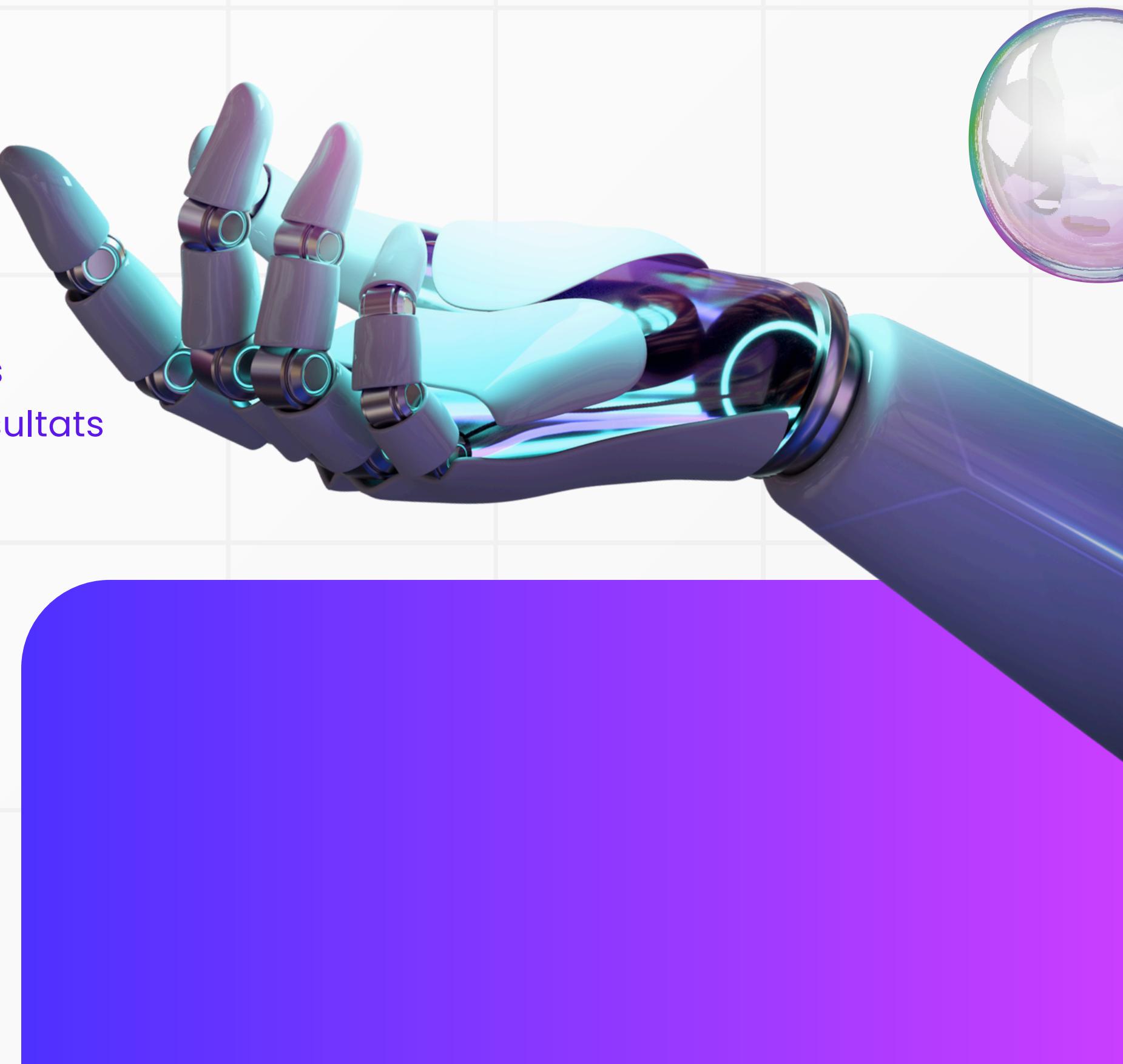


# ZOIDBERG

T-DEV-810

# Nos missions

- Utiliser une procédure train-validation-test
- Utiliser une procédure de validation croisée
- Comparer nos résultats
- Utiliser les jeux de données pour ajuster nos algorithmes
- Explorer et tester diverses méthodes et comparer les résultats



# Nos objectifs

- Solution permettant un premier tri des patient en fonction du pré-diagnostique
- Fournir un outil fiable qui minimiserais les faux négatifs.
- Permettre la prise en charge des cas urgents plus rapidement
- Augmentation de l'efficience et diminution des besoins en ressources humaines.

# Pré-traitement des Données

## Exploration des données

- Images jpeg en nuances de gris
- Différentes tailles d'images
- Train: 1342 normales(25,78%), 3875 pneumonia (74,28%), (Total: 5217)
- Test: 234 normales (37,5%), 390 pneumonia (62,5%), (Total: 624)
- Val: 8 normales (50%), 8 pneumonia (50%)
- 2 classes possibles (Normal, Pneumonia)

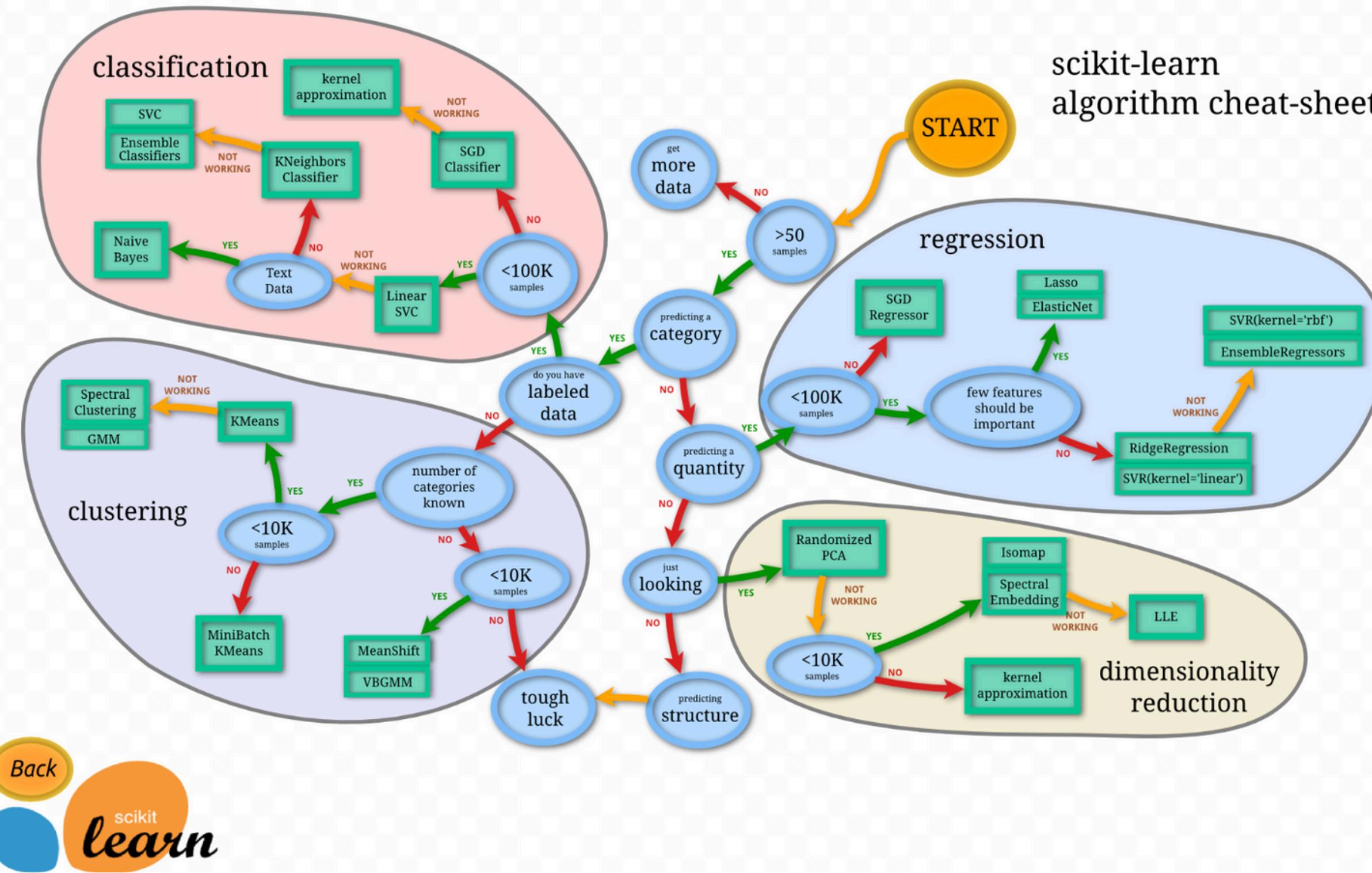
# Pré-traitement des Données

## Nettoyage, Normalisation et redimensionnement des données

- Equilibrage des classes
- Redimensionnement des images à une taille standardisée (224 x 224)
- Conversion des images en matrices
- Normalisation des valeurs de pixels (0-255 à 0-1)

# choix du modèle

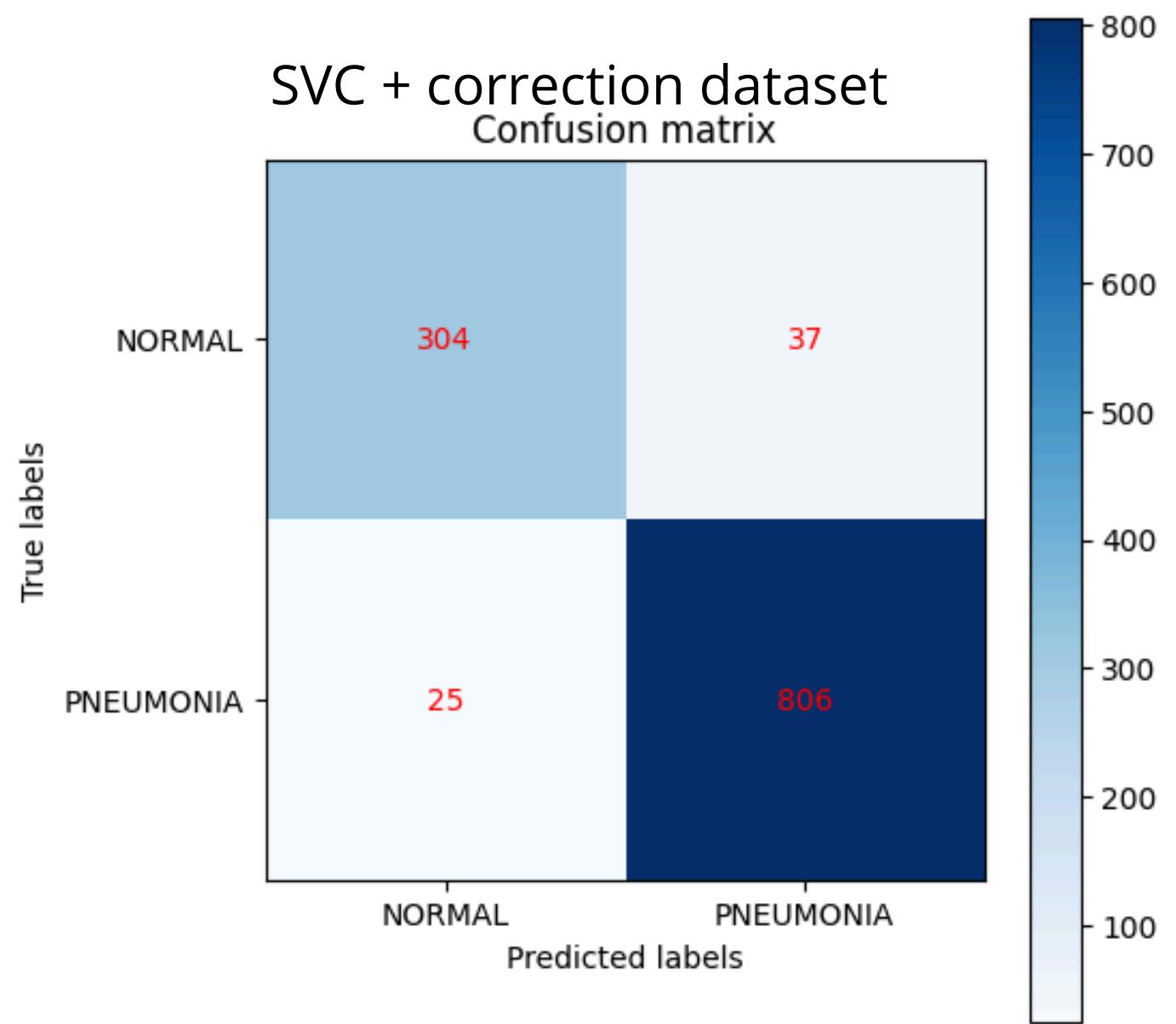
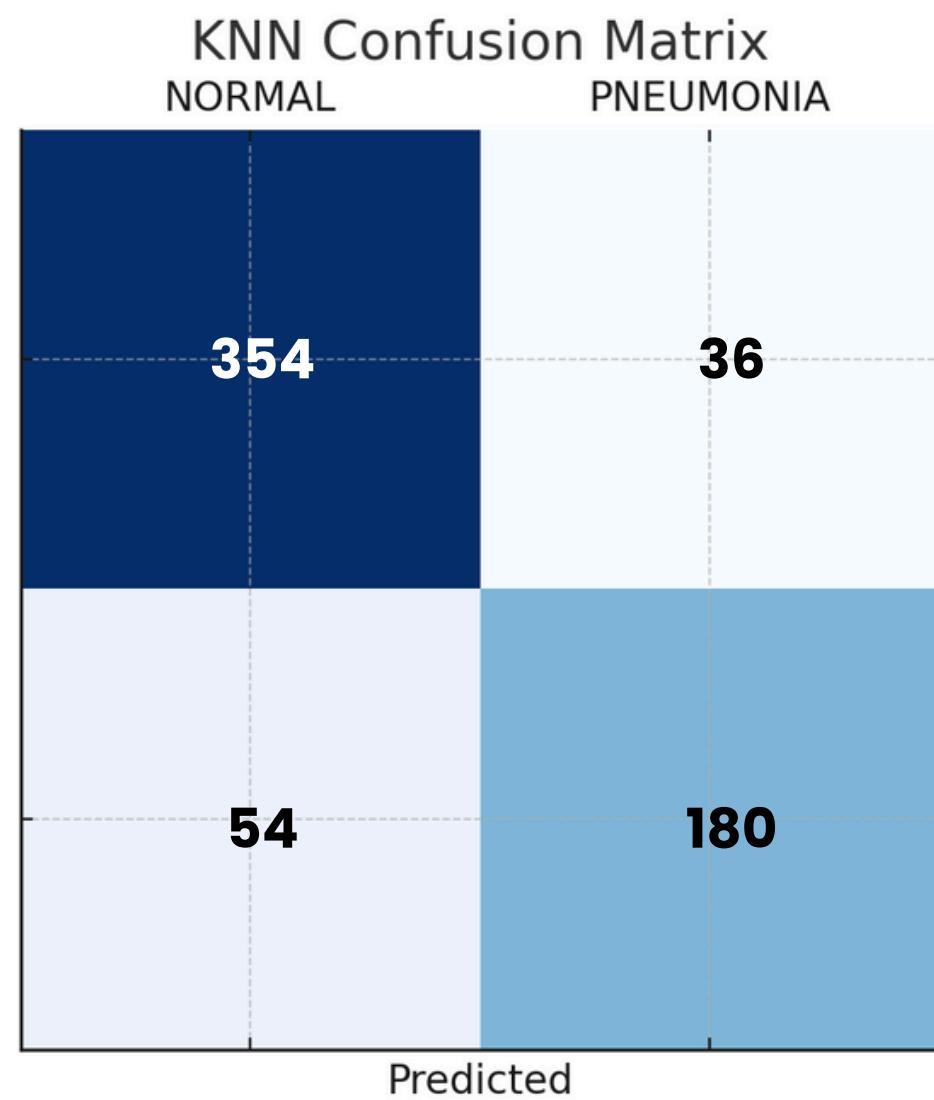
scikit-learn  
algorithm cheat-sheet



Back

scikit  
learn

# *comparaison des matrices de confusions*



# résultat

Ancien modèle

```
Train recall: 0.9201651754989677  
Test recall: 0.8413461538461539
```

Nouveau modèle

```
Train recall: 0.9854735618826264  
Test recall: 0.9699157641395909
```

# *prédiction*

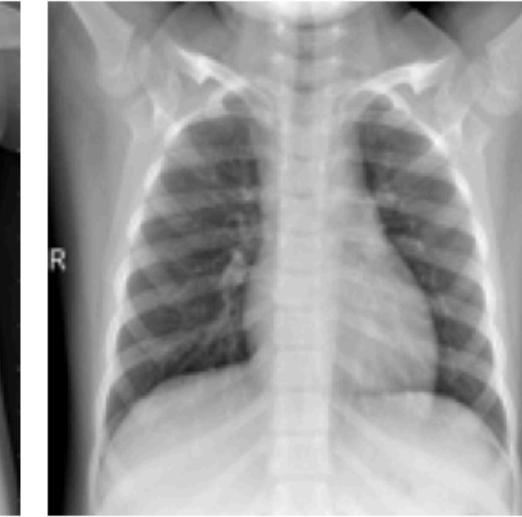
True: PNEUMONIA  
Predicted: PNEUMONIA



True: PNEUMONIA  
Predicted: PNEUMONIA



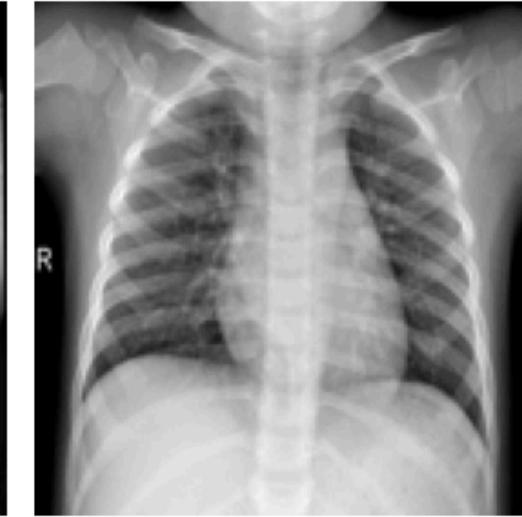
True: NORMAL  
Predicted: NORMAL



True: NORMAL  
Predicted: NORMAL



True: NORMAL  
Predicted: NORMAL



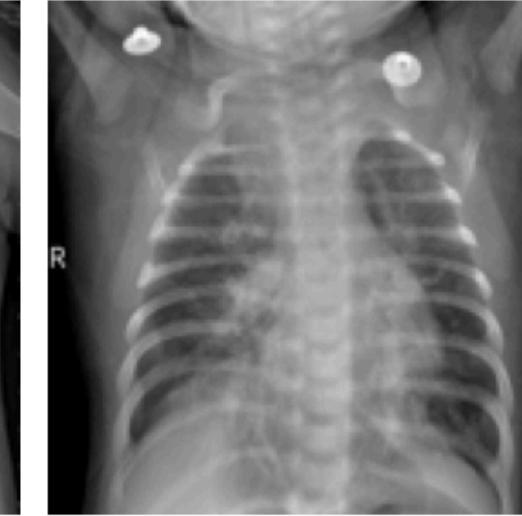
True: NORMAL  
Predicted: PNEUMONIA



True: PNEUMONIA  
Predicted: NORMAL



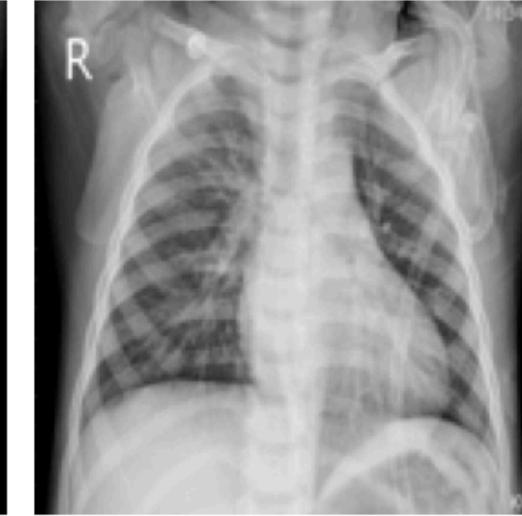
True: PNEUMONIA  
Predicted: NORMAL



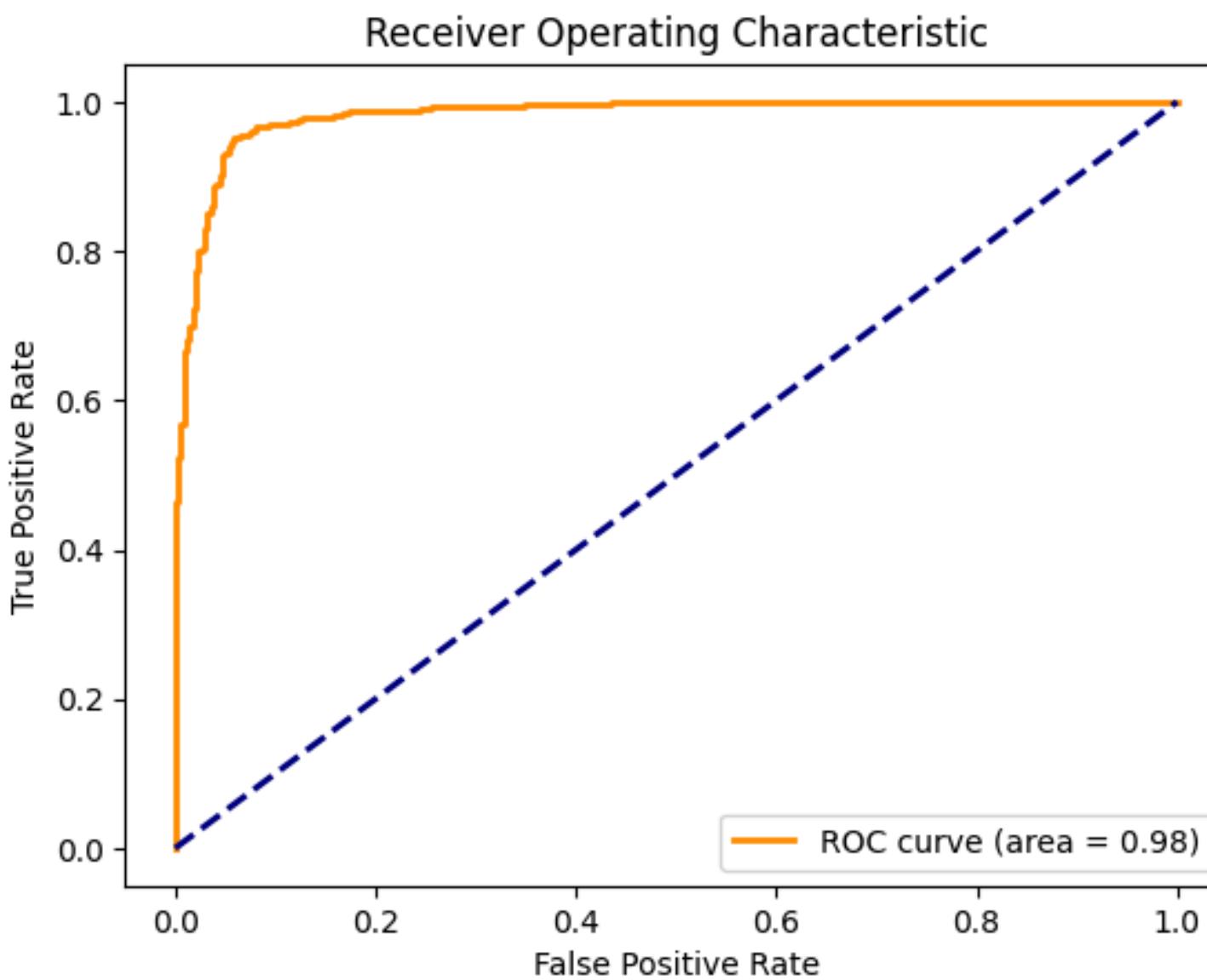
True: PNEUMONIA  
Predicted: NORMAL



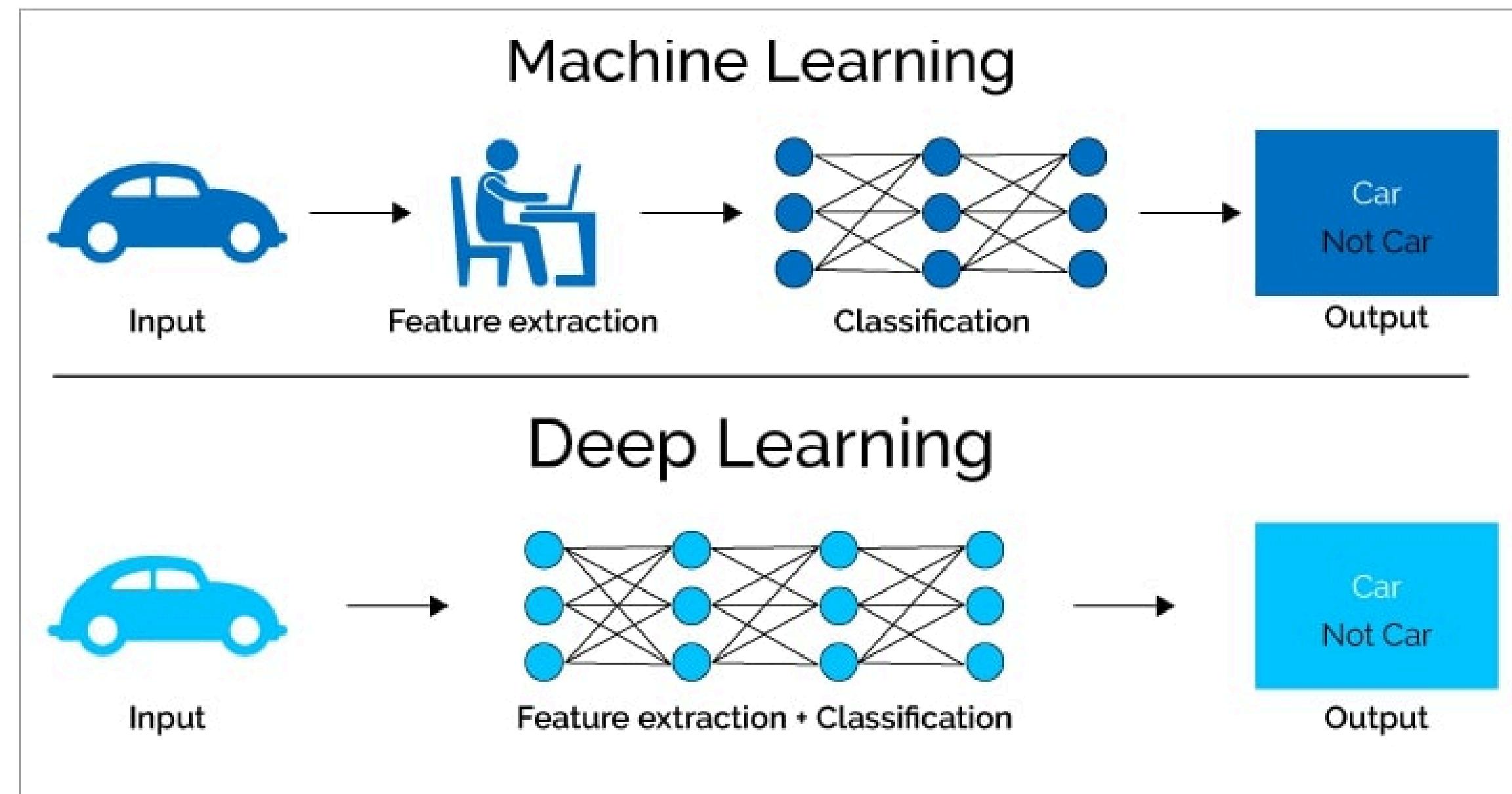
True: PNEUMONIA  
Predicted: NORMAL



# ROC CURVE



# Machine vs Deep Learning



# Deep Learning

## Selection du modèle

### CNN

- Efficace pour le traitement d'images.
- Capture les caractéristiques spatiales.
- Réduction du nombre de paramètres
- Beaucoup de données nécessaires
- Ressources computationnelles importantes

### RNN

- Efficace pour les données séquentielles
- Capable de conserver une mémoire des états précédents
- Moins performant pour le traitement d'images.

### DNN

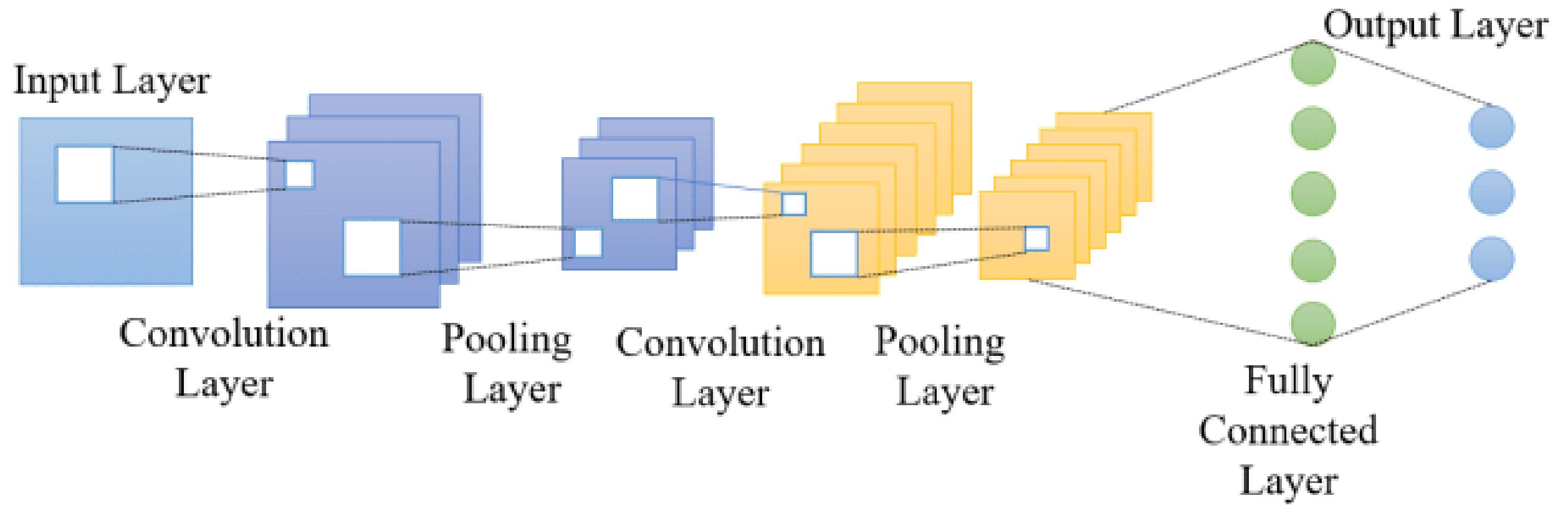
- Flexible, applicable à divers types de données.
- Bonne capacité à apprendre des représentations complexes.
- Risque de surapprentissage si les données sont insuffisantes.
- Ressources computationnelles importantes

# Deep Learning

## Selection du modèle : CNN

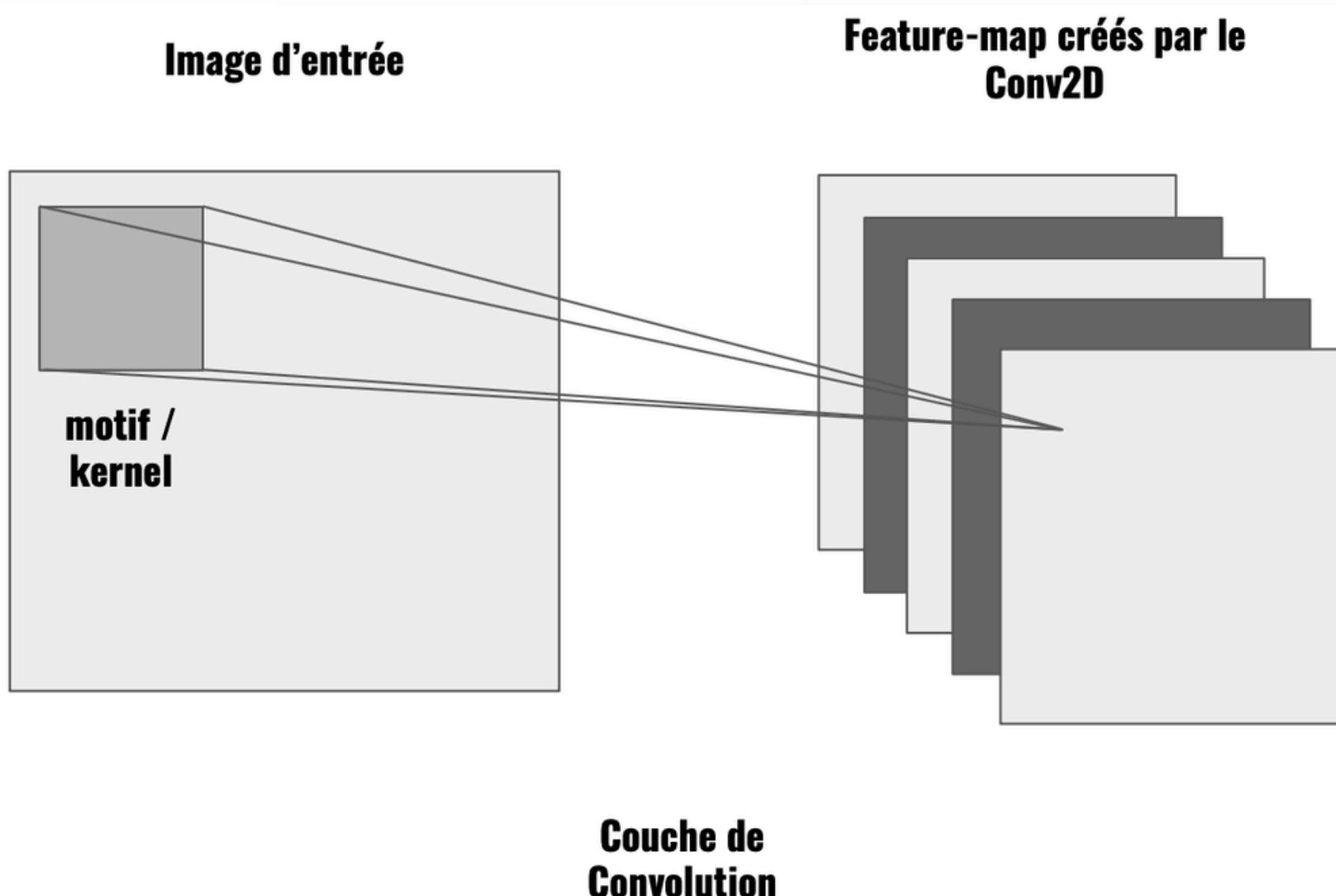
Architechture:

- Couches de convolution (détection des motifs locaux des images)
- Couches de pooling (reduction de la dimentialité: - charge computationnelle, - sur-apprentissage)
- Couche dense (Combinaison des caractéristiques extraites pour la classification finale)



# Deep Learning

## CNN : Couche de convolution



```
28 def convolve2d(image, kernel, stride=1, padding=0):
29     if padding > 0:
30         image = np.pad(image, [(padding, padding), (padding, padding), (0, 0)], mode='constant')
31
32     kernel_height, kernel_width, input_channels, num_kernels = kernel.shape
33     image_height, image_width, _ = image.shape
34
35     output_height = (image_height - kernel_height) // stride + 1
36     output_width = (image_width - kernel_width) // stride + 1
37
38     output = np.zeros((output_height, output_width, num_kernels))
39
40     for k in range(num_kernels):
41         for y in range(0, image_height - kernel_height + 1, stride):
42             for x in range(0, image_width - kernel_width + 1, stride):
43                 region = image[y:y + kernel_height, x:x + kernel_width, :]
44                 output[y // stride, x // stride, k] = np.sum(region * kernel[:, :, :, k]) + kernel[:, :, :, k].sum()
45
46     return output
```

# Deep Learning

## CNN : Couche de pooling

Carte de caractéristiques

1	4	2	7
2	6	8	5
3	4	0	7
1	2	3	1

Max pooling avec filtre 2x2 et stride de 2

$$\text{Max}(3, 4, 1, 2) = 4$$

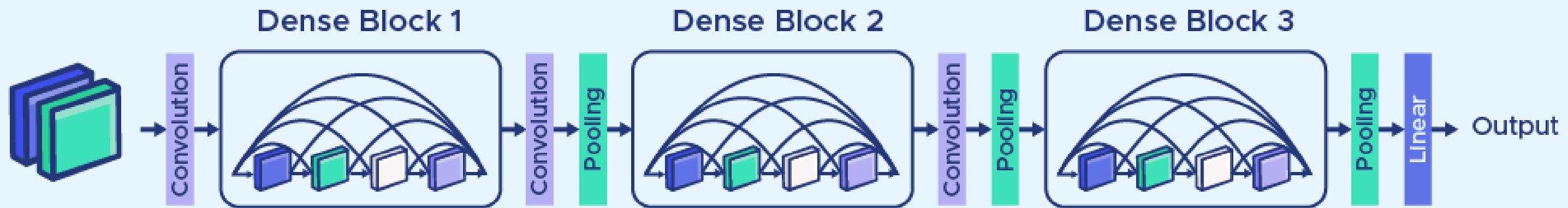
Carte de caractéristiques sous-échantillonnée

6	8
4	7

```
51 def max_pooling(image, size=2, stride=2):
52     output_height = (image.shape[0] - size) // stride + 1
53     output_width = (image.shape[1] - size) // stride + 1
54     output = np.zeros((output_height, output_width, image.shape[2]))
55
56     for y in range(0, image.shape[0] - size + 1, stride):
57         for x in range(0, image.shape[1] - size + 1, stride):
58             for c in range(image.shape[2]):
59                 region = image[y:y + size, x:x + size, c]
60                 output[y // stride, x // stride, c] = np.max(region)
61
62     return output
```

# Deep Learning

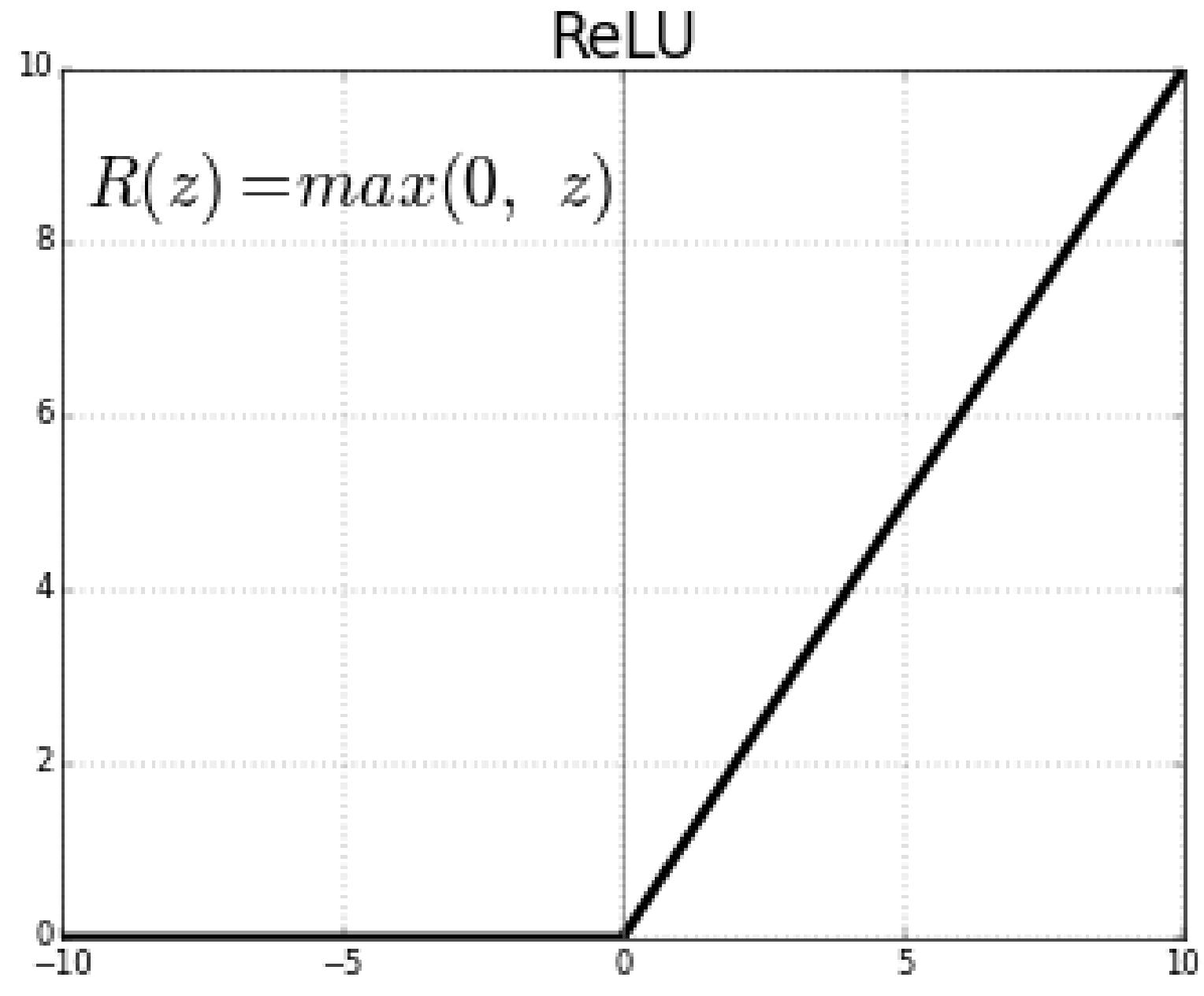
CNN : Couche dense ou fully connected



```
64 def dense(x, weights, bias):  
65     return np.dot(x, weights) + bias
```

# Deep Learning

## CNN : Fonctions d'activation : ReLU

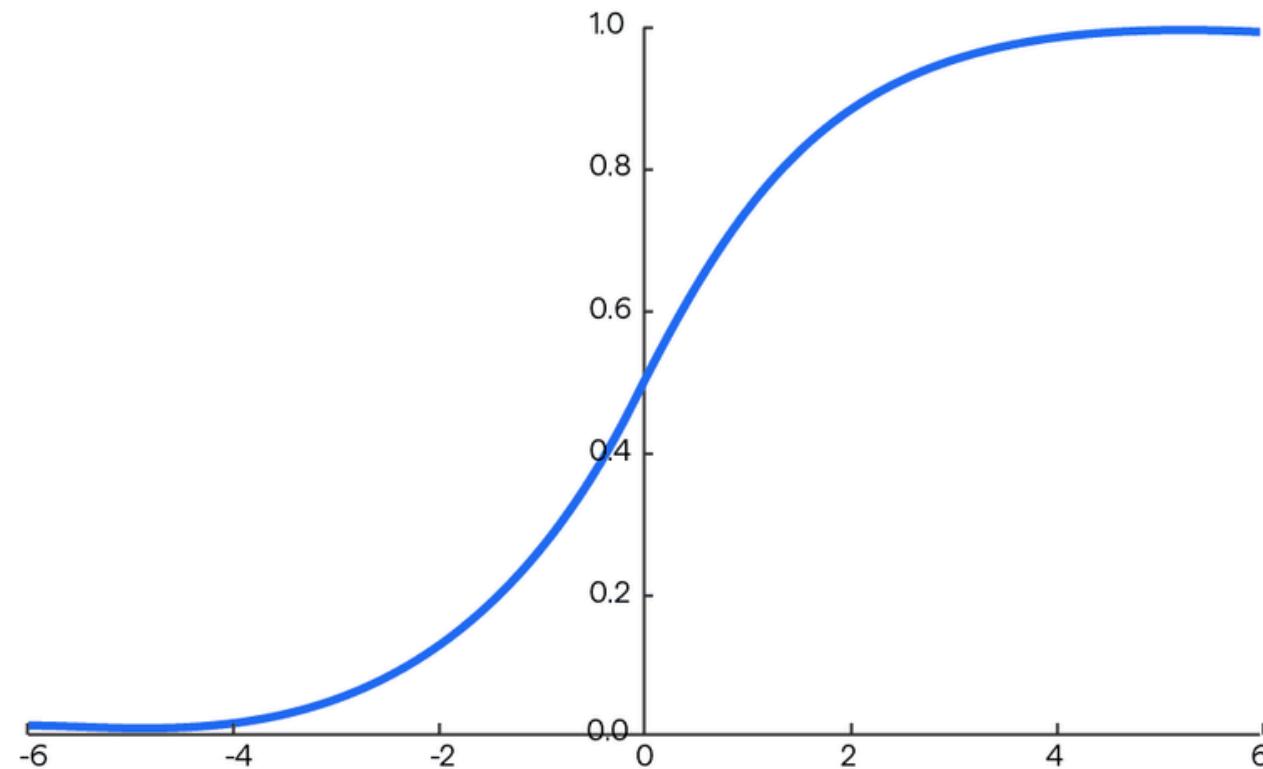


```
48 ✓ def relu(x):  
49 |     return np.maximum(0, x)
```

# Deep Learning

## CNN : Fonctions d'activation : softmax

### Softmax Function



```
104 ✓ def softmax(x):  
105     e_x = np.exp(x - np.max(x))  
106     return e_x / e_x.sum(axis=0)
```

# Deep Learning

## Selection des hyperparamètres

**Grid Search :** Explore toutes les combinaisons possibles d'hyperparamètres dans la grille donnée.  
Peut être très coûteux en termes de temps et de ressources

**Random Search :** Explore des combinaisons de manière aléatoire dans la grille donnée.  
Moins coûteux en termes de temps et de calcul .

# Deep Learning

## Selection de la métrique

**Accuracy**: Le taux de classification correcte parmi toutes les prédictions.

**Recall**: La proportion de vrais positifs détectés parmi toutes les instances qui sont réellement positives.

**Precision** : La proportion de prédictions positives correctes parmi toutes les prédictions positives faites par le modèle.

**F1 Score** : Une mesure combinée de précision et de rappel, calculée comme la moyenne harmonique des deux.

# Deep Learning

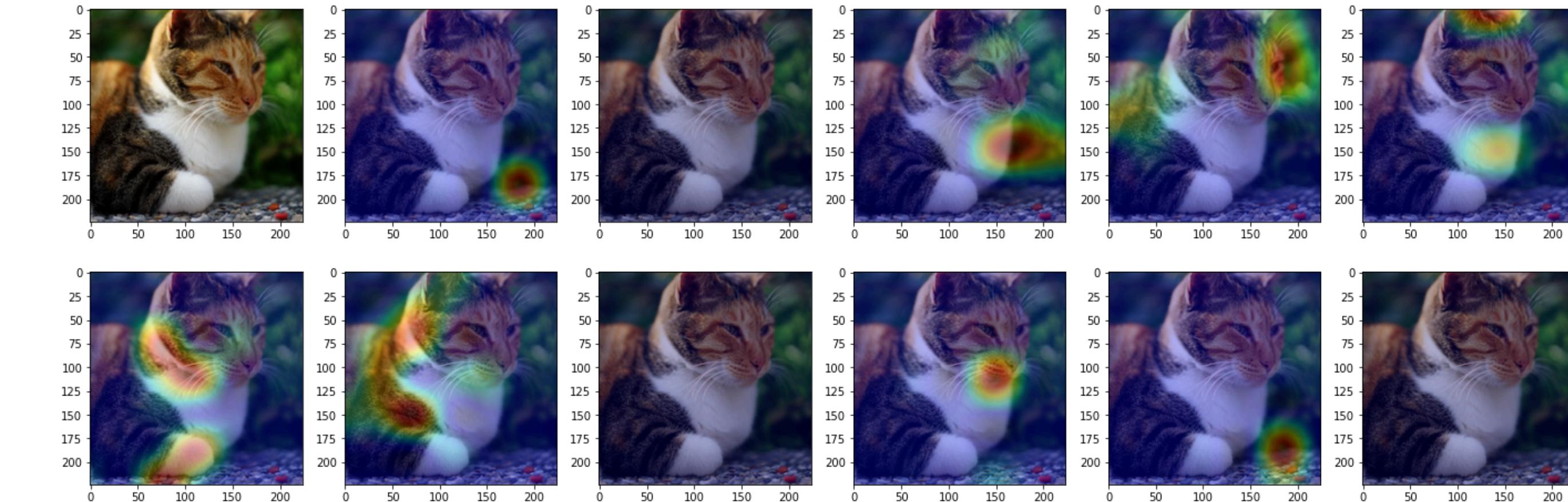
## Validation du modèle : **Cross-Validation**

- Calcul de la performance moyenne du modèle sur toutes les itérations.
- Évaluation de la variabilité de la performance du modèle.

**avantages** : Evaluation plus fiable de la performance du modèle.

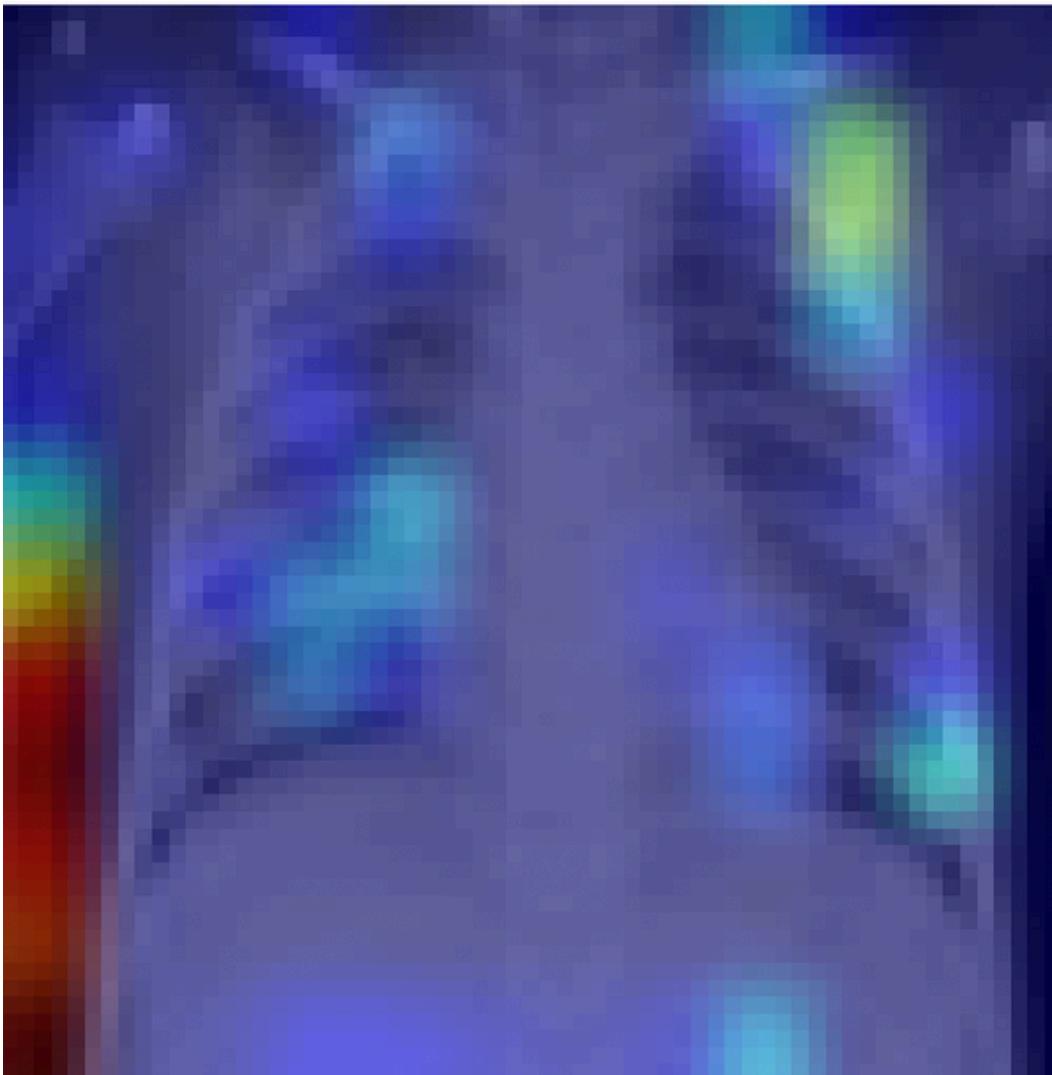
**Inconvénients** : Coûteux en termes de calcul pour des grands ensembles de données

# heatmap



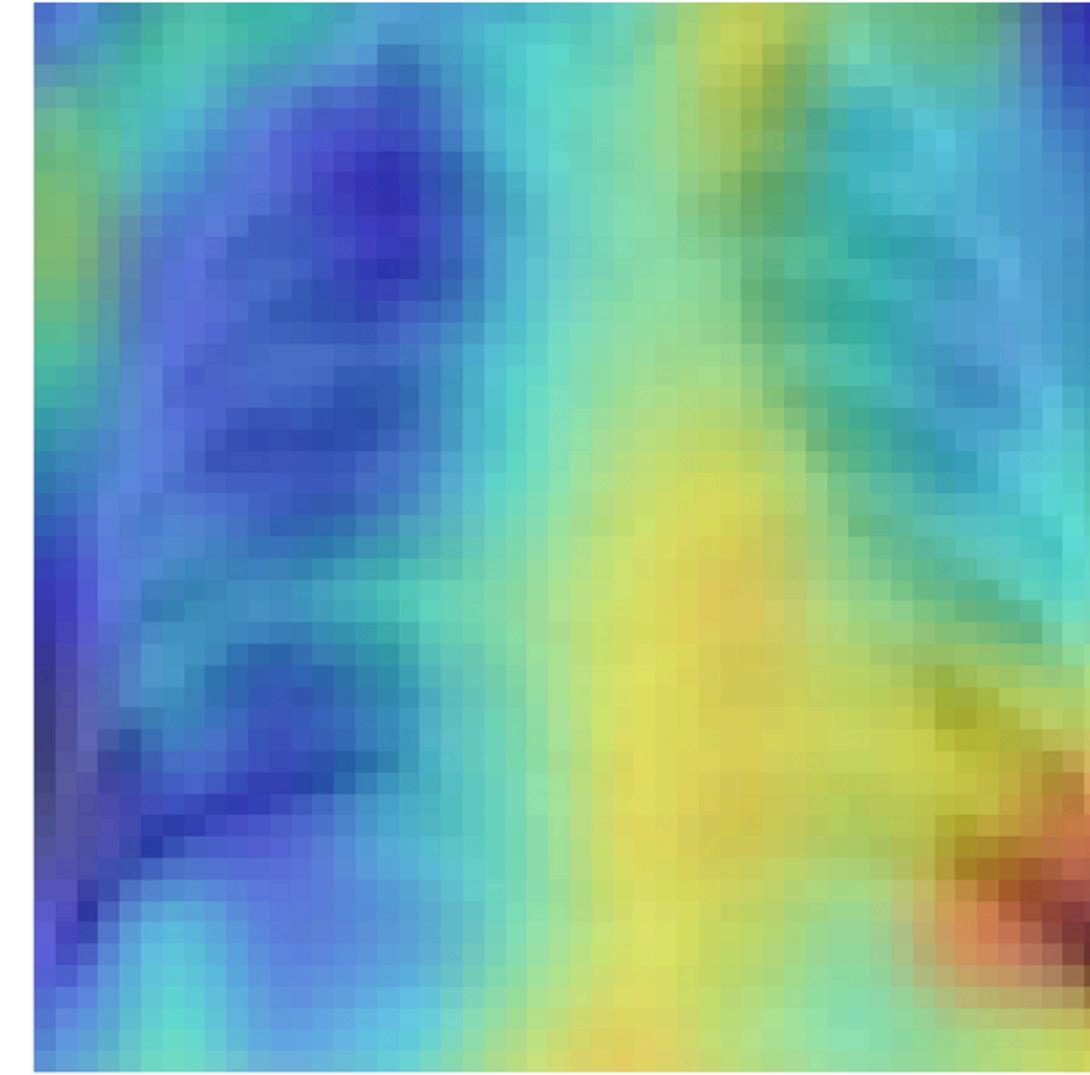
# *croping*

Sans croping



Class: Pneumonia  
Predicted class: PNEUMONIA  
Confidence: [0.9985572]

Avec croping



Class: Pneumonia  
Predicted class: PNEUMONIA  
Confidence: [0.99553156]

# *modèle convolutif*

Choix de VGG16 pour le transfert d'apprentissage

- Modèle pré-entraîné réputé
- Possède des filtres capables d'extraire des caractéristiques pertinentes à partir des images

Choix du modèle convolutif (CNN)

- Adapté aux données d'images
- Performances supérieures dans de nombreuses tâches de vision par ordinateur

# *modèle convolutif*

## *etude comparative : metrics & optimizer*

- *deux jeux de données [petit|gros]*
- *deux méthodes [adam vs rmsprop]*
- *deux metrics [recall vs recall + accuracy]*

Test 1 : - metrics : recall + acc - optimizer : adam

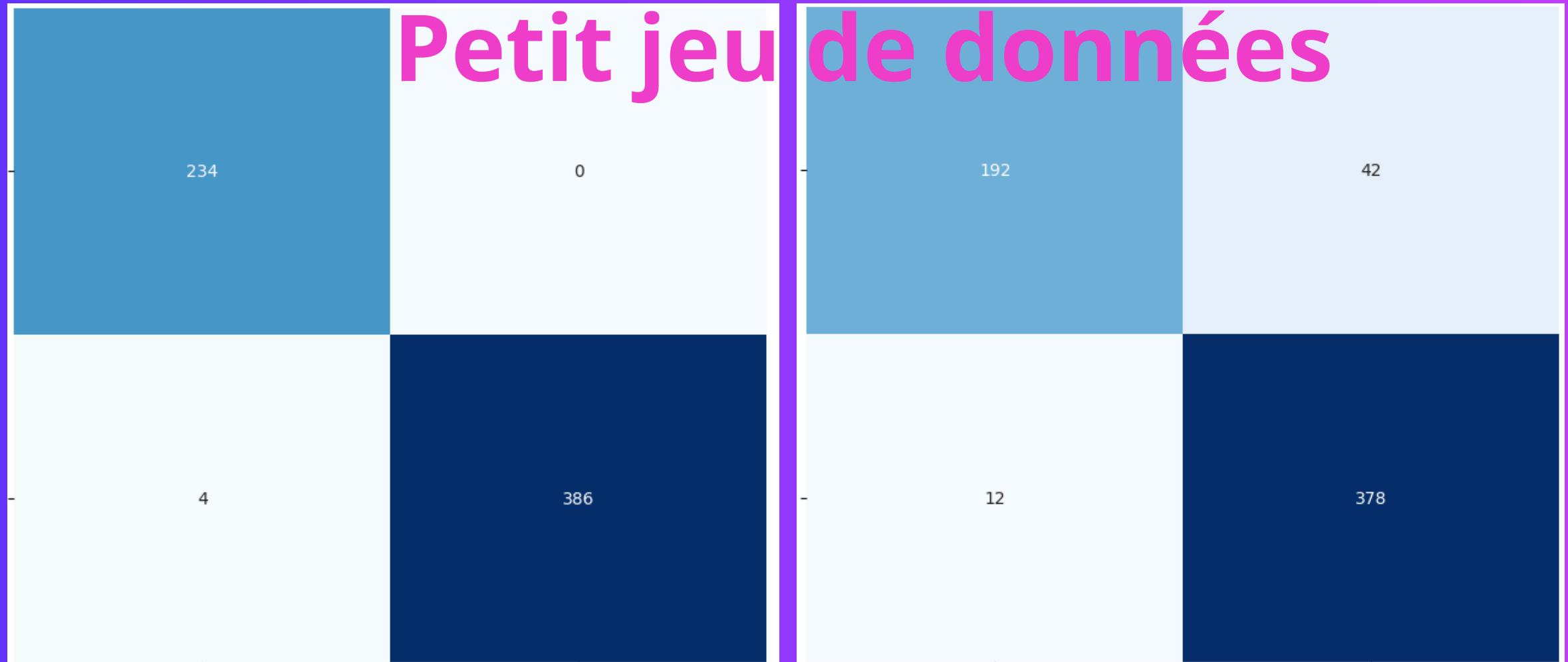
Test 2 : - metrics : recall - optimizer : adam

Test 3 : - metrics : recall + acc - optimizer : rmsprop

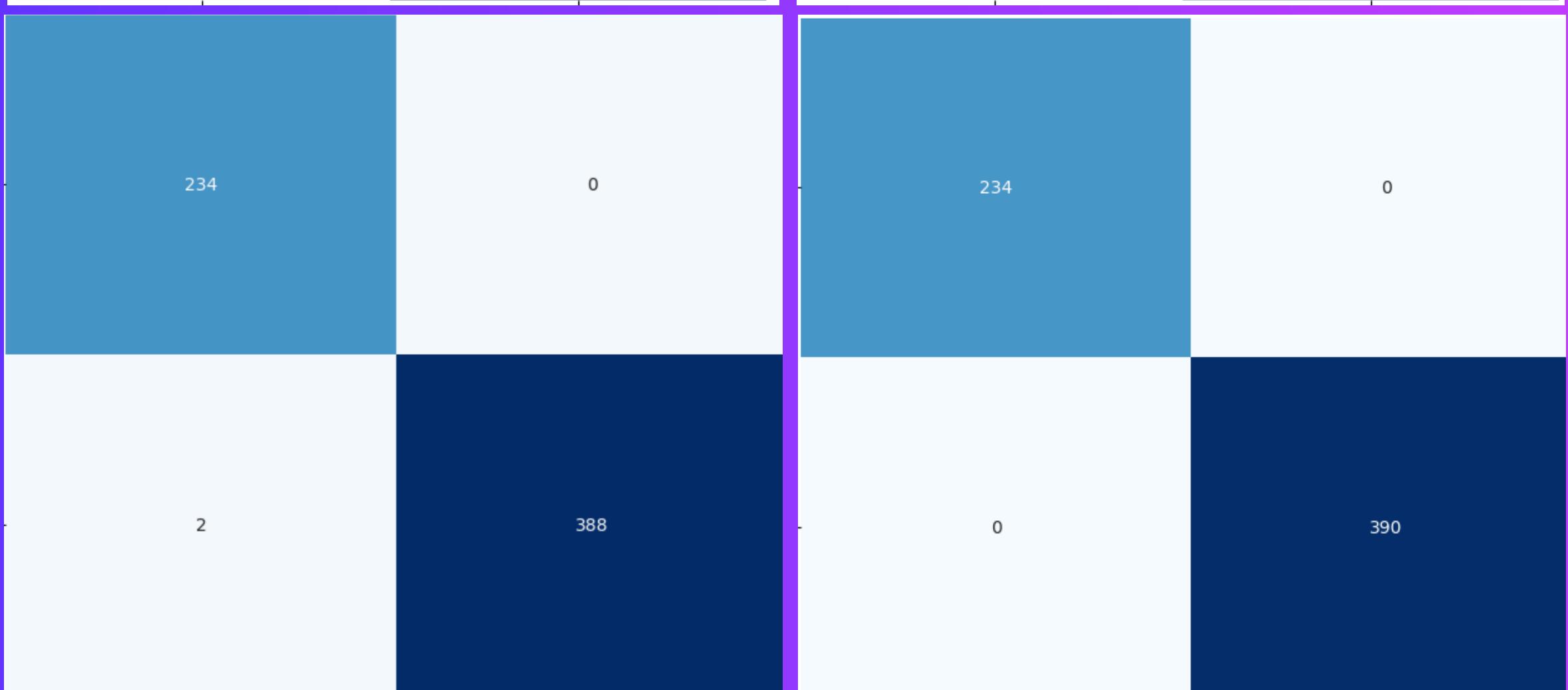
Test 4 : - metrics : recall - optimizer : rmsprop

# Petit jeu de données

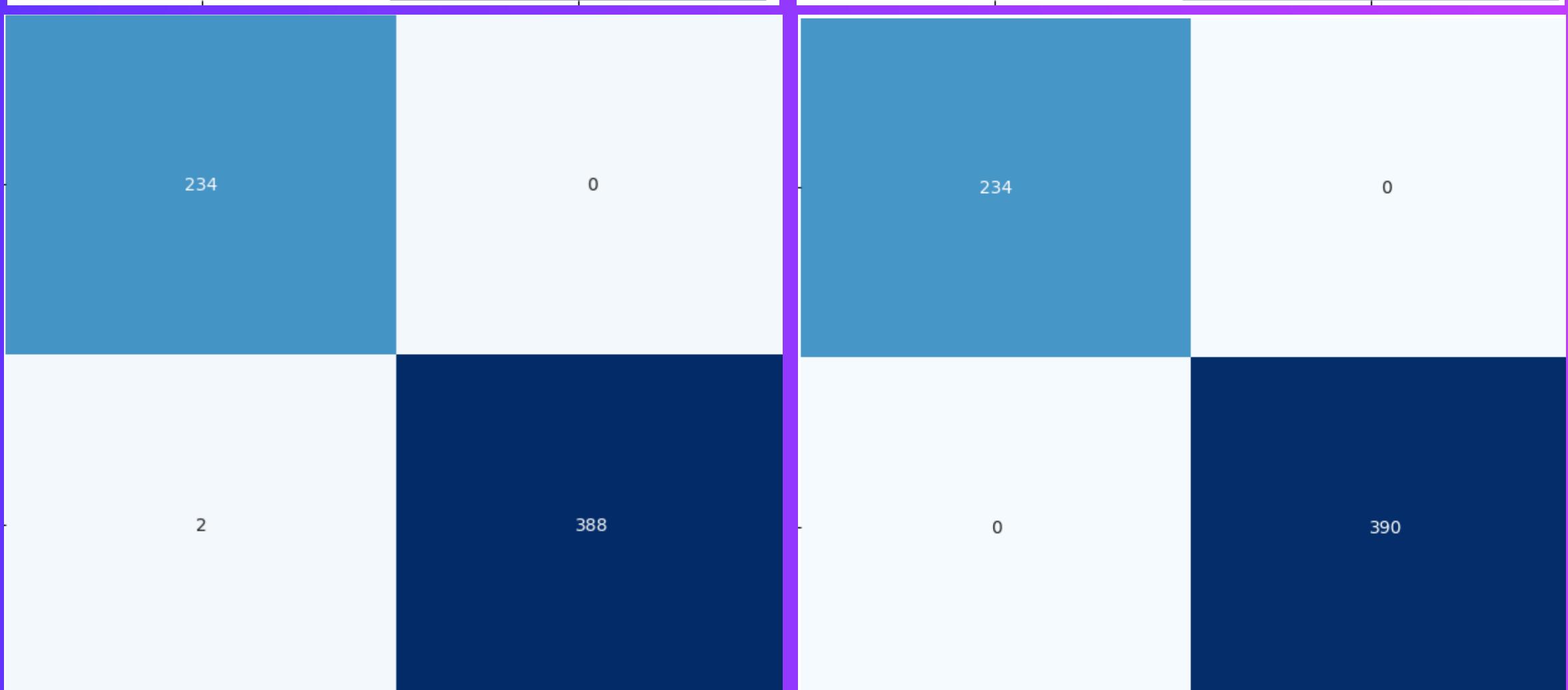
***test 1 :***  
***adam***  
***recall +***  
***accuracy***



***test 2 :***  
***adam***  
***recall***



***test 4 :***  
***rmsprop***  
***recall +***  
***accuracy***

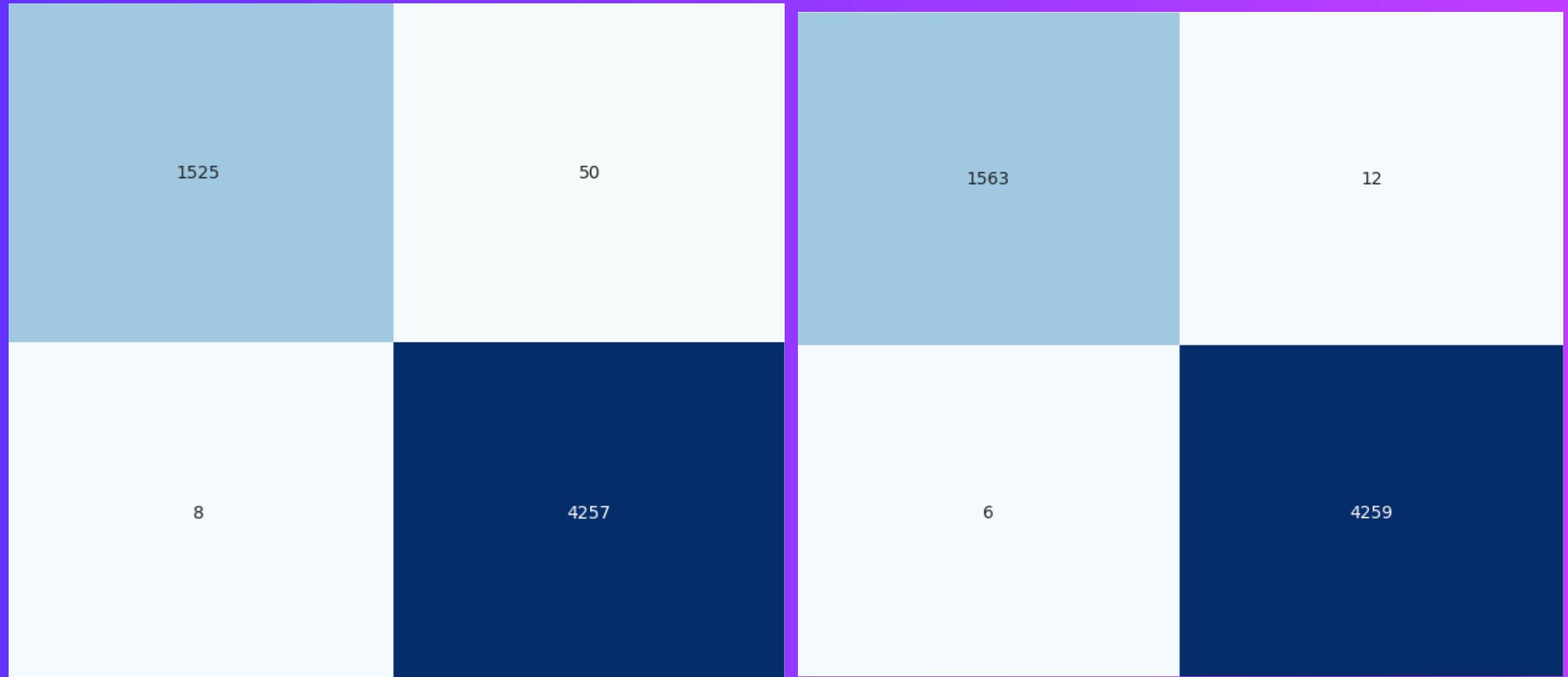


# Gros jeu de données

***test 1 :***  
***adam***  
***recall +***  
***accuracy***



***test 2 :***  
***adam***  
***recall***

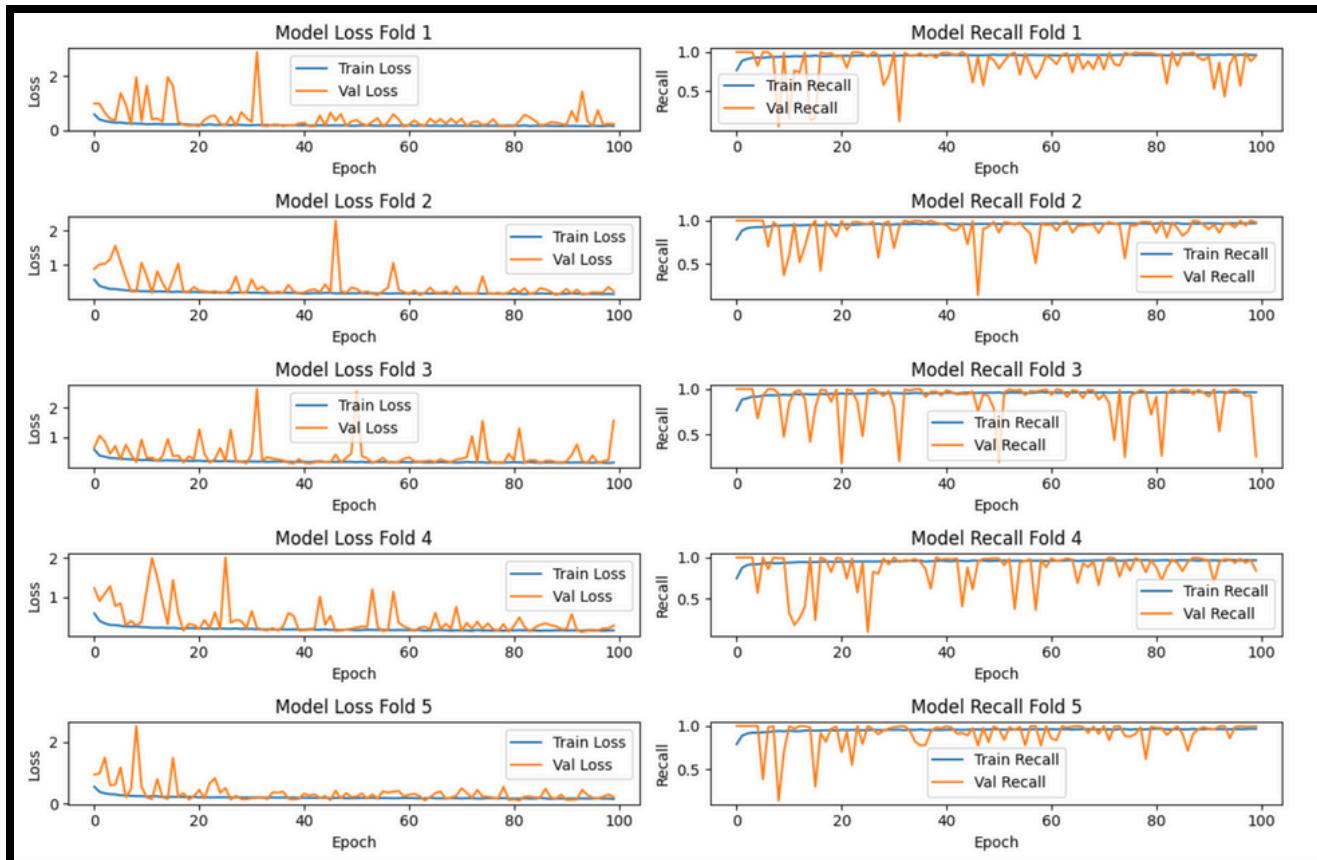


***test 4 :***  
***rmsprop***  
***recall +***  
***accuracy***



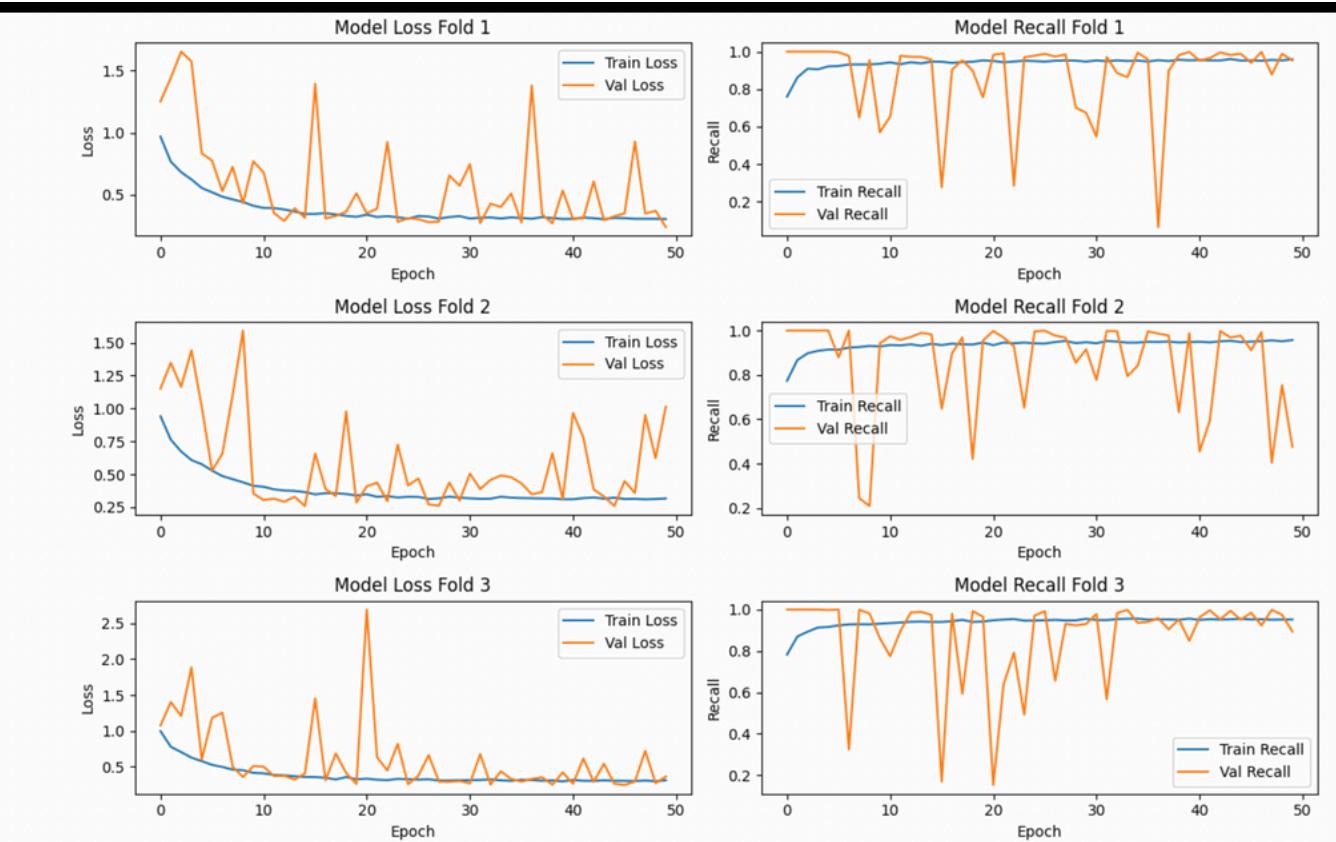
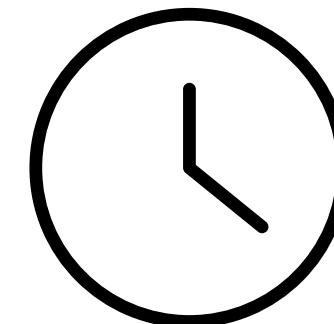
OVERFITTUNG



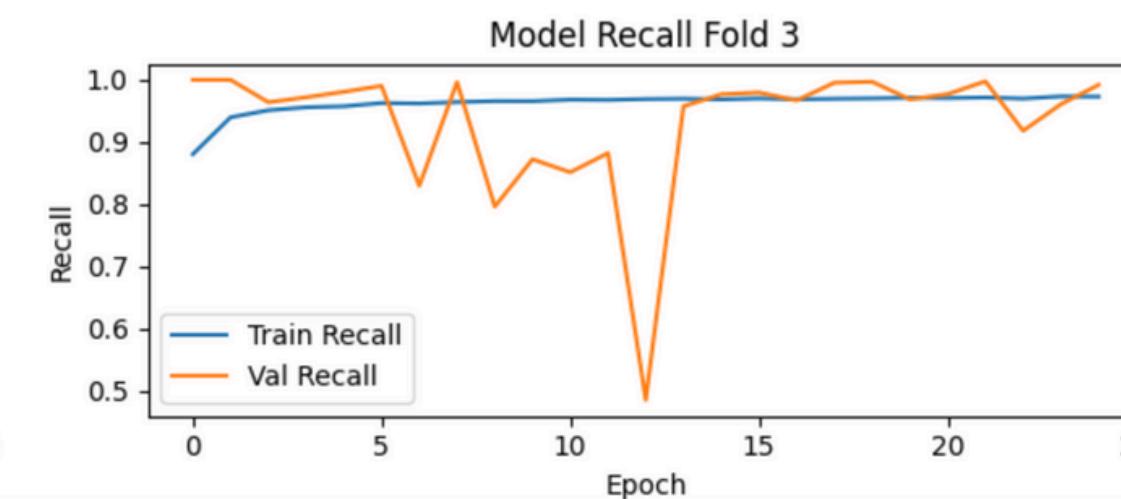
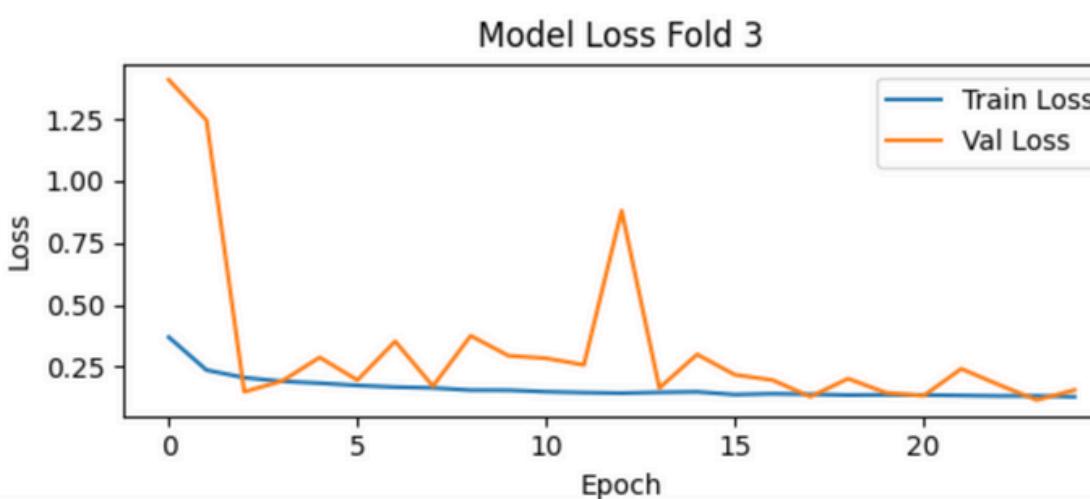
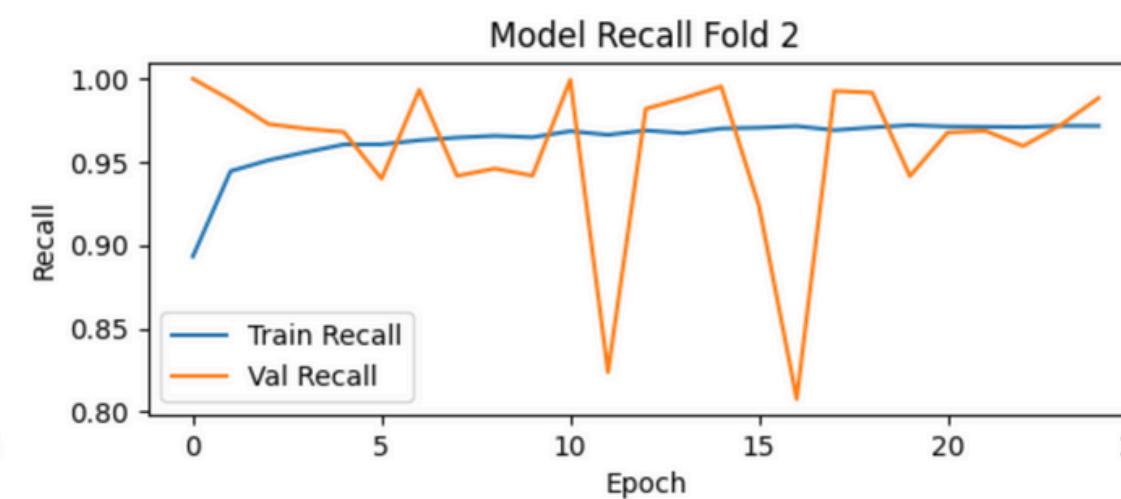
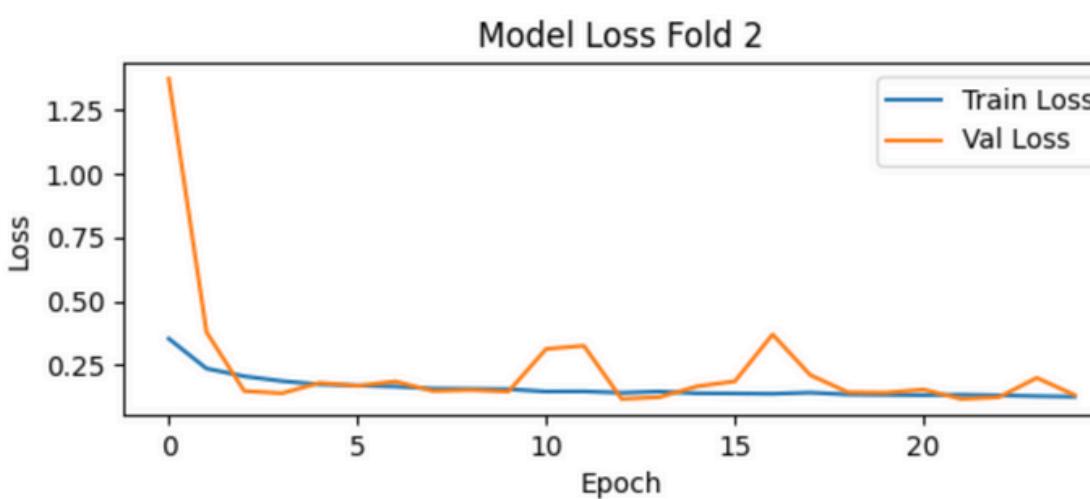
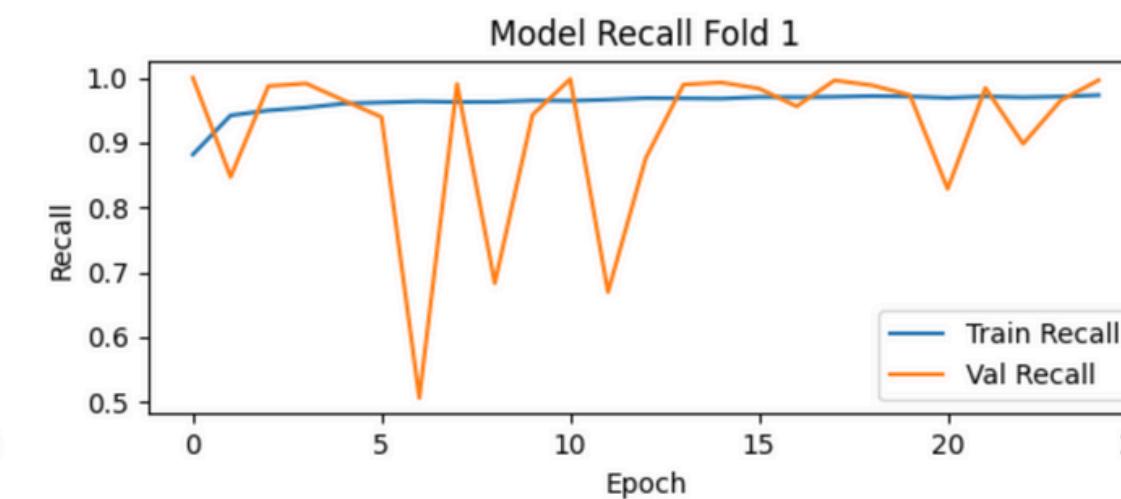
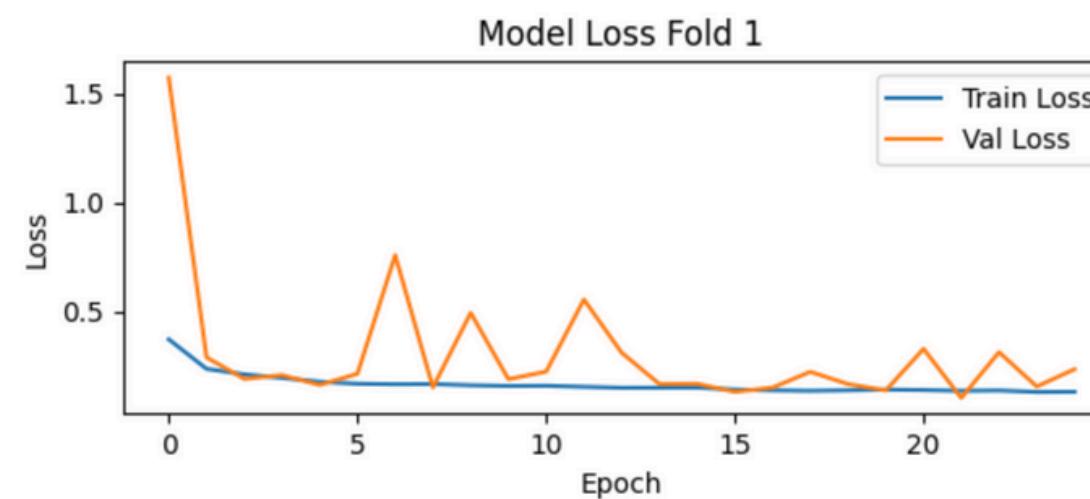


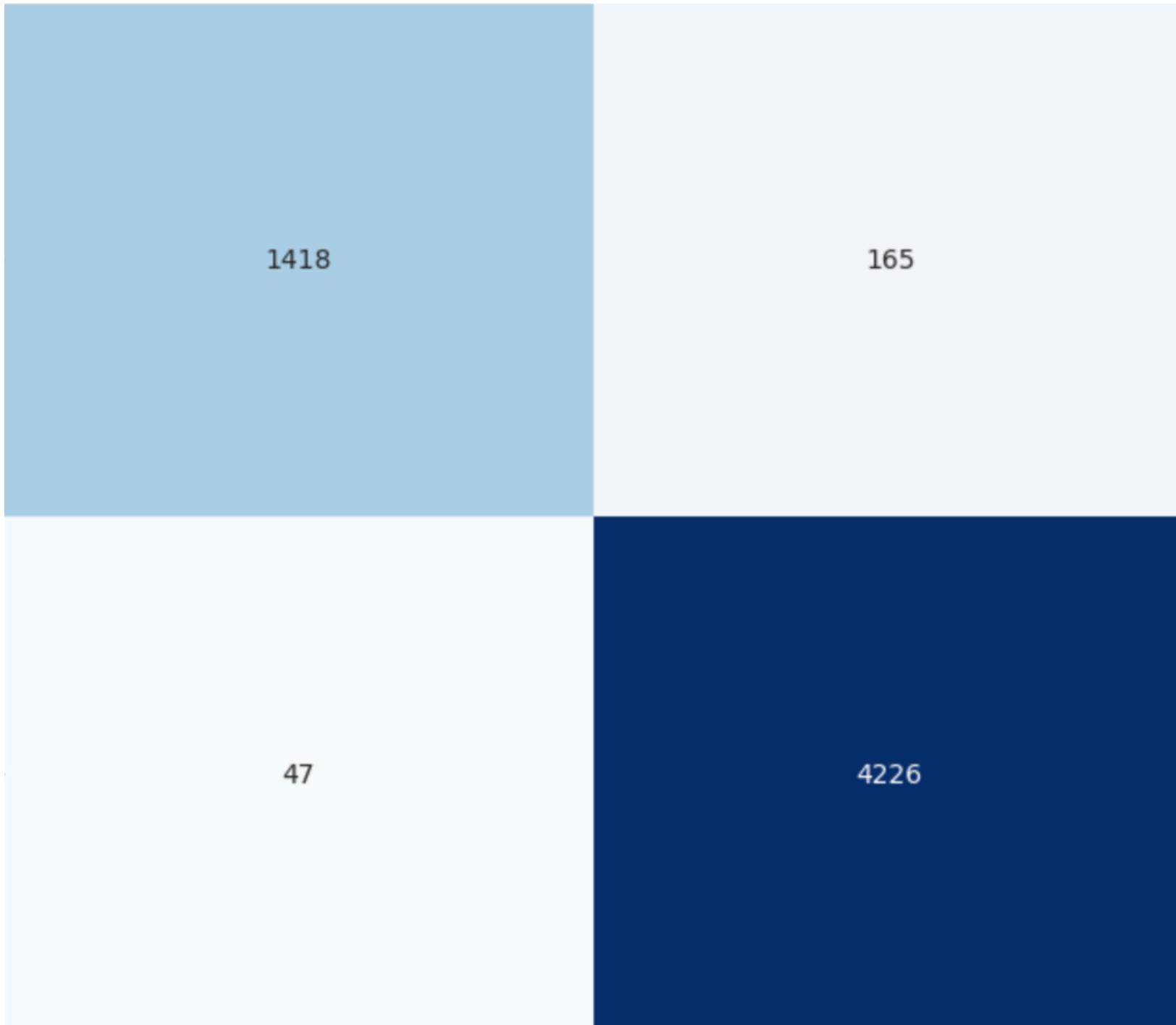
- Test sur l'optimiseur Adam et RMSProp
- Batch size de 8 à 2048
- Nombre d'éPOCHS de 3 à 150

Temps d'entraînement de 10 sec à 2H



- Prétraitement des données avec découpage basé sur la heatmap
- Augmentation des données (multiplication du jeu de données par 5)
- Compilation du modèle basé sur le rappel (recall) avec l'optimiseur RMSprop
- Entrainement sur 25 Epochs
- Validation croisée (K-fold) avec une taille de lot (batch size) de 64 sur 3 folds



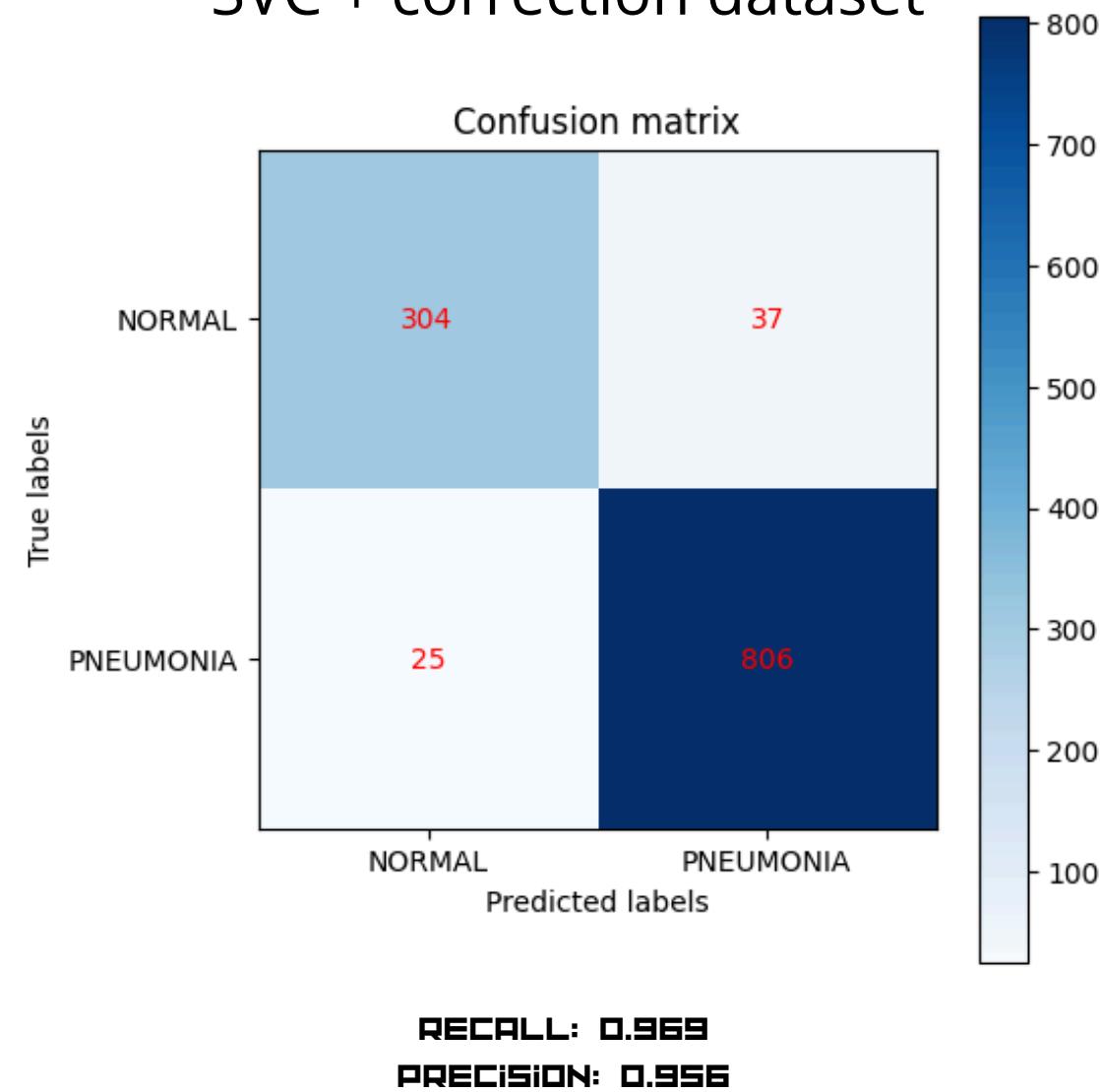


**RECALL: 0.989**

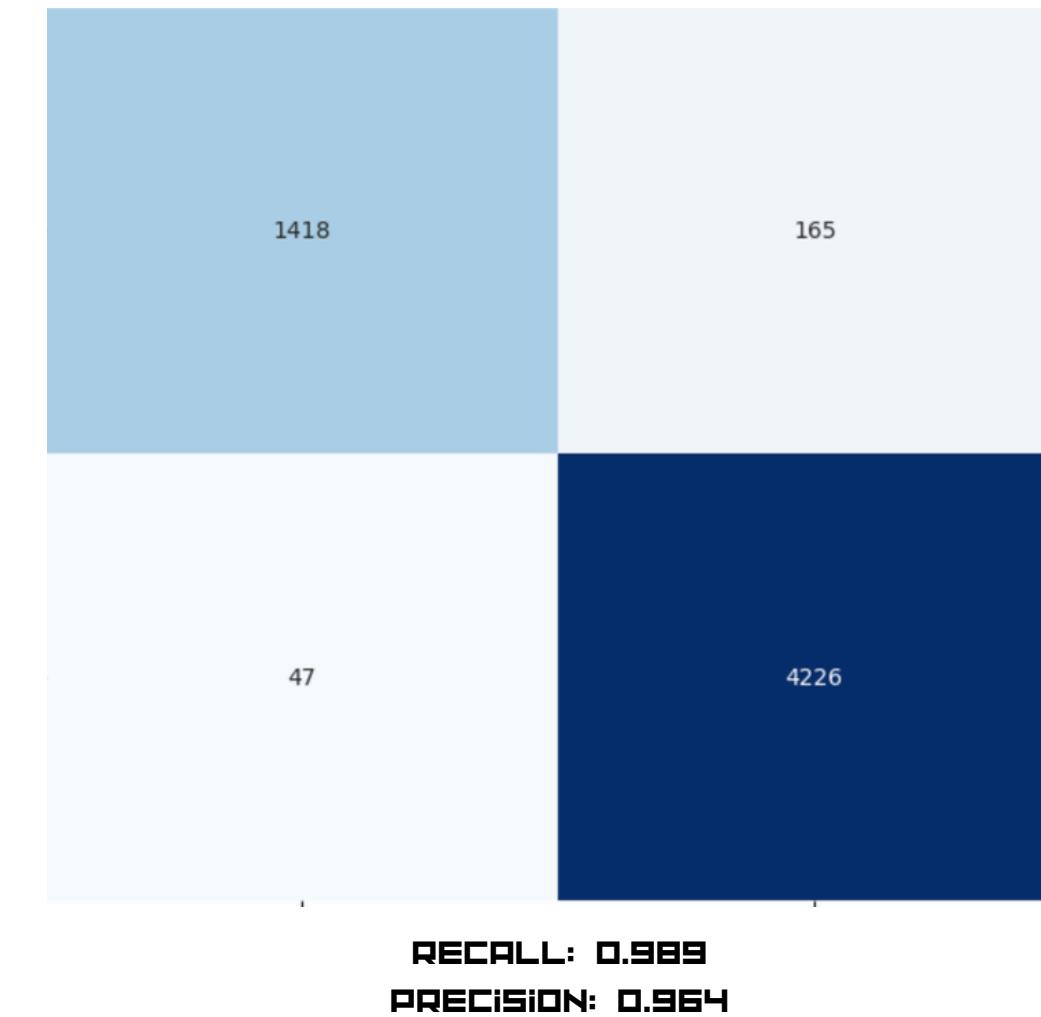
**PRECISION: 0.964**

# Conclusion

SVC + correction dataset



CNN + Heatmap cropped +correction dataset + data augmentation



**Merci pour votre écoute.**