TIPE:

CLASSIFICATION DU CANCER DU SEIN PAR CNN.

Thème: Santé Prévention

2021-2022

DAHHASSI Chaymae

Plan:

- ✓ Introduction
- ✓ Revue de littérature
- ✓ Approche de DL pour la classification du cancer du sein
  - ➤ Dataset
  - ➤ Architecture CNN
- ✓ Résultats et performance
- ✓ Annexe

C'est quoi un neurone?

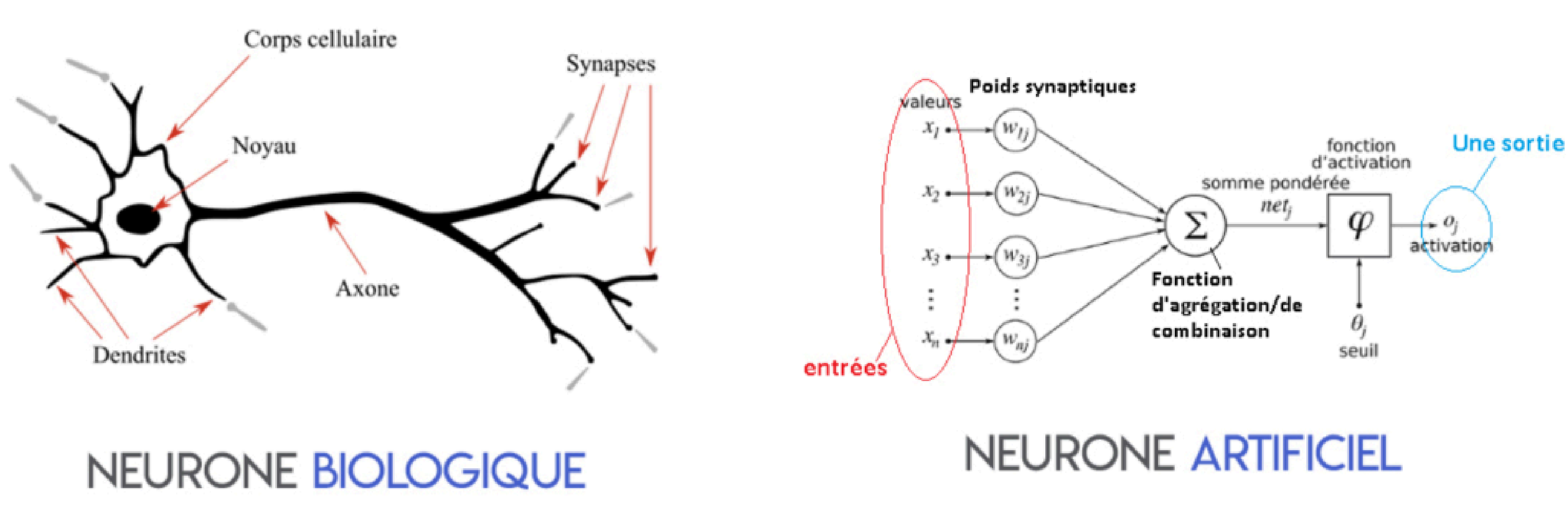


Fig. 1.1- Schéma du neurone biologique et artificiel [1]

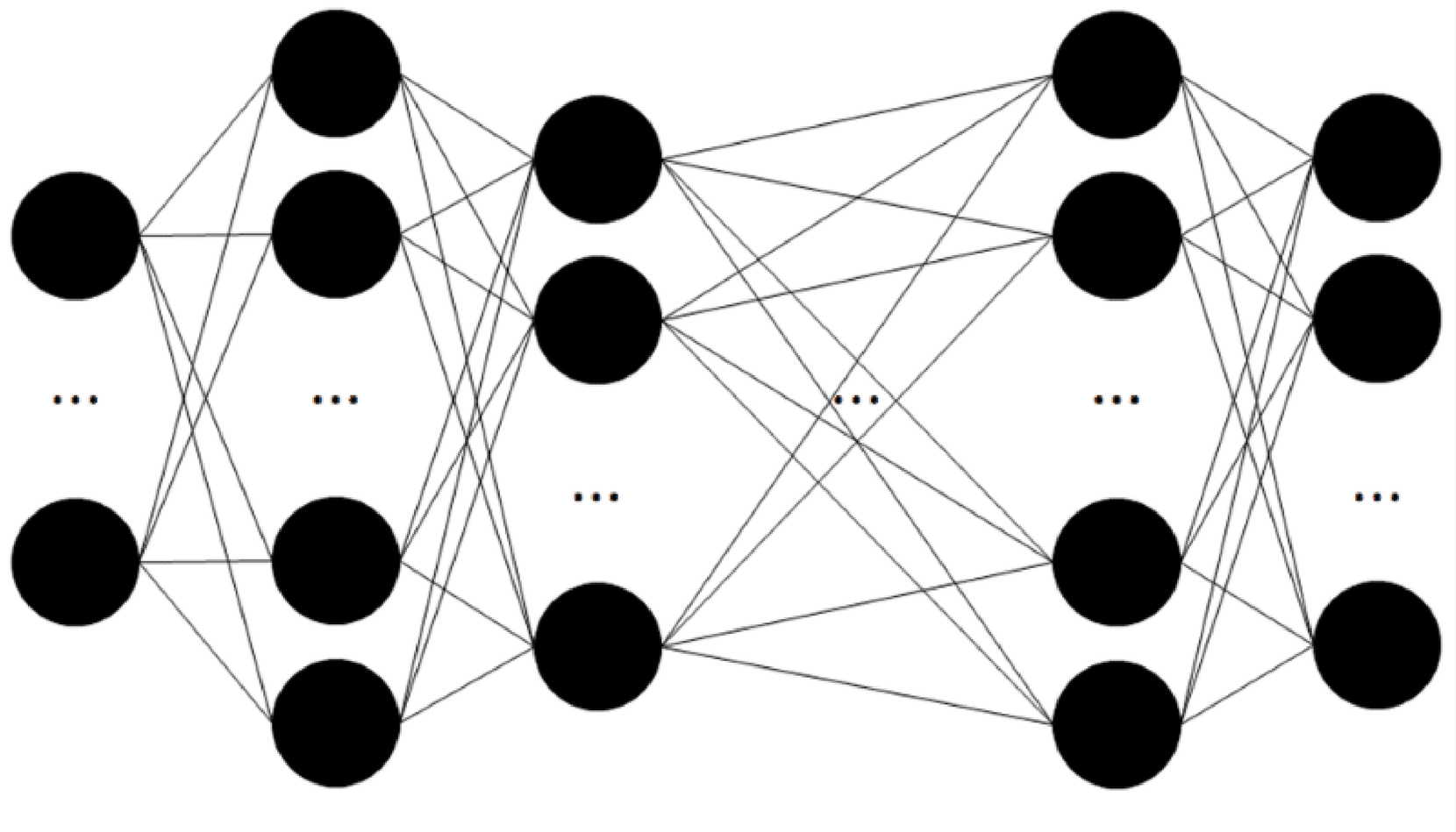


Fig. 1.2- Réseau de neurones artificiels [2]

C'est quoi les CNN et pourquoi les choisir?

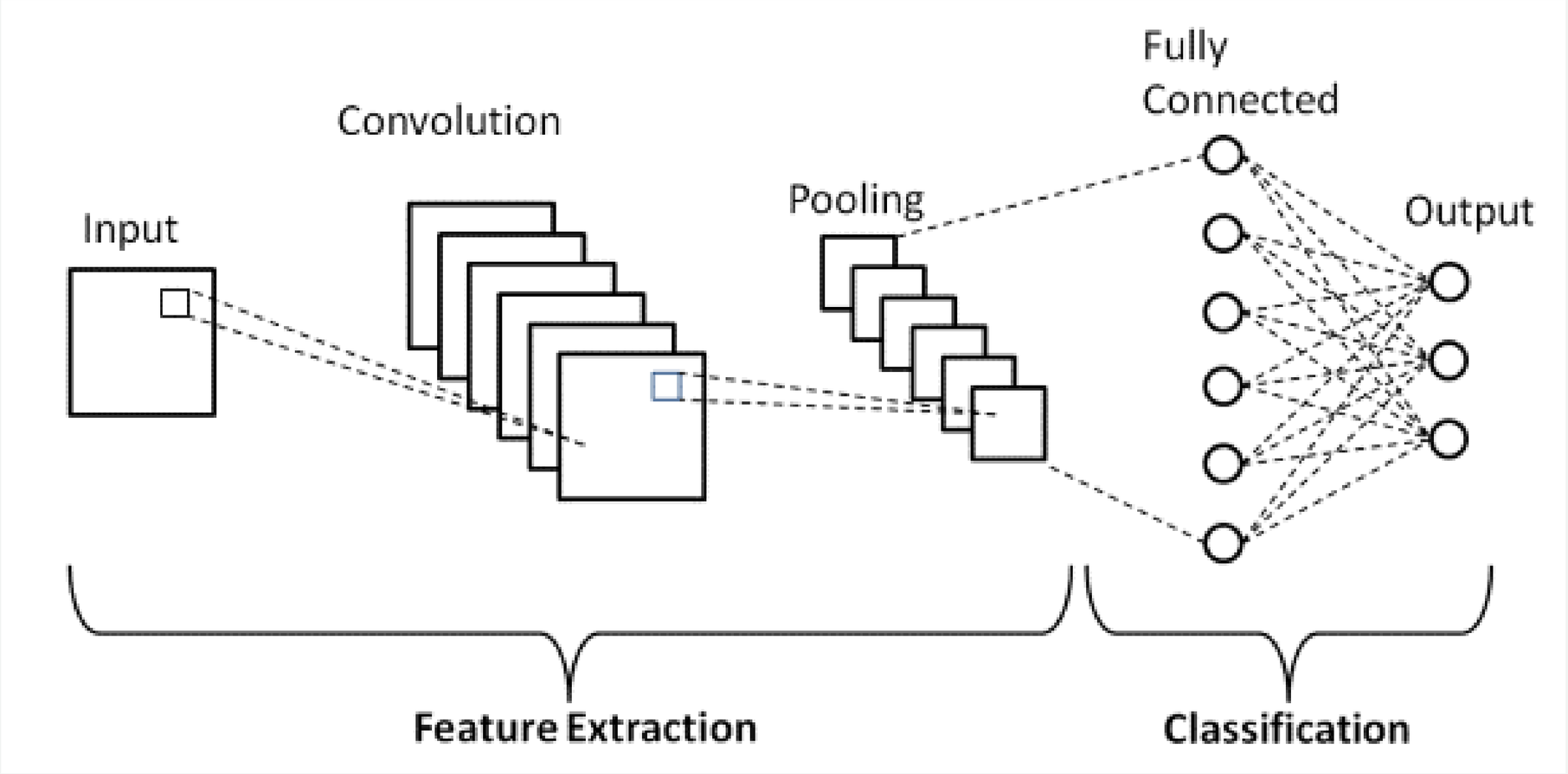


Fig. 1.3- Architecture CNN basique [3]

Ref.	Dataset	Features	Method	Accuracy
[4]	Wisconsin Breast Cancer (original) dataset (WBCD)	11 attributes 699 instances	SVM, C4.5, NB, k-NN	Best accuracy for SVM 97.13%
[5]	Two datasets	First dataset: 11 attributes with 699 instances Second dataset: 117 attributes with 102294 instances	SVM with three functions and two features: bagging and boosting	96.85%, 95%
[6],[7]	Wisconsin Breast Cancer dataset	30 attributes with 699 instances	Random Forest, Naive Bayes, SVM, KNN	97.9%
[8]	Wisconsin Prognostic Breast Cancer dataset	32 attributes with 194 instances	Compare (K-means, EM, PAM, and fuzzy c-means) with SVM and C5.0	Better results for SVM with accuracy 97%

[9]	Wisconsin Diagnostic Breast Cancer (WDBC) dataset	30 attributes of 569 patients with 569 instances	Compare supervised learning (SL) with semi-supervised learning (SSL) for 9 algorithms	K-NN (SL = 98% & SSL = 97%) and logistics regression (SL = 97% & SSL = 98%)
[10]	Wisconsin Breast Cancer (original) dataset (WBCD)	11 attributes with 699 instances	Boosting Artificial Neural Network (BANN) with two SVMs	100%
[11]	Wisconsin Breast Cancer dataset (WBCD)	32 attributes with 569 instances	SVM with stochastic gradient descent optimization, simple logistic regression learning, and multilayer perceptron network	99.44%

[12]	DDSM, IN breast, and BCDR	DDSM: 5316 images, 641 cases of patients IN breast: 200 images for 50 cases BCDR: 600 images from 300 patients	CNN	97.35%, 95.50%, 96.67% for three datasets respectively
[13]	WBCD	9 attributes with 5699 instances	Deep neural network (DNN)	98.62%
[14]	The Datasets were collected at two medical institutions	990 images, 540 Malignant masses, and 450 benign lesions	CNN	87.68%
[15]	Data was collected from (2010-2016) at five imaging sites affiliated with New York University School of Medicine	1,001,093 images from 141,473 patients	CNN	The accuracy was calculated based on area under a curve (AUC)

[16]	Data was collected independently	12,000 cases, including 4000 samples proven cancers	CNN	The accuracy was calculated based on area under a curve (AUC)
[17]	Private dataset	67,520 mammographic images from 16,968 women	CNN	95%

Dataset: Breast Cancer Wisconsin (Diagnostic) Data Set (WBCD)

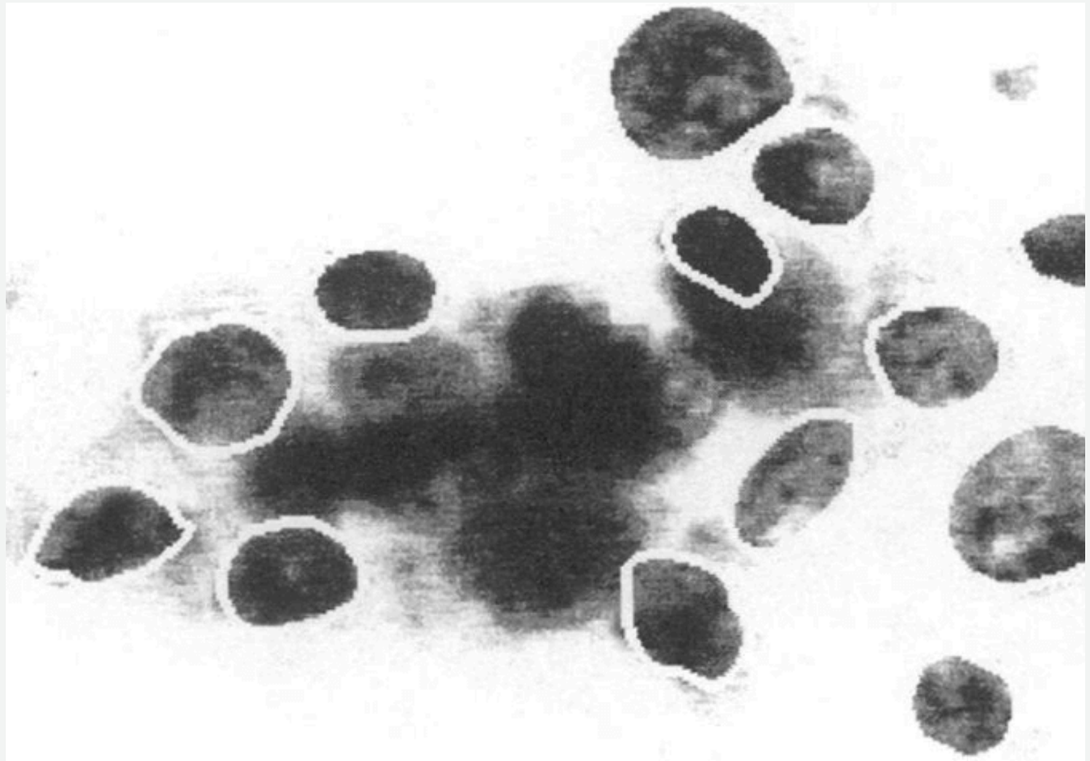


Fig. 2.1- Image histopathologique maligne obtenue par biopsie à l'aiguille fine (FNA)

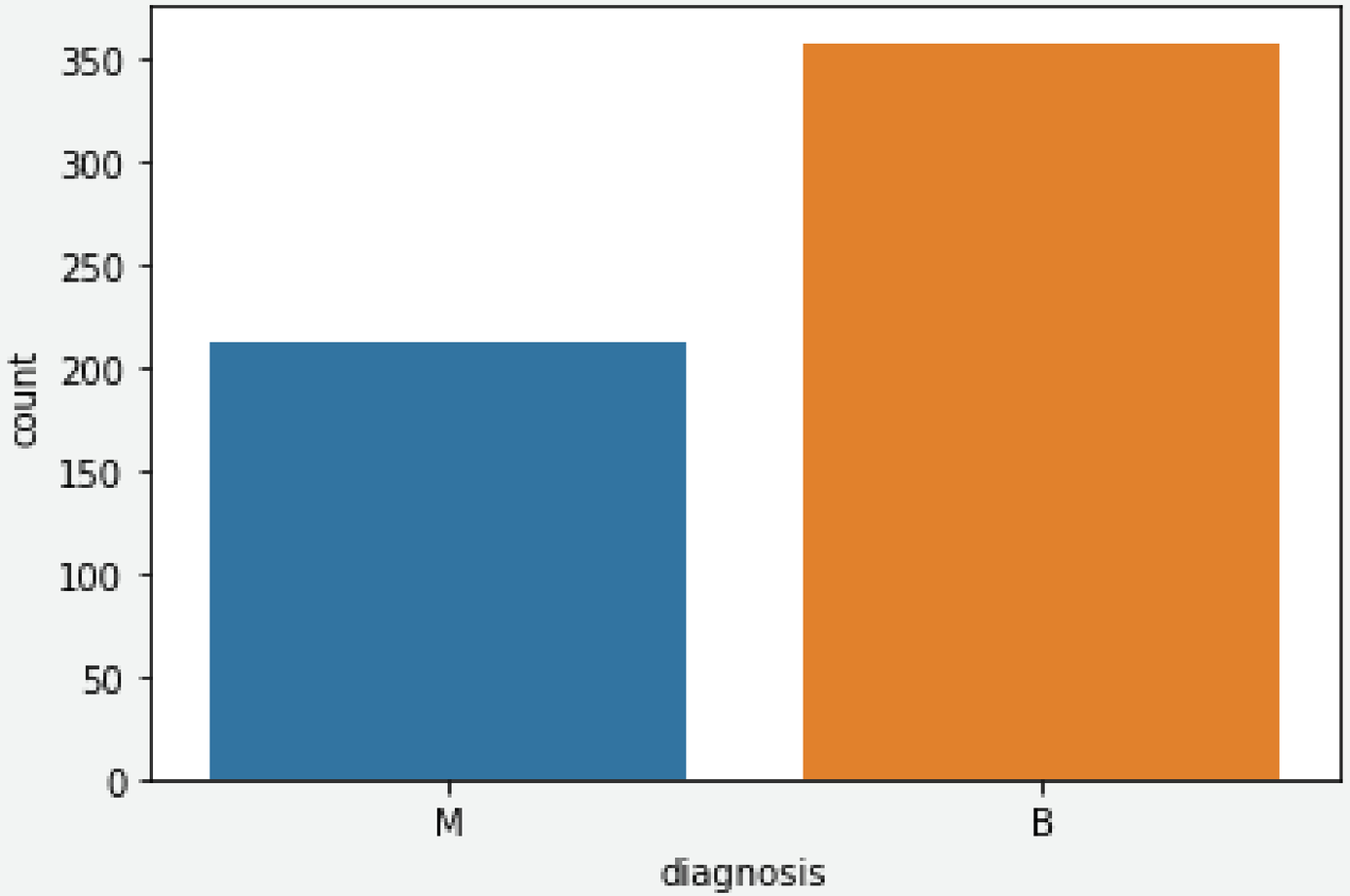
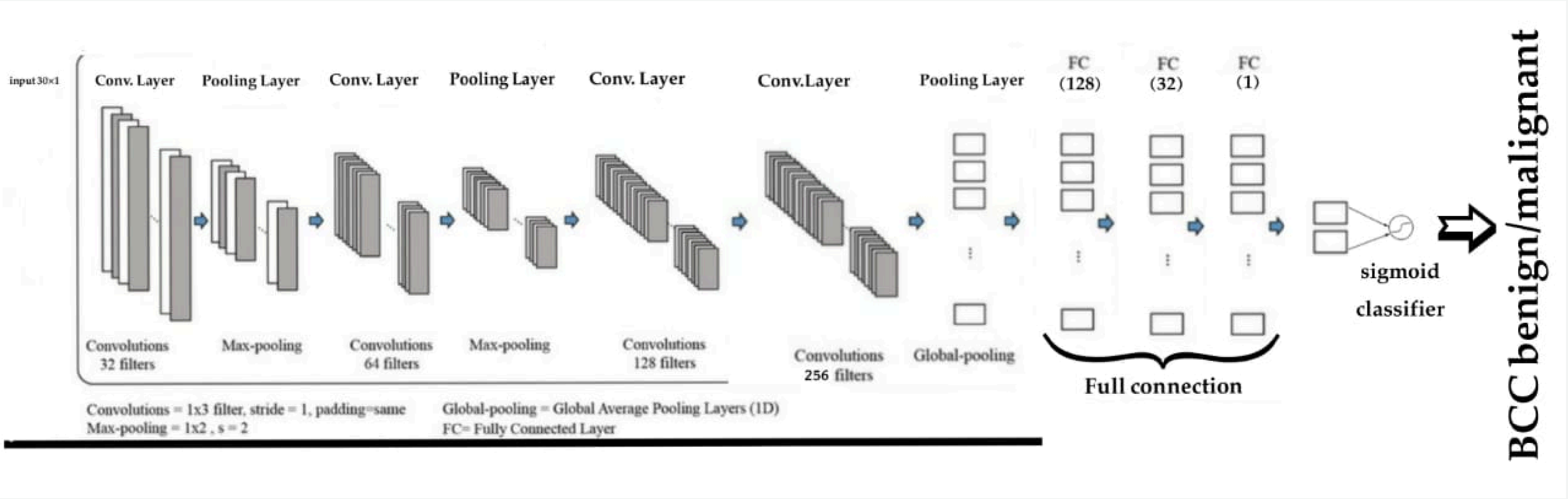


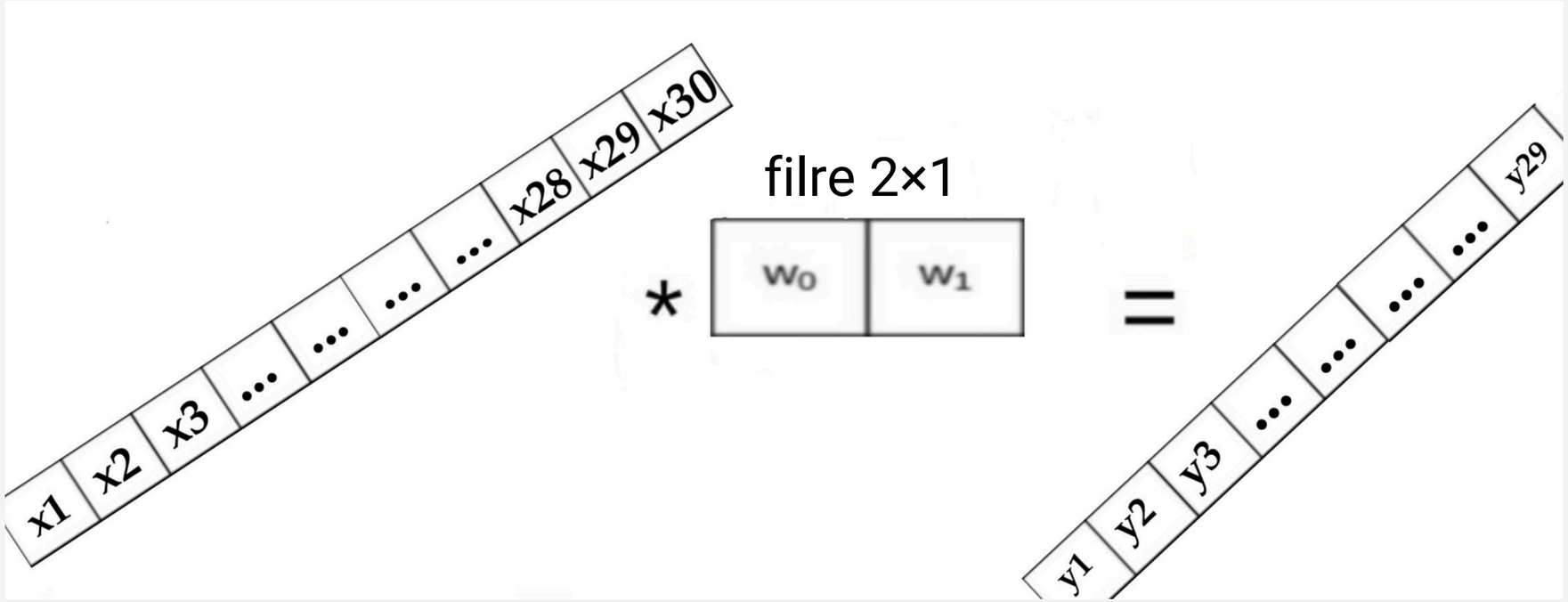
Fig. 2.2- Distribution des tumeurs bénignes et malignes dans WBCD

Architecture CNN proposée:



La couche de convolution:

La convolution à une dimension:

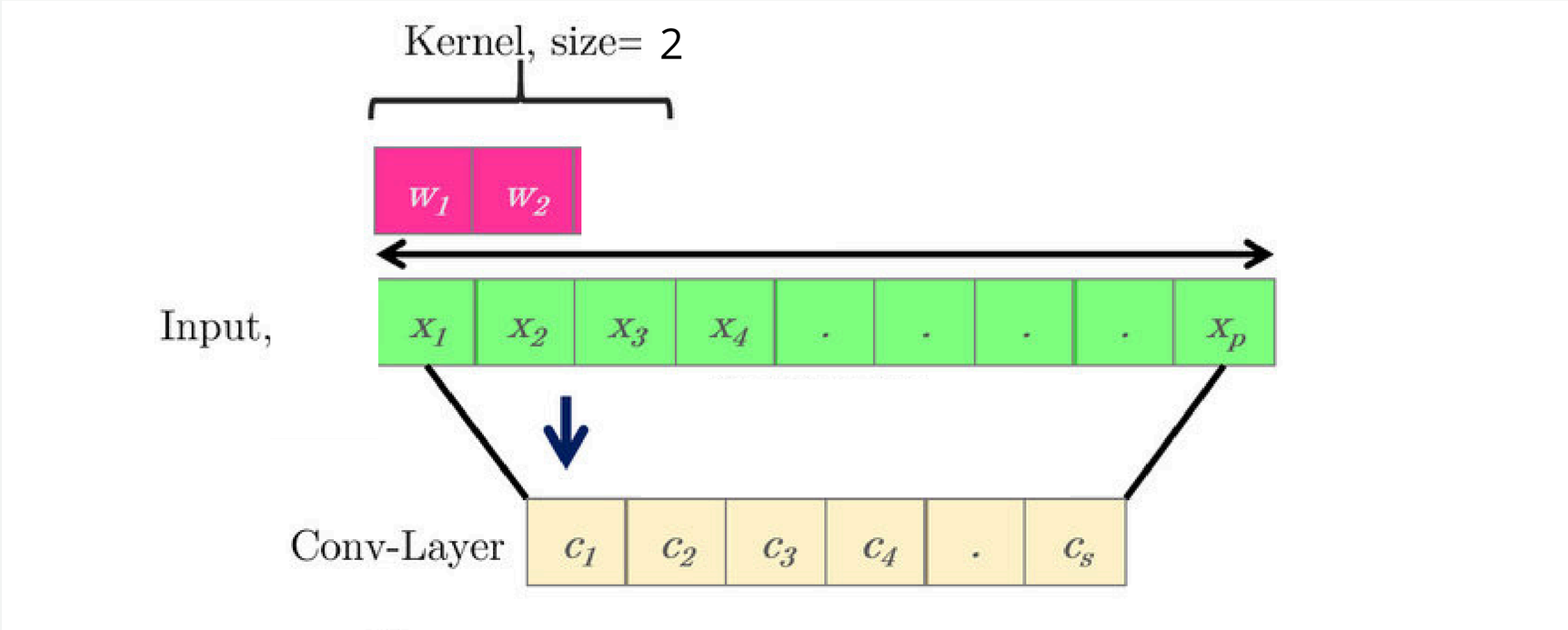


$y1 = x1 \times w0 + x2 \times w1$

$y2 = x2 \times w0 + x3 \times w1$

...

$y29 = x29 \times w0 + x30 \times w1$


Kernel, size= 2
w_1 w_2
Input,
x_1 x_2 x_3 x_4 . . . . x_p
Conv-Layer
c_1 c_2 c_3 c_4 . c_s

$$C_1 = X_1 \times W_1 + X_2 \times W_2$$

$$C_2 = X_2 \times W_1 + X_3 \times W_2$$

$$C_3 = X_3 \times W_1 + X_4 \times W_2$$

$$\dots$$

$$C_s = X_{p-1} \times W_1 + X_p \times W_2$$

La couche de correction ou d'activation Relu:

Fonction LeakyReLU:

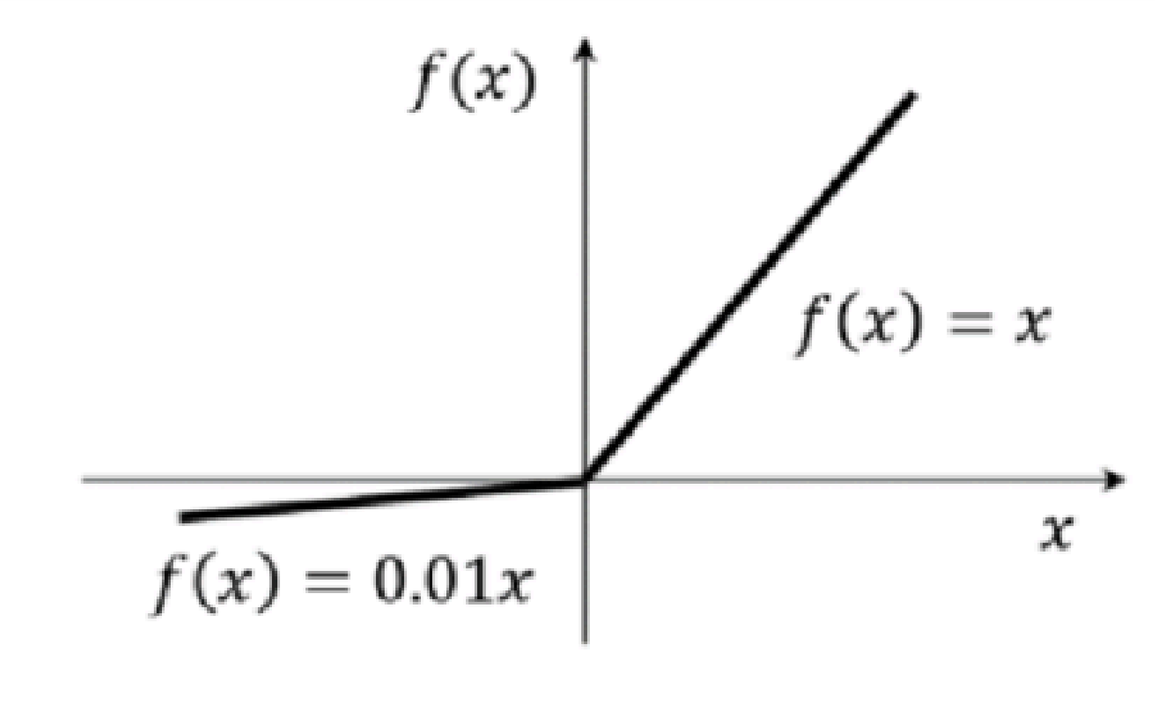


Fig. 2.3- Fonction LeakyReLU

La couche de pooling:

1- Max Pooling 1D:

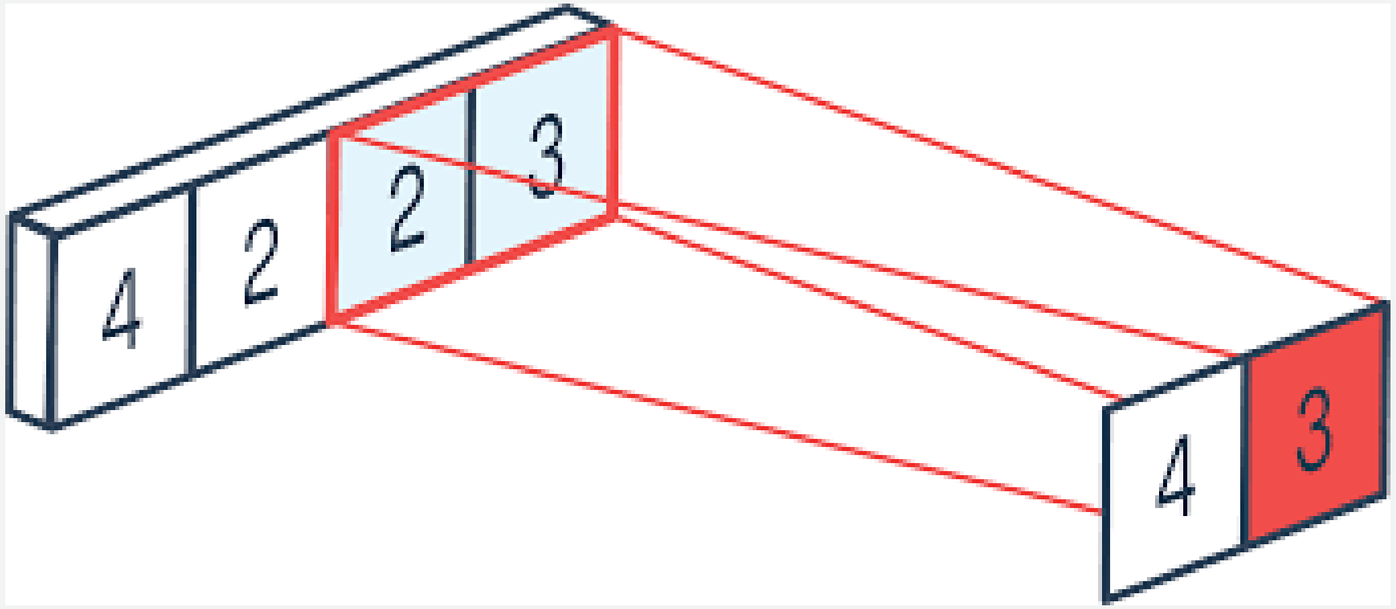


Fig. 2.4- Block Max pooling à une dimension, pooling size=2

2- Global average pooling 1D:

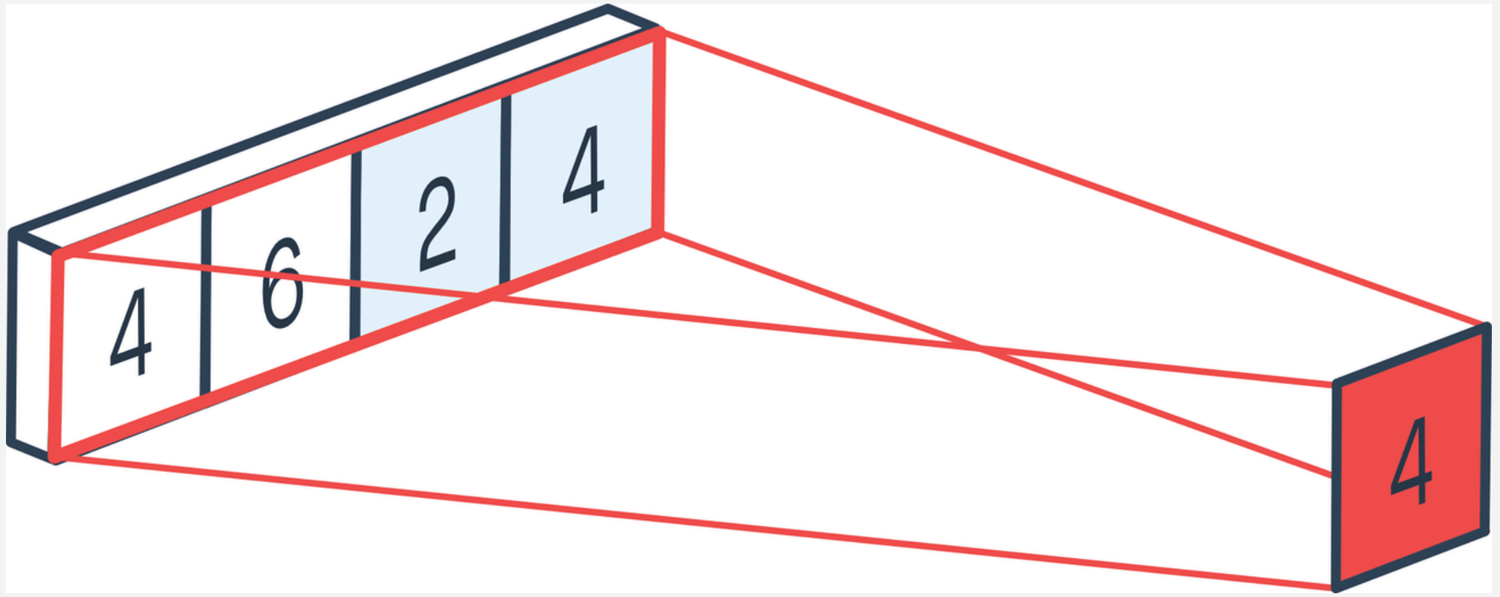


Fig. 2.5- Global average pooling à une dimension

La couche fully-connected:

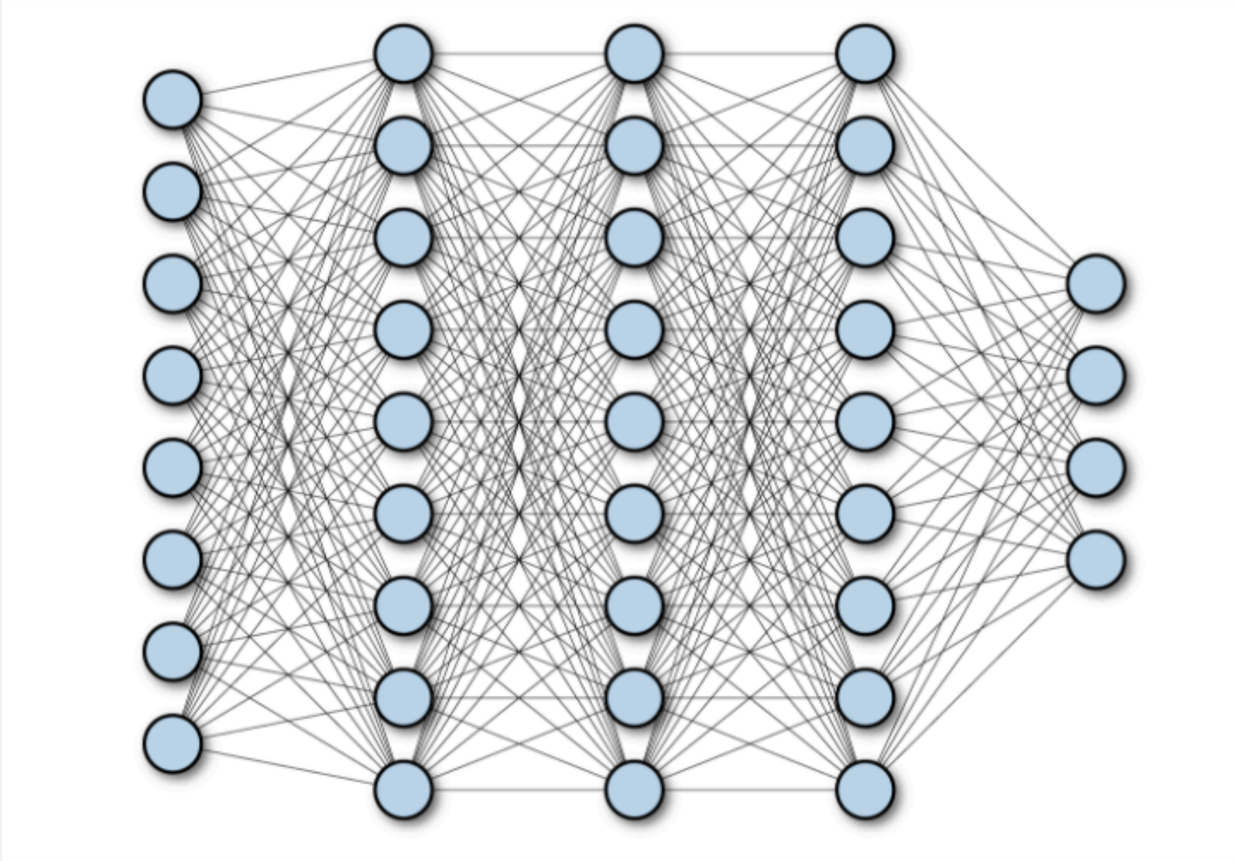


Fig. 2.6- Couche fully-connected

Fonction sigmoide : $\forall x \in \mathbb{R}$, $\sigma(x) = \frac{1}{1+e^{-x}}$

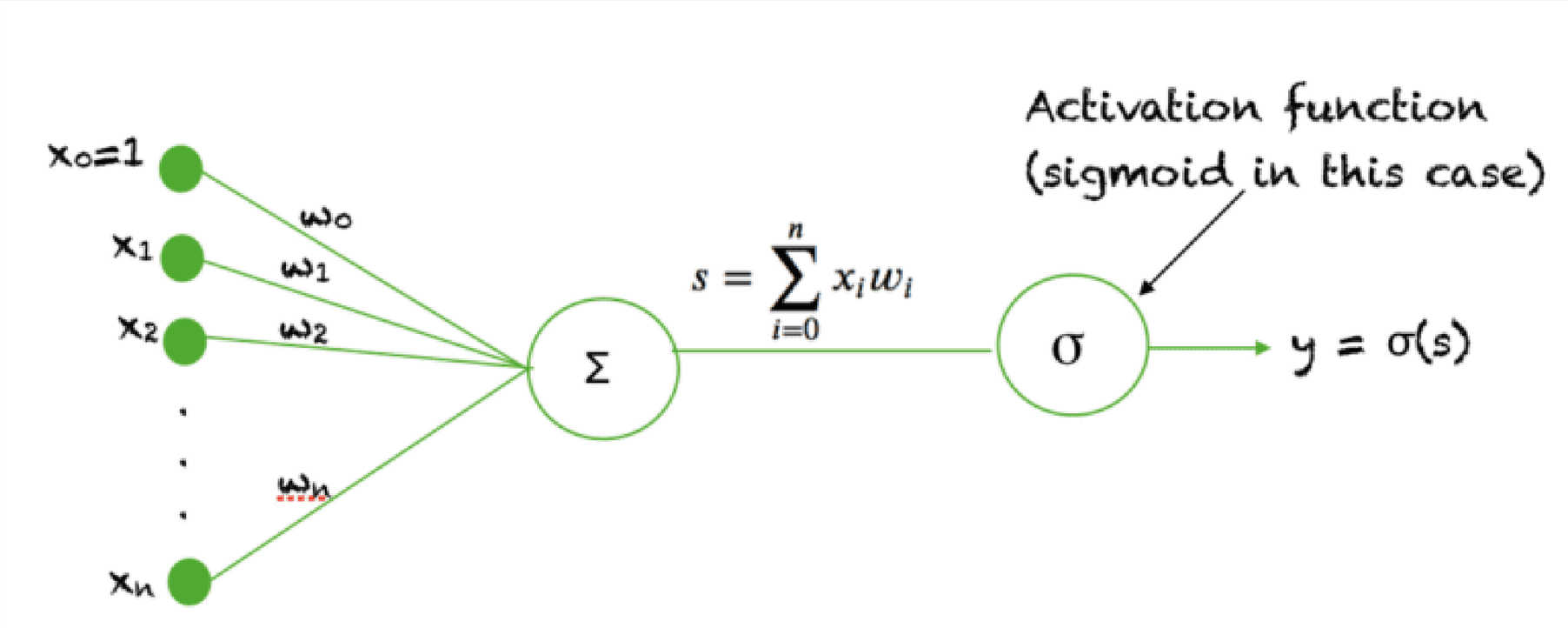


Fig. 2.7- Fonction d'activation Sigmoide

Risque de surapprentissage ?

L1 Régularisation:

Terme de régularisation:

$$\Omega(W) = ||W||_1 = \sum_i \sum_j |w_{ij}|$$

$w_{i\ j}$: terme de la i-éme ligne et la j-ème colone de la matrice des poids.

Nouvelle fonction Loss:

$$\hat{\mathcal{L}}(W) = \alpha||W||_1 + \mathcal{L}(W)$$

α : taux de régularisation.

L(W) : ancienne fonction Loss.

Fonction Loss: binary cross-entropy / log loss

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

$p(y_i)$ est la probabilité de la classe 1 et $(1-p(y_i))$ est la probabilité de la classe 0.

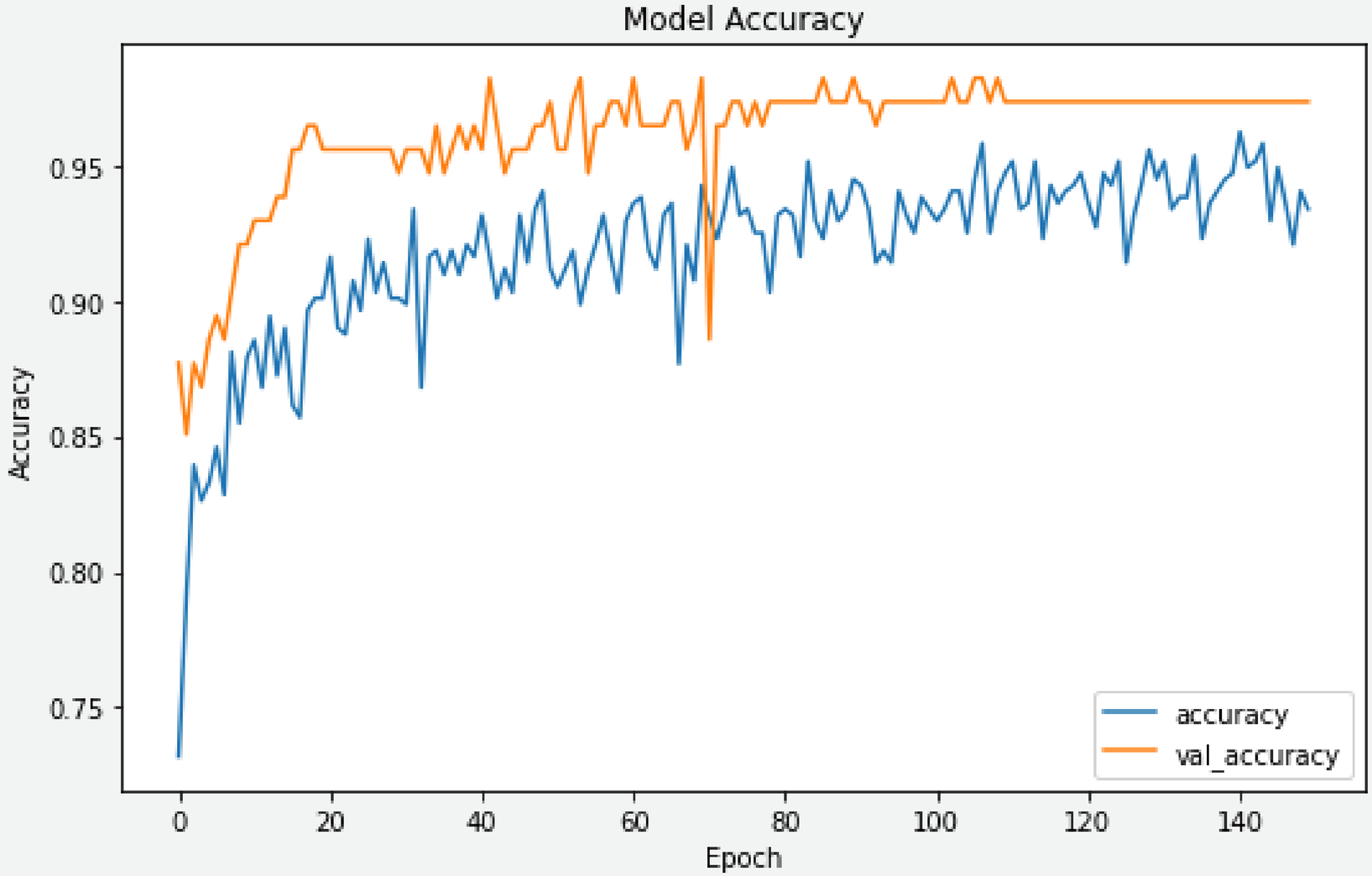


Fig. 3.1- La précision du modèle CNN

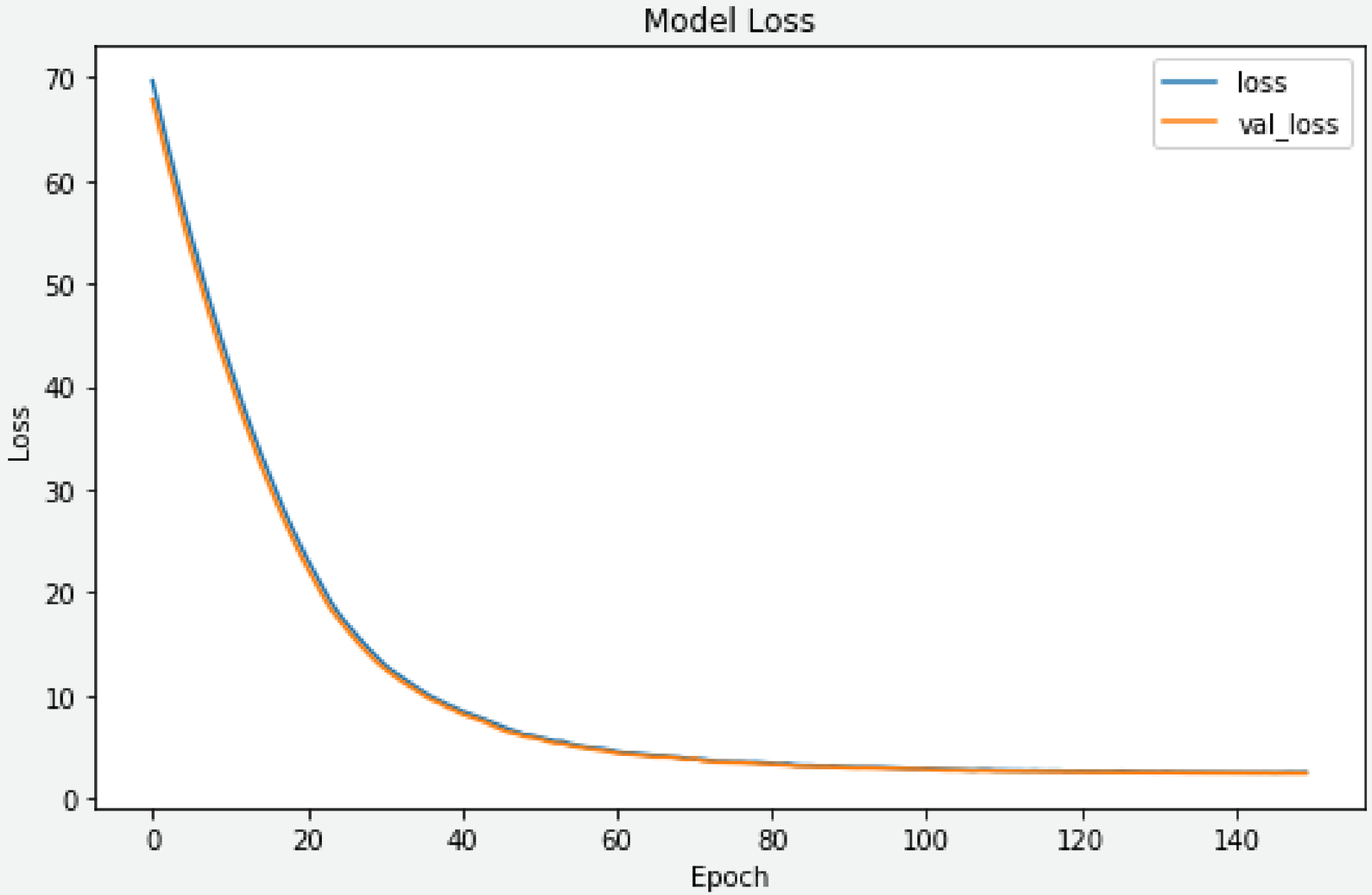


Fig 3.2- Courbe de perte du modèle CNN

	précision	recall	f1-score	support
0 (bénigne)	0.97	0.99	0.98	71
1 (maligne)	0.98	0.95	0.96	43
Précision	_	_	0.97	114

Fig 3.3- Tableau pour évaluer la performance du modèle CNN

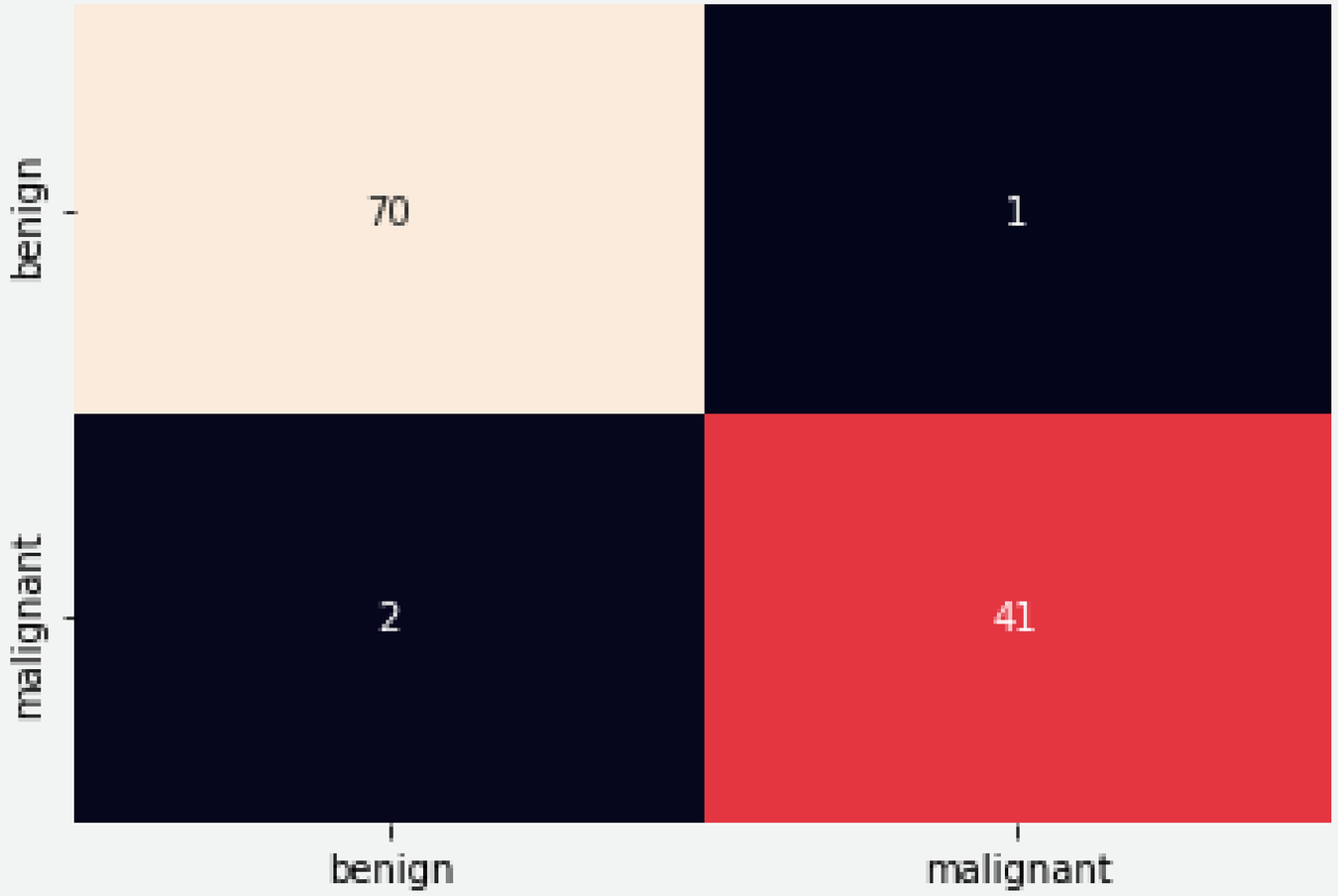


Fig. 3.4- Matrice de confusion du modèle CNN

Modèle CNN plus performant ?

Quelle différence entre les deux modèles ?

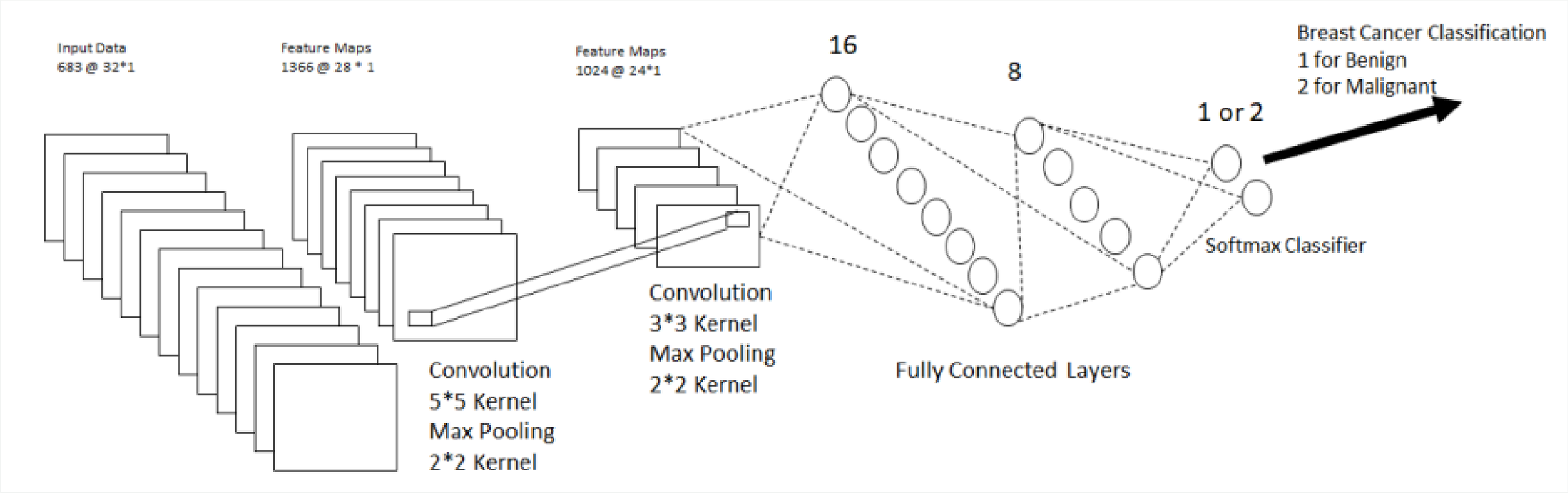


Fig. 4.1- Architecture CNN pour la classification du cancer du sein. [18]

Fonction Softmax:

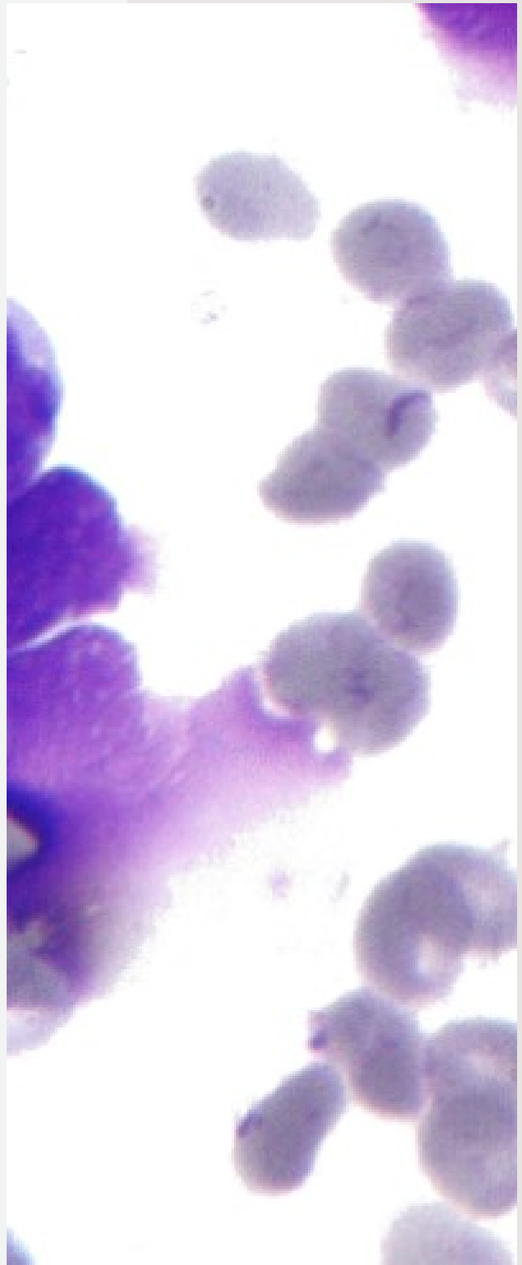
$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$\vec{z}$: Le vecteur d'entrée de la fonction softmax composé de K valeurs z_j.

K: nombre de classes.

Method Proposed	Train + validate to Test partition (%)	Accuracy (%)	Sensitivity (%)	Selectivity (%)
Our proposed method	80 – 20	99.1	100	97.8

Fig. 4.2- Evaluation de la performance de l'architecture CNN (WBCD 569 échantillons)



Conclusion:

Merci pour votre attention

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
!pip install -q keras
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, Conv1D, MaxPool1D, GlobalAveragePooling1D
from sklearn import preprocessing
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from keras.layers import LeakyReLU

dataset_dir = '../input/breast-cancer-wisconsin-data/data.csv'
SAVE_MODEL_DIR = '../kaggle/working/best_model.h5'
```

```python
#importer les données
class dataset():
    def __init__(self):
        self.data = pd.read_csv(dataset_dir)

    def label_encoding(self):
        le = preprocessing.LabelEncoder()
        self.data['diagnosis'] = le.fit_transform(self.data['diagnosis'])
        return self.data
    #définir une fonction pour préparer les X et Y des données

    def prepare_data(self):
        self.data = self.label_encoding()
        self.X = self.data.drop(columns=['diagnosis', 'id', 'Unnamed: 32'], axis=1)
        self.Y = self.data['diagnosis']
        return self.X, self.Y
    #définir une fonction pour séparer les données en données d'entrainement et de test

    def data_preparation(self):
        self.X, self.Y = self.prepare_data()
        self.X_train, self.X_test, self.Y_train, self.Y_test = train_test_split(self.X, self.Y, test_size=0.2, random_state=42)
        self.scaler = StandardScaler()
        self.X_train = self.scaler.fit_transform(self.X_train)
        self.X_test = self.scaler.transform(self.X_test)
        self.X_train = self.X_train.reshape(self.X_train.shape[0], self.X_train.shape[1], 1)
        self.X_test = self.X_test.reshape(self.X_test.shape[0], self.X_test.shape[1], 1)
        return self.X_train, self.X_test, self.Y_train, self.Y_test
```

```python
#définir le modèle CNN avec l'architecture suivante:
#couche de convolution avec 32 filtres et un kernel de taille 3
#utiliser la fonction d'activation LeakyRelu
#utiliser la fonction régularisation L1 de facteur 0,01 appliquée à la matrice kernel
#normalisation de batch
#dropout
class CNN():
    def __init__(self, X_train, X_test, Y_train, Y_test):
        self.X_train = X_train
        self.X_test = X_test
        self.Y_train = Y_train
        self.Y_test = Y_test
        self.model = Sequential()

    def model_architecture(self):
        self.model.add(Conv1D(32, 2, input_shape=(30, 1), kernel_regularizer=keras.regularizers.l1(0.01), kernel_initializer='HeNormal'))
        self.model.add(LeakyReLU(alpha=0.01))
        self.model.add(BatchNormalization())
        self.model.add(Dropout(0.2))
        self.model.add(MaxPool1D(pool_size=2, padding='same'))
        self.model.add(Conv1D(64, 2, kernel_regularizer=keras.regularizers.l1(0.01), kernel_initializer='HeNormal'))
        self.model.add(LeakyReLU(alpha=0.01))
        self.model.add(BatchNormalization())
        self.model.add(Dropout(0.2))
        self.model.add(MaxPool1D(pool_size=2, padding='same'))
        self.model.add(Conv1D(128, 2, kernel_regularizer=keras.regularizers.l1(0.01), kernel_initializer='HeNormal'))
        self.model.add(LeakyReLU(alpha=0.01))
        self.model.add(BatchNormalization())
```

```
        self.model.add(Dropout(0.2))
        self.model.add(Conv1D(256, 2, kernel_regularizer=keras.regularizers.l1(0.01), kernel_initializer='HeNormal'))
        self.model.add(LeakyReLU(alpha=0.01))
        self.model.add(BatchNormalization())
        self.model.add(Dropout(0.2))
        self.model.add(GlobalAveragePooling1D())
        self.model.add(Dense(128))
        self.model.add(LeakyReLU(alpha=0.01))
        self.model.add(Dropout(0.2))
        self.model.add(Dense(32))
        self.model.add(Dense(1, activation='sigmoid'))
        return self.model

    def plot_history_accuracy(self, history):
        fig, axs = plt.subplots(1, 2, figsize=(15, 5))
        axs[0].plot(history.history['accuracy'], label='accuracy')
        axs[0].plot(history.history['val_accuracy'], label='val_accuracy')
        axs[0].set_ylabel('Accuracy')
        axs[0].set_xlabel('Epoch')
        axs[0].set_title('Model Accuracy')
        axs[0].legend(loc='lower right')
        axs[1].plot(history.history['loss'], label='loss')
        axs[1].plot(history.history['val_loss'], label='val_loss')
        axs[1].set_ylabel('Loss')
        axs[1].set_xlabel('Epoch')
        axs[1].set_title('Model Loss')
        axs[1].legend(loc='upper right')
        plt.tight_layout()
        plt.show()
```

```python
    #définir une fonction pour entrainer le modèle

    def train_evaluation_model(self, BATCH_SIZE=32, EPOCHS=50):
        self.model = self.model_architecture()
        self.model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
        #définir une fonction de rappel pour enregistrer le modèle avec la meilleure accuracy dans la phase de validation
        callback = tf.keras.callbacks.ModelCheckpoint(filepath=SAVE_MODEL_DIR, monitor='val_accuracy', save_best_only=True, verbose=1)
        #réduire le taux d'apprentissage
        reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy', factor=0.8, patience=6, min_lr=0.00001, verbose=1)
        history = self.model.fit(self.X_train, self.Y_train, batch_size=BATCH_SIZE, epochs=EPOCHS, validation_data=(self.X_test, self.Y_test), verbose=1, callbacks=[callback, reduce_lr])
        self.plot_history_accuracy(history)
        predict_labels = (self.model.predict(self.X_test) > 0.5).astype(np.int32)
        print(classification_report(self.Y_test, predict_labels))
        print("Accuracy on the test Step:", accuracy_score(self.Y_test, predict_labels))
        #la matrice de convolution
        conf_matrix = confusion_matrix(self.Y_test, predict_labels)
      sns.heatmap(conf_matrix, annot=True, fmt="d", cbar=False, xticklabels=['benign', 'malignant'], yticklabels=['benign', 'malignant'])
    #définir une fonction pour évaluer le modèle
    def evaluate_model(self):
        self.model1 = self.model_architecture()
        self.model1.load_weights(SAVE_MODEL_DIR)
        predict_labels = (self.model1.predict(self.X_test) > 0.5).astype(np.int)
        print(classification_report(self.Y_test, predict_labels))
        print("Accuracy on the test Step:", accuracy_score(self.Y_test, predict_labels))
        #la matrice de convolution
        conf_matrix = confusion_matrix(self.Y_test, predict_labels)
        sns.heatmap(conf_matrix, annot=True, fmt="d", cbar=False, xticklabels=['benign', 'malignant'], yticklabels=['benign', 'malignant'])
```

```python
#entrainer, valider et tester le modèle
def main():
    dataset_obj = dataset()
    #préparer les données
    X, Y = dataset_obj.prepare_data()
    print(X.shape, Y.shape)
    #diviser les données en données d'entrainement et en données de test
    X_train, X_test, Y_train, Y_test = dataset_obj.data_preparation()
    cnn_obj = CNN(X_train, X_test, Y_train, Y_test)
    #entrainer le modèle
    cnn_obj.train_evaluation_model(BATCH_SIZE=12, EPOCHS=150)

if __name__ == '__main__':
    main()
```

Références:

[1] https://deeplylearning.fr/cours-theoriques-deep-learning/fonctionnement-du-neurone-artificiel/
[2] https://fr.blog.businessdecision.com/tutoriel-machine-learning-comprendre-ce-quest-un-reseau-de-neurones-et-en-creer-un/
[3] https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697
[4] H. Asri, H. Mousannif, H. Al Moatassime, and T. Noel, “Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis,” Procedia Comput. Sci., vol. 83, no. Fams, pp. 1064–1069, 2016.
[5] M. W. Huang, C. W. Chen, W. C. Lin, S. W. Ke, and C. F. Tsai, “SVM and SVM ensembles in breast cancer prediction,” PLoS One, vol. 12, no. 1, pp. 1–14, 2017.
[6] Y. K. and M. Bahaj, “Applying Best Machine Learning Algorithms for Breast Cancer Prediction and Classification,” in International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), 2018, pp. 1–5
[7] Y. K. and M. Bahaj, “Feature Selection with Fast Correlation-Based Filter for Breast Cancer Prediction and Classification Using Machine Learning Algorithms,” in 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2018, pp. 1–6
[8] R. Rawal, “BREAST CANCER PREDICTION USING MACHINE LEARNING,” J. Emerg. Technol. Innov. Res., vol. 7, no. 5, 2020
[9] N. Al-Azzam and I. Shatnawi, “Comparing supervised and semi-supervised Machine Learning Models on Diagnosing Breast Cancer,” Ann. Med. Surg., vol. 62, no. December 2020, pp. 53–64, 2021

[10] M. Abdar and V. Makarenkov, “CWV-BANN-SVM ensemble learning classifier for an accurate diagnosis of breast cancer,” Meas. J. Int. Meas. Confed., vol. 146, no. May, pp. 557–570, 2019
[11] A. S. Assiri, S. Nazir, and S. A. Velastin, “Breast Tumor Classification Using an Ensemble Machine Learning Method,” J. Imaging, vol. 6, no. 6, p. 39, May 2020
[12] A. O. Chougrad H, Zouaki H, “Deep convolutional neural networks for breast cancer screening,” Comput Methods Prog Biomed, vol. 157, 19–30, 2018
[13] P. V. S. S. R. C. M. S. Karthik, R. Srinivasa Perumal, “Breast cancer classification using deep neural networks,” Knowl Comput Its Appl Knowl Manip Process Tech, vol. 1, pp. 227–241, 2018
[14] H. Cai, Q. Huang, W. Rong, Y. Song, J. Li, J. Wang, J. Chen, and L. Li, “Breast Microcalcification Diagnosis Using Deep Convolutional Neural Network from Digital Mammograms,” Comput. Math. Methods Med., vol. 2019, no. January 2020, 2019
[15] N. W. et Al, “Deep Neural Networks Improve Radiologists’ Performance in Breast Cancer Screening,” IEEE Trans. Med. Imaging, vol. 39, no. 4, pp. 1184–1194, 2020
[16] E. F. Conant, A. Y. Toledano, S. Periaswamy, S. V. Fotin, J. Go, J. E. Boatsman, and J. W. Hoffmeister, “Improving Accuracy and Efficiency with Concurrent Use of Artificial Intelligence for Digital Breast Tomosynthesis,”
[17] G. V. Ionescu, M. Fergie, M. Berks, E. F. Harkness, J. Hulleman, A. R. Brentnall, J. Cuzick, D. G. Evans, and S. M. Astley, “Prediction of reader estimates of mammographic density using convolutional neural networks,” J. Med. Imaging
[18] https://doi.org/10.1088/1742-6596/1584/1/012005