMPARAISON PMC VS DISTANCE DE MAHALANOBIS

*pour la détection d'anomalies*

01

Comparaison avec  
une méthode  
statistique

Question 1 & 2

## PERCEPTRON MULTI-COUCHES

Seuil d'anomalie: 24.32507

Taux de détection d'anomalies parmi les données normales: 0.05

Taux de détection d'anomalies parmi les anomalies: 0.9900222

## DISTANCE DE MAHALANOBIS

Seuil d'anomalie : 27.99553

Taux de détection des anomalies parmi les données normales : 0.05

Taux de détection des anomalies parmi les anomalies : 0.25

# PRÉSENTATION DES GANs

10



Question 5

- Le **générateur** : Son objectif est de créer des données synthétiques qui ressemblent le plus possible aux données réelles. Il prend comme entrée un vecteur de bruit aléatoire issu d'une distribution gaussienne ou uniforme et le transforme en une donnée générée.
- Le **discriminateur** : Ce réseau est un classificateur binaire. Il apprend à distinguer les données générées par le générateur des données réelles issues du dataset d'entraînement. Son rôle est de "critiquer" les données produites par le générateur.

Le processus d'apprentissage repose sur un **jeu minimax** entre les deux réseaux :

- Le générateur tente de tromper le discriminateur en produisant des données réalistes.
- Le discriminateur cherche à détecter les faux parmi les vrais.

# IMPLÉMENTATION D'UN GAN

03

## Implémentation d'un GAN

Implémentation sur Python d'un GAN pour la détection d'anomalies dans un jeu de données industriel

Question 3

### MVTec Anomaly Dataset

#### Pill

```

pill/
└── ground_truth/
    ├── color (24 images)
    ├── combined (16 images)
    ├── contamination (20 images)
    ├── crack (25 images)
    ├── faulty_imprint (18 images)
    ├── pill_type (8 images)
    └── scratch (23 images)

    test/
        ├── color (24 images)
        ├── combined (16 images)
        ├── contamination (20 images)
        ├── crack (25 images)
        ├── faulty_imprint (18 images)
        ├── good (25 images)
        ├── pill_type (8 images)
        └── scratch (23 images)

    train/
        └── good/
            └── good (266 images)

```

Exemples d'anomalies  
de couleur :



Exemples d'anomalies  
de type :



Exemples d'anomalies  
de contamination :



Exemples de masks  
dans le dossier  
ground\_truth :

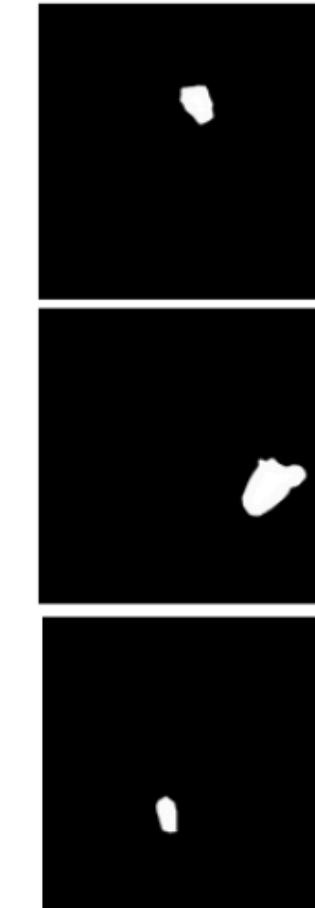


Figure 3.1 : Extrait du dataset Pill de MVTec

# ENTRAINEMENT DU GAN

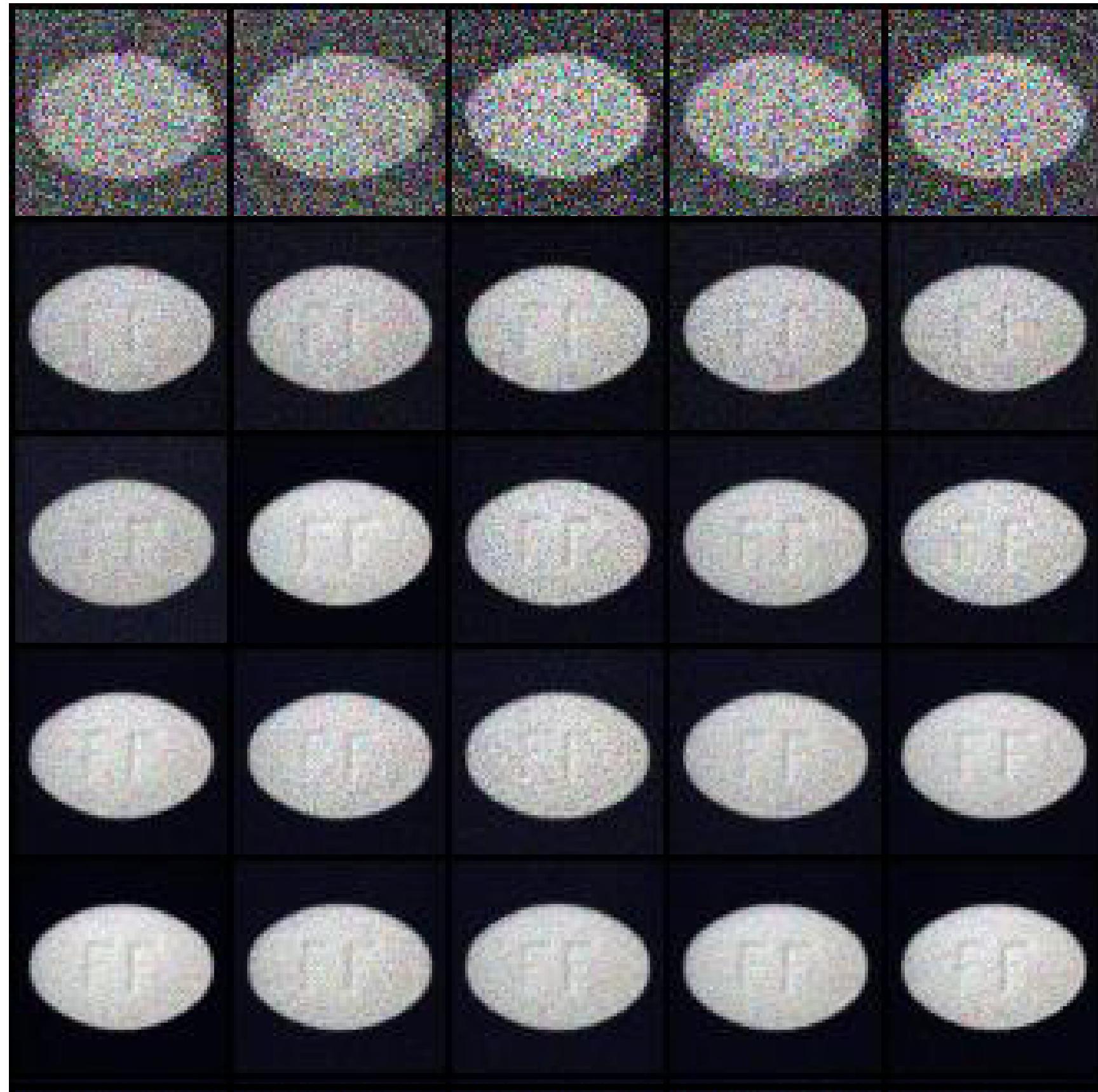
12

03

## Implémentation d'un GAN

Implémentation sur  
Python d'un GAN pour la  
déttection d'anomalies  
dans un jeu de données  
industriel

Question 3



epoch 10

epoch 200

epoch 300

epoch 400

epoch 500

# ÉVALUATION DU GAN

03

## Implémentation d'un GAN

Implémentation sur Python d'un GAN pour la détection d'anomalies dans un jeu de données industriel

Question 3

### Intersection over Union (IoU)

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}}$$

Cette métrique mesure le rapport entre la zone d'intersection entre la segmentation prédite et la vérité terrain (*ground truth*) et la zone totale couverte par les deux (*union*).

### Dice Score (Coefficient de Similarité de Sørensen-Dice)

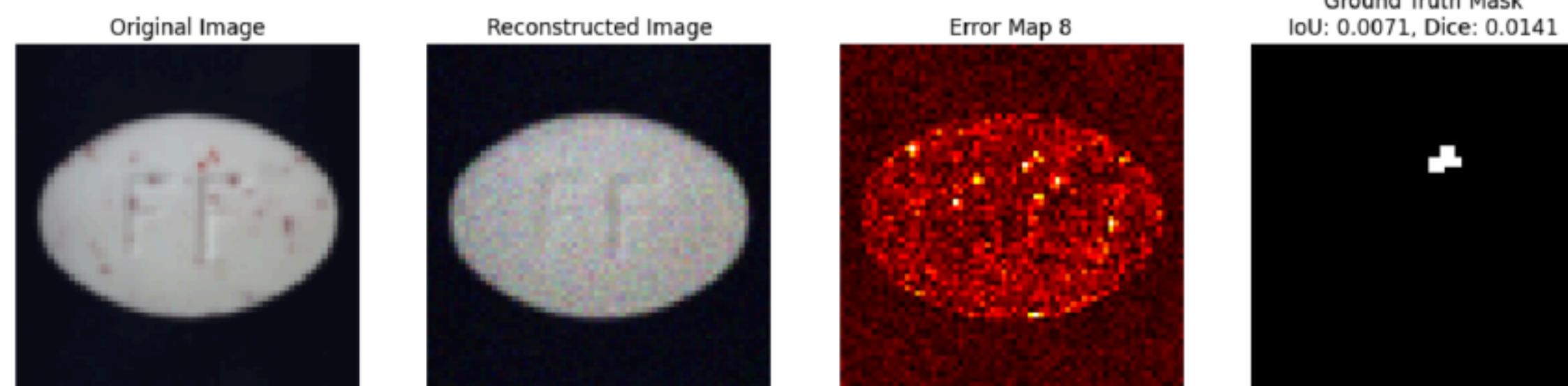
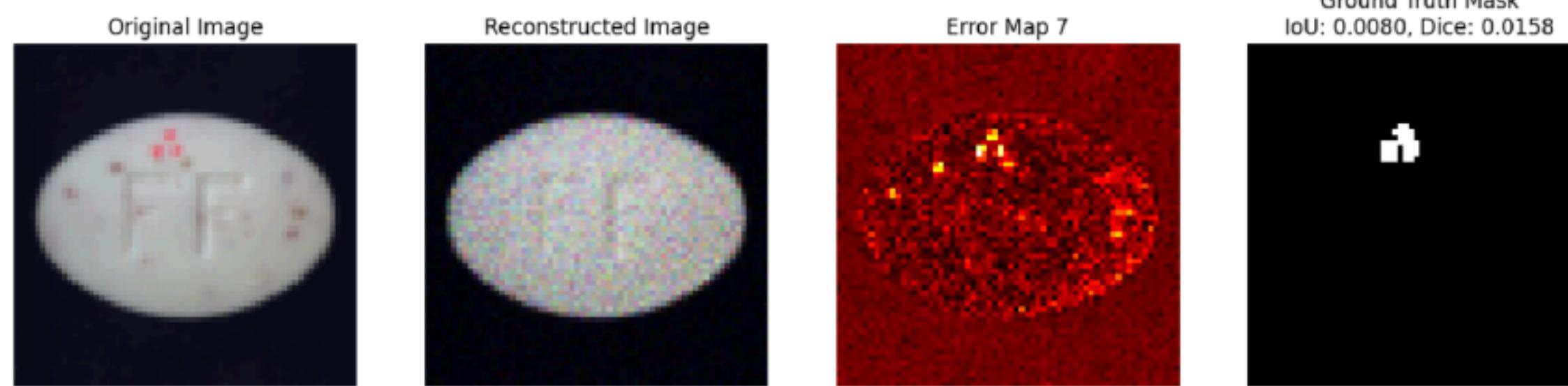
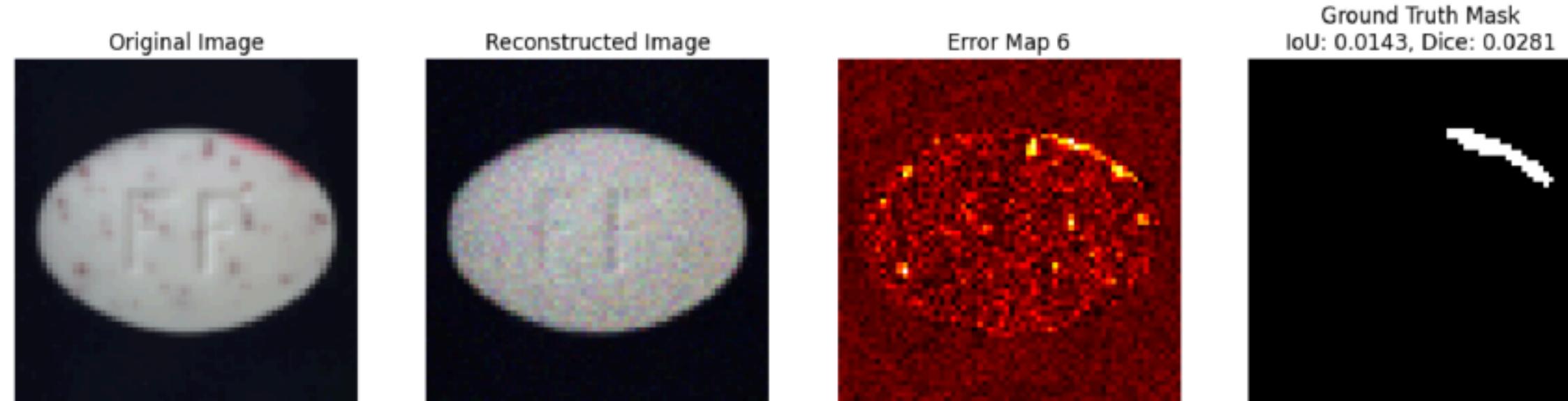
$$\text{Dice} = \frac{2 \times \text{Intersection}}{\text{Cardinalité Prédite} + \text{Cardinalité Vérité Terrain}}$$

03

### Implémentation d'un GAN

Implémentation sur Python d'un GAN pour la détection d'anomalies dans un jeu de données industriel

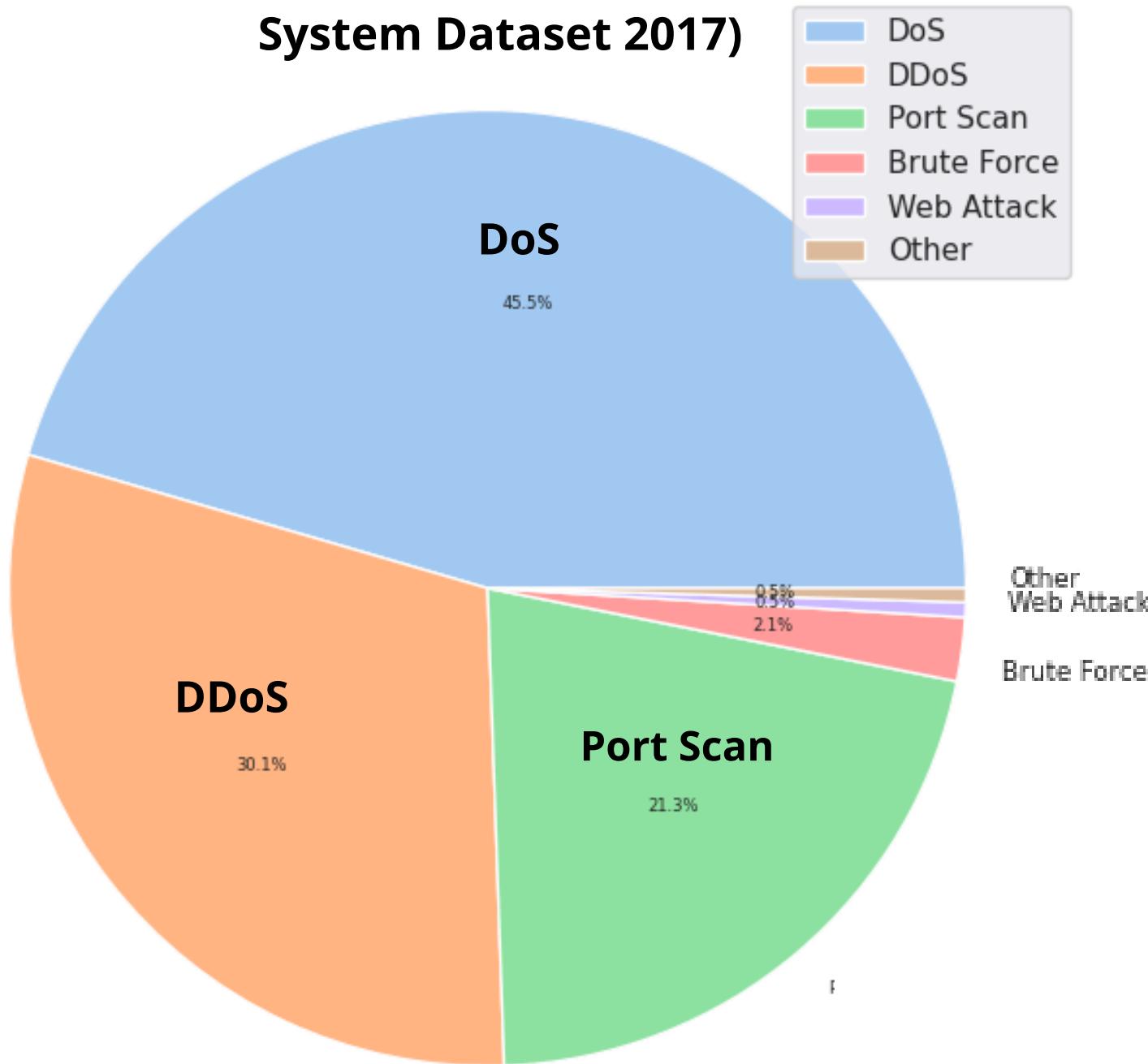
Question 3



# IMPLÉMENTATION D'UNE CARTE DE KOHONEN

15

## CICIDS2017 (Canadian Institute for Cybersecurity Intrusion Detection System Dataset 2017)



04

### Implémentation d'une carte de Kohonen

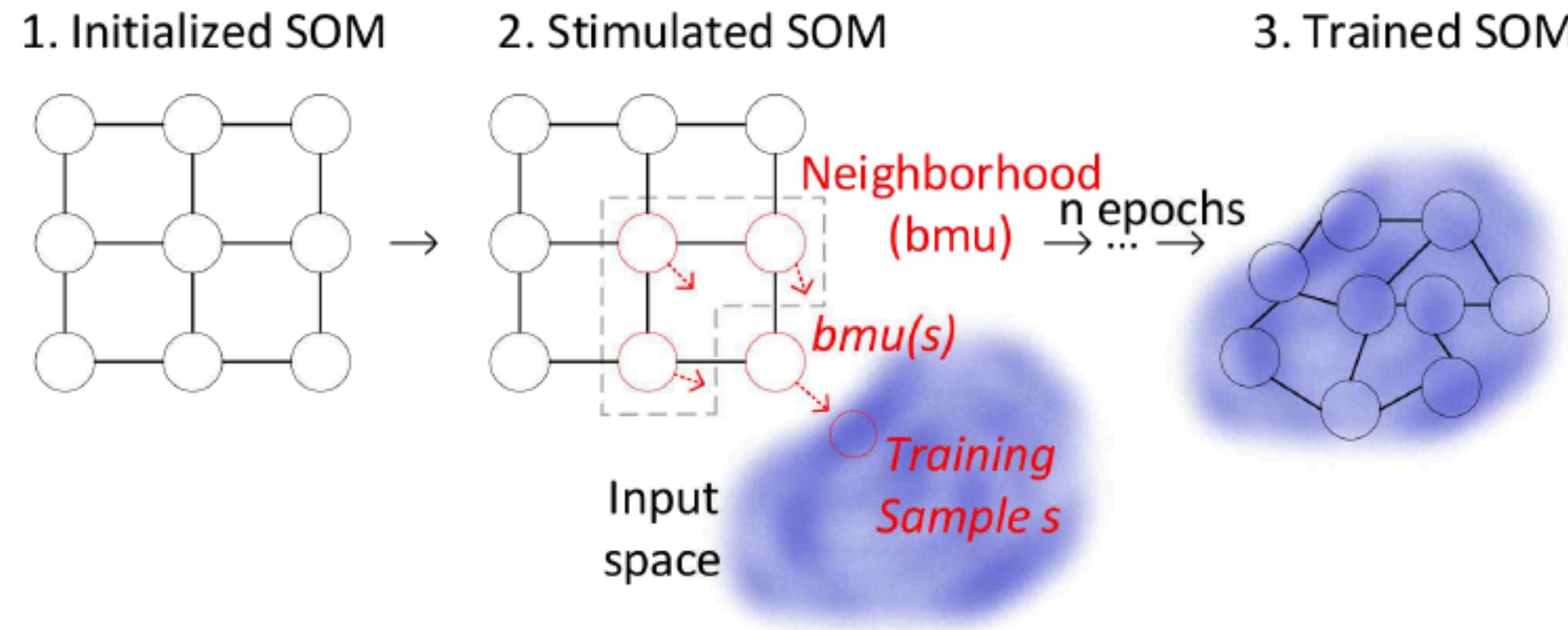
Implémentation sur  
Python pour la détection  
d'anomalies en cyber-  
sécurité

Question 4

- [Friday-WorkingHours-Afternoon-DDos.pcap\\_ISCX](#)
- [Friday-WorkingHours-Afternoon-PortScan.pcap\\_ISCX](#)
- [Friday-WorkingHours-Morning.pcap\\_ISCX](#)
- [Monday-WorkingHours.pcap\\_ISCX](#)
- [Thursday-WorkingHours-Afternoon-Infiltration.pcap\\_ISCX](#)
- [Thursday-WorkingHours-Morning-WebAttacks.pcap\\_ISCX](#)
- [Tuesday-WorkingHours.pcap\\_ISCX](#)
- [Wednesday-workingHours.pcap\\_ISCX](#)

# IMPLEMENTATION D'UNE CARTE DE KOHONEN

16



Hormann, R. and Fischer, E. (2019), Detecting anomalies by using self-organizing maps in industrial environments, pp. 336–344.

# CARTE DE KOHONEN : RÉSULTATS

17

04

## Implémentation d'une carte de Kohonen

Implémentation sur Python pour la détection d'anomalies en cybersécurité

Question 4

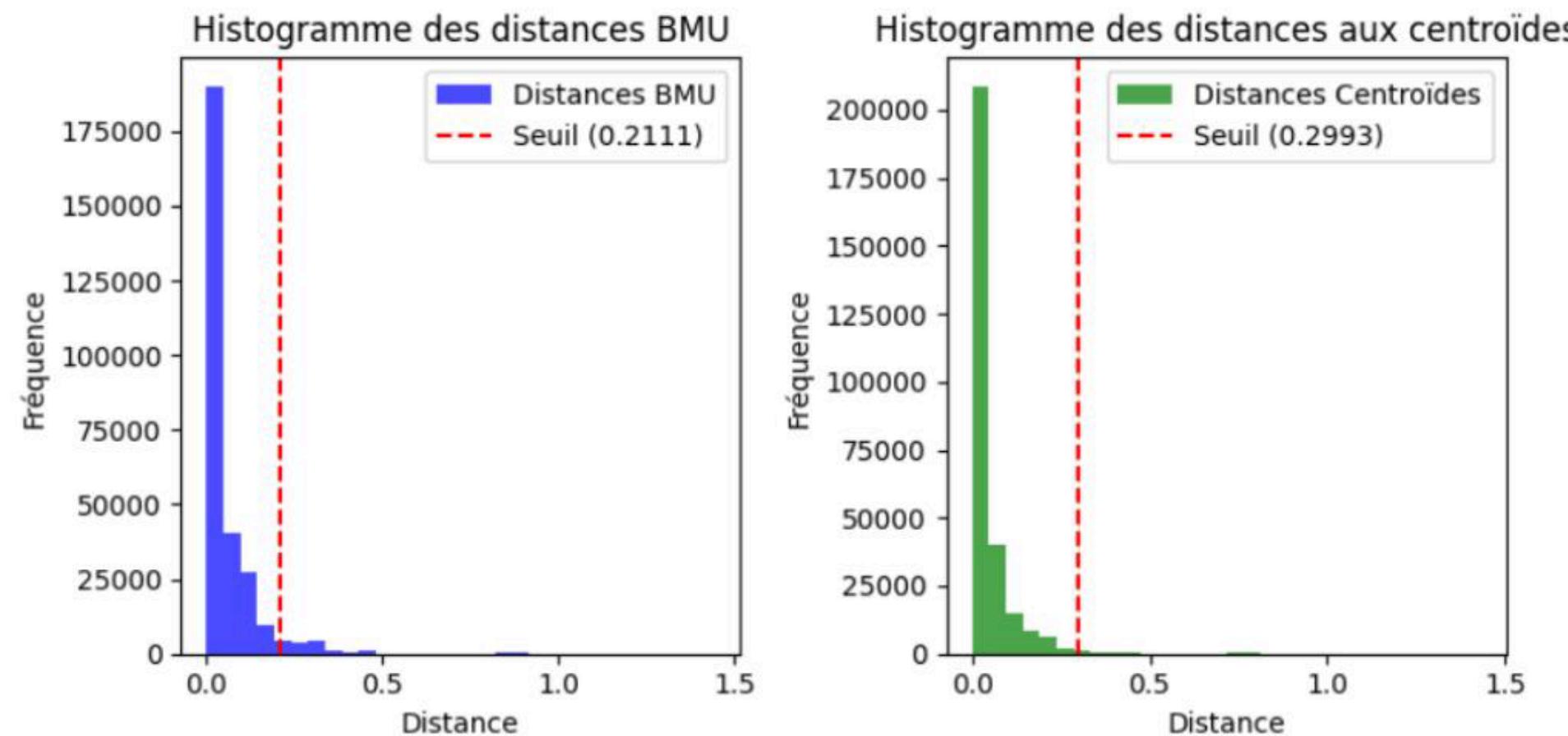


Figure 4.2 : Histogrammes des distances BMU et distances centroïdes

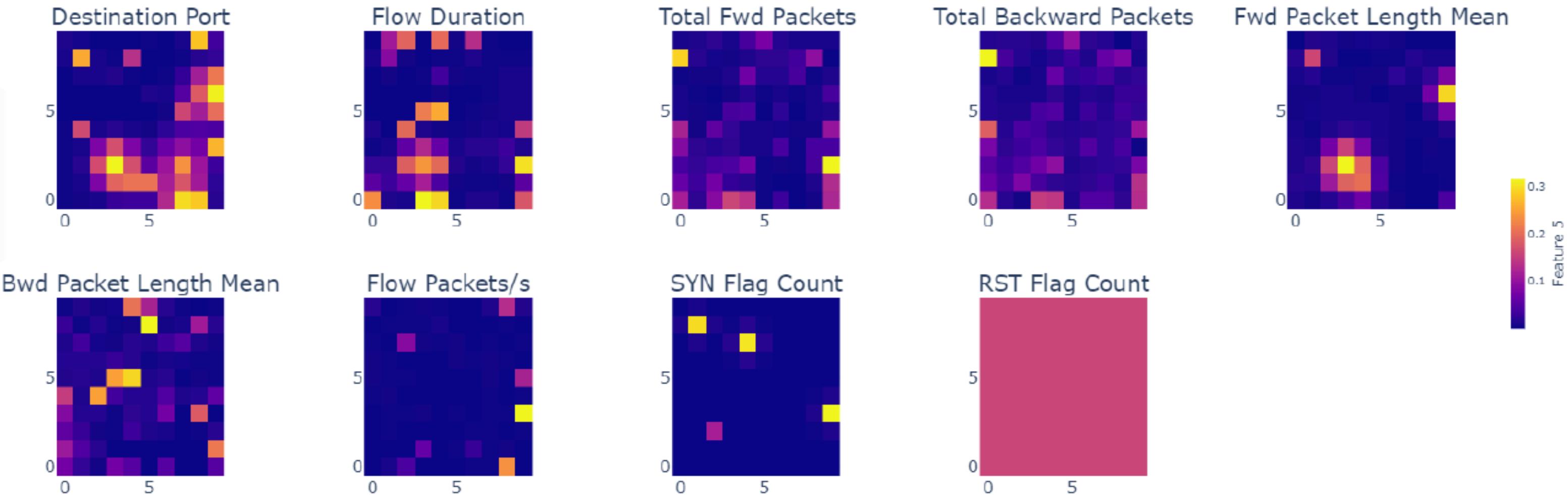
	precision	recall	f1-score	support
Infiltration	Bot	0.00	0.00	0.00
	BruteForce	0.00	0.00	0.00
	DDoS	0.62	0.47	0.54
	DoS	0.82	0.53	0.65
	Infiltration	0.00	0.00	0.00
	Normal	0.91	0.96	0.94
	PortScan	0.65	0.71	0.68
	WebAttack	0.00	0.00	0.00
accuracy			0.88	283077
macro avg	0.38	0.34	0.35	283077
weighted avg	0.87	0.88	0.87	283077

# CARTE DE KOHONEN : RÉSULTATS

**04**  
**Implémentation d'une carte de Kohonen**  
 Implémentation sur Python pour la détection d'anomalies en cybersécurité

Question 4

Codebooks pour chaque Feature (SOM)



# CARTE DE KOHONEN : RÉSULTATS

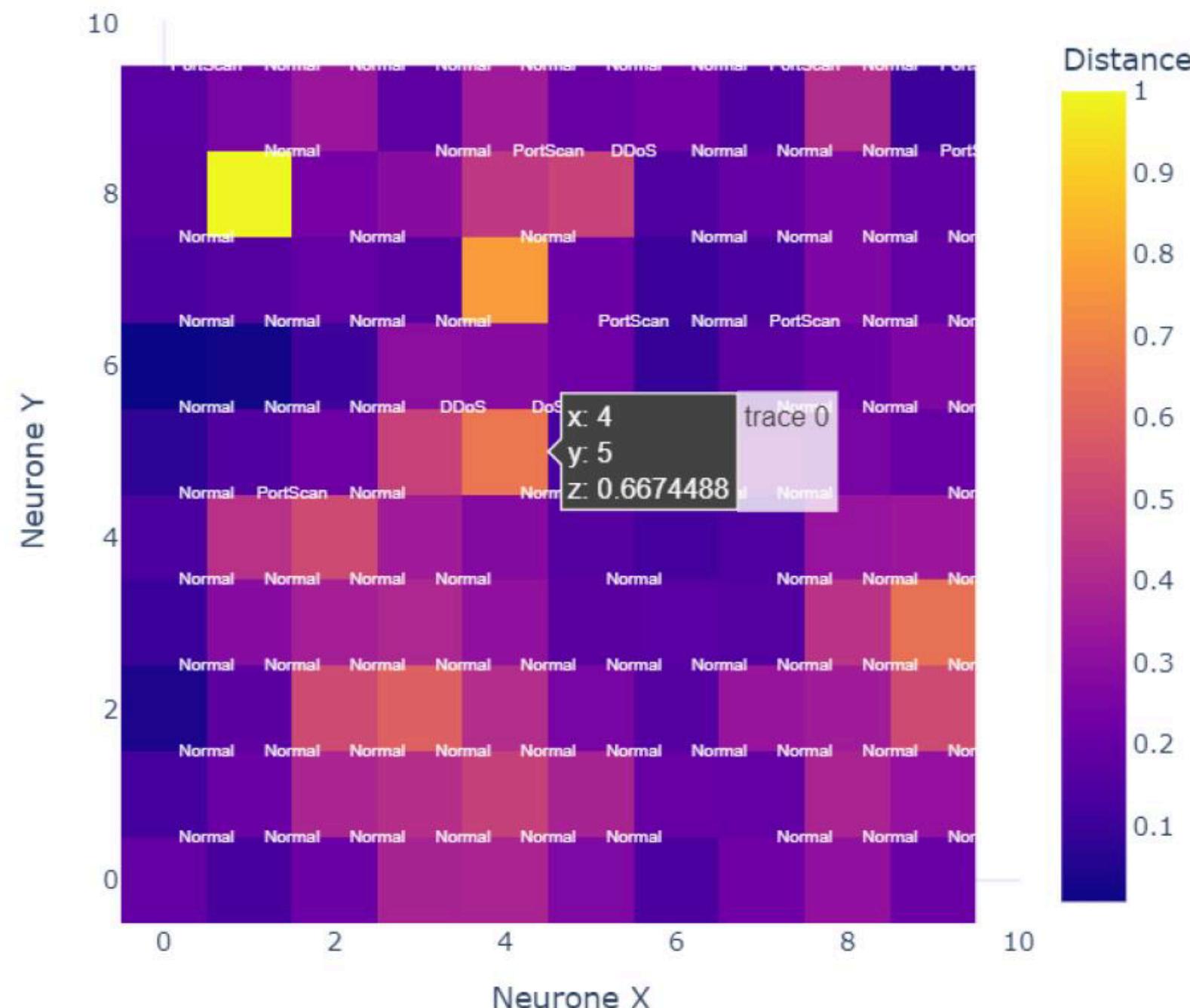
**04**

**Implémentation d'une carte de Kohonen**

Implémentation sur Python pour la détection d'anomalies en cybersécurité

Question 4

U-Matrix avec superposition des classes dominantes



# IMPLÉMENTATION D'UN DBN

*pour la détection d'anomalies*

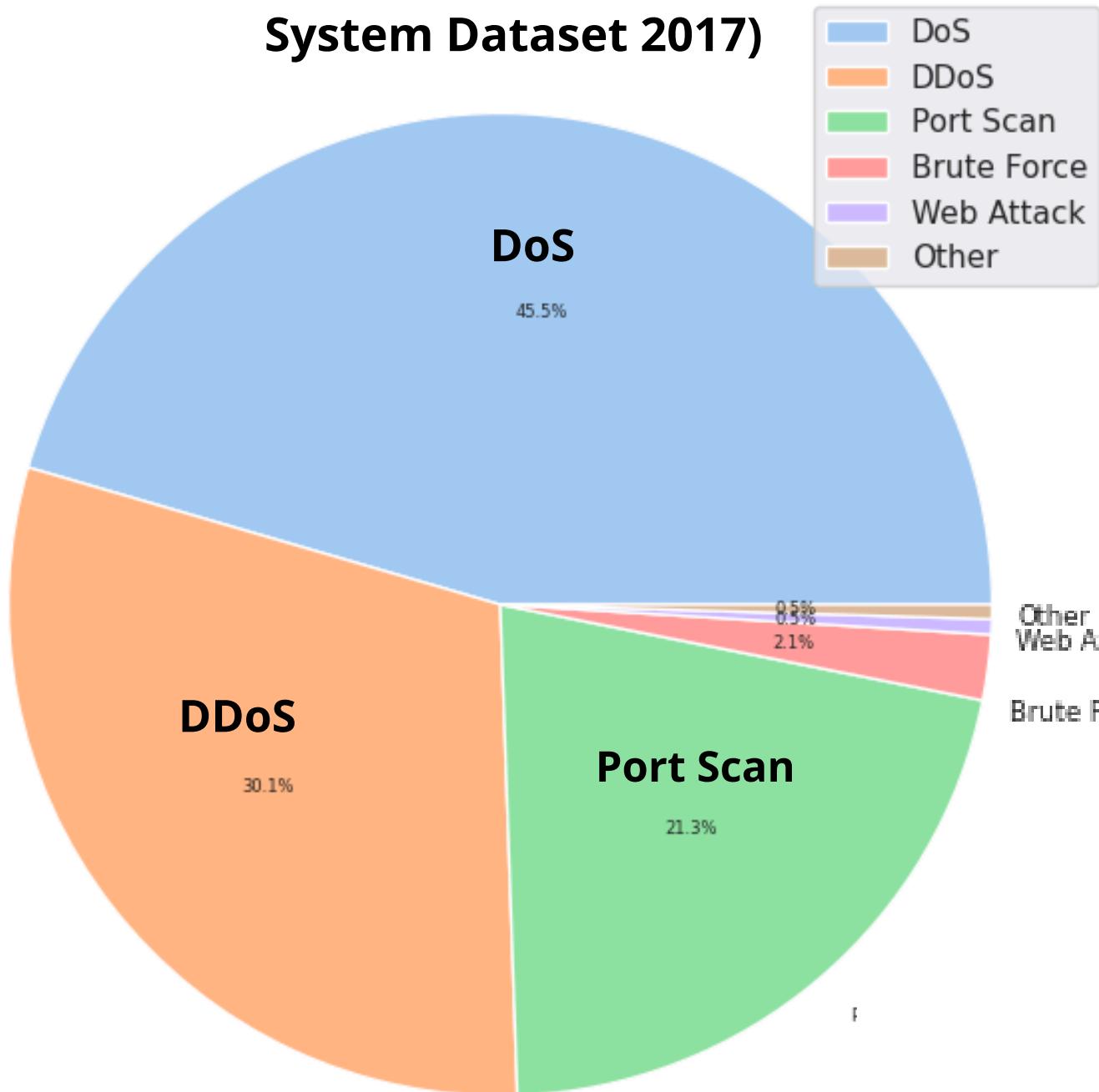
05

## Implémentation d'un DBN

Implémentation sur Python pour la détection d'anomalies en cyber-sécurité

*Question 6*

**CICIDS2017 (Canadian Institute for  
Cybersecurity Intrusion Detection  
System Dataset 2017)**



- [Friday-WorkingHours-Afternoon-DDos.pcap\\_ISCX](#)
- [Friday-WorkingHours-Afternoon-PortScan.pcap\\_ISCX](#)
- [Friday-WorkingHours-Morning.pcap\\_ISCX](#)
- [Monday-WorkingHours.pcap\\_ISCX](#)
- [Thursday-WorkingHours-Afternoon-Infiltration.pcap\\_ISCX](#)
- [Thursday-WorkingHours-Morning-WebAttacks.pcap\\_ISCX](#)
- [Tuesday-WorkingHours.pcap\\_ISCX](#)
- [Wednesday-workingHours.pcap\\_ISCX](#)

# ÉTHIQUE

21



Souveraineté  
**des données**

Impact sur la  
**Société**

**Énergie**  
et durabilité

MERCI POUR VOTRE ATTENTION

*Des questions ?*

