الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدةد. مولاي الطاهر

كلية العلوم

قسم: الإعلام الآلي

Mémoire de Master

Spécialité : Réseaux Informatiques et Systèmes Répartis

Thème

Une évaluation des méthodes de présentation des données non image dans l'architecture CNN

Présenté par :

Bendjerad Zoulikha

Bendjerad Samira

Dirigé par :

Dr.Hassene Chaibi

Promotion 2023 - 2024

Remerciement

Nous remercions tout d'abord, ALLAH qui nous a donné la force et le courage pour élaborer ce modeste travail

La mise en oeuvre de ce mémoire est bien plus que la concrétisation d'efforts personnels mais le résultats d'efforts multiples que nous ne voulons pas occulter;ainsi nous tenons à exprimer notre gratitude envers notre encadreur le **Docteur Chaibi Hassene** dont les conseilles avisés ont éclairés le chemin qui nous a mener à la concrétisation de ce travail .

On tient aussi à remercier les membres du jury **Dr Latreche Abdelkrim** et **Dr Zerrouki Kada** pour avoir accepté d'examiner et d'évaluer ce travail.

Dédicace

On dédie ce modeste travail A la mémoire de nos parents A notre frère ,

À notre sœurs et à toute la famille.

A nos amis pour leurs soutiens et leurs encouragements .

À tous Nos enseignants.

A tous ceux qui nous ont aidées de prés ou de loin.

Zoulikha et Samira

Résumé

Avec l'évolution technologique et l'explosion des données massives (big data), les méthodes traditionnelles d'analyse des données, telles que la régression linéaire, les arbres de décision et les méthodes de clustering, sont confrontées à des défis sans précédent. Ces méthodes, bien qu'efficaces dans de nombreux contextes, montrent leurs limites lorsqu'elles doivent traiter des quantités massives de données, en particulier celles qui sont non structurées, complexes, riches en informations et caractérisées par une dimensionnalité élevée. En réponse à ces défis, les réseaux de neurones convolutionnels (CNN), qui ont démontré une efficacité remarquable dans la classification des données image, présentent un potentiel prometteur pour l'analyse des données non-image.

Ce mémoire se concentre sur l'évaluation des méthodes de présentation des données non-image dans l'architecture des CNN. En particulier, nous nous intéressons à l'application des CNN aux données génomiques, qui sont des exemples typiques de données de grande dimension et riches en informations. La transformation adéquate de ces données en une forme compatible avec les CNN est essentielle pour tirer parti des capacités de ces modèles.

Nous avons mené une série de tests pour évaluer les performances des CNN sur des bases de données génomiques. Nos expériences montrent que, malgré les défis inhérents à la transformation des données non-image en formats adaptés aux CNN, il est possible de réaliser des performances significatives, surpassant parfois celles des méthodes traditionnelles. Nos résultats mettent en lumière les avantages et les limitations de différentes approches de transformation des données, offrant des insights précieux pour les chercheurs et les praticiens dans le domaine.

En conclusion, ce mémoire fournit une analyse détaillée des techniques de présentation des données non-image pour les CNN, démontrant leur potentiel pour surmonter les limitations des méthodes traditionnelles d'analyse de données. Nous espérons que notre étude contribuera à une adoption plus large des CNN pour des applications de big data, ouvrant ainsi de nouvelles perspectives pour l'analyse des données complexes et de grande dimension.

Mots-clés : Big Data , Régression linéaire, , Arbres de décision, Clustering , CNN.

Abstract

With the technological evolution and explosion of big data (big data), traditional methods of data analysis, such as linear regression, decision trees and clustering methods, are facing unprecedented challenges. These methods, while effective in many contexts, show their limitations when dealing with massive amounts of data, especially those that are unstructured, complex, information-rich and characterized by high dimensionality. In response to these challenges, convolutional neural networks (CNN), which have demonstrated remarkable efficiency in classifying image data, have promising potential for the analysis of non-image data.

This thesis focuses on the evaluation of non-image data presentation methods in the CNN architecture. In particular, we are interested in the application of CNN to genomic data, which are typical examples of large and information-rich data. The proper transformation of this data into a form compatible with CNN is essential to leverage the capabilities of these models.

We conducted a series of tests to evaluate the performance of CNN on genomic databases. Our experiments show that, despite the challenges inherent in transforming non-image data into formats adapted to CNN , it is possible to achieve significant performance, sometimes surpassing those of traditional methods. Our results highlight the benefits and limitations of different data transformation approaches, providing valuable insights for researchers and practitioners in the field.

In conclusion, this thesis provides a detailed analysis of non-image data presentation techniques for CNN, demonstrating their potential to overcome the limitations of traditional data analysis methods. We hope that our study will contribute to a wider adoption of CNN for big data applications, thus opening up new perspectives for the analysis of large and complex data.

Keywords : Big data , linear regression, decision trees , clustering ,CNN .

ملخص

مع التطور التكنولوجي الهائل و كمية البيانات الضخمة أصبحت الأساليب التقليدية لتحليل البيانات مثل الانحدار الخطي وأشجار القرار وأساليب التجميع محدودة رغم فاعليتها و ذلك نظرا للعدد الهائل من البيانات الغنية بالمعلومات لا سيما غير المنظمة منها ، المعقدة و التي تتميز بأبعاد عالية . واستجابة لهذه التحديات أظهرت الشبكات العصبية التلافيفية كفاءة ملحوظة في تصنيف البيانات المصورة و بإمكانيات واعدة لتحليل البيانات غير المصورة.

تركز هذه الأطروحة على تقييم طرق عرض البيانات غير المصورة في بنية الشبكات العصبية التلافيفية و على وجه الخصوص، نحن مهتمون بتطبيق هذه الشبكات على البيانات الجينية و التي تعتبر أمثلة نموذجية للبيانات الضخمة والغنية بالمعلومات. بحيث يجب تحويل البيانات إلى شكل متوافق مع الشبكات العصبية التلافيفة للاستفادة من قدرات هذه النماذج.

أجرينا سلسلة من الاختبارات لتقييم أداء الشبكات العصبية التلافيفة على قواعد البيانات الجينية، و قد أظهرت تجاربنا أنه على الرغم من التحديات الكامنة في تحويل البيانات غير المصورة إلى تنسيقات مكيفة مع هذه الشبكات فمن الممكن تحقيق نتائج جيدة جدا قد تتجاوز تلك الخاصة بالطرق التقليدية .

سلسة التجارب و النتائج المتحصل عليها سلطت الضوء على فوائد وقيود مناهج تحويل البيانات المختلفة، و التي ستساهم لا شك في توضيح الرؤى و مساعدة الباحثين والممارسين في هذا المجال على تطوير و تحسين أداء هذه النماذج.

تقدم هذه الأطروحة تحليلاً مفصلاً لتقنيات عرض البيانات غير المصورة و التي أظهرت قدرتها على تحقيق نتائج ممتازة متجاوزة الطرق التقليدية لتحليل البيانات و تعقيداتها. نأمل أن تساهم دراستنا في اعتماد الشبكات العصبية التلافيفة على نطاق أوسع لتحليل البيانات الضخمة، وبالتالي فتح آفاق جديدة لتحليل البيانات الضخمة المعقدة و الغنية بالمعلومات . .

كلمات مفتاحية البيانات الضخمة ، الشبكات العصبية التلافيفية ، الانحدار الخطي، أشجار القرار، أساليب التجميع

Sommaire

Liste des Abréviations

CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
IoT	Internet Of Things
Relu	Rectified Linear Unit
MLP	Multilayer perceptron
DL	Deep Learning
ML	Machine Learning
ADN	Acide Désoxyribonucléique
TCGA	The Cancer Genome Atlas
Ada-Boost	Adaptive Boosting
BRCA	Breast Invasive Carcinoma
HNSC	Head And Neck Squamous Cell Carcinoma
KIRC	Kidney Renal Clear Cell Carcinoma
LGG	Brain Lower Grade Glioma
LUAD	Lung Adenocarcinoma
LUSC	Lung Squamous Cell Carcinoma
THCA	Thyroid Carcinoma

COAD	Colon Adenocarcinoma
OCR	Optical Character Recognition
API	Application Programming Interface
VGG16	Visual Geometry Group 16
RGB	Red, Green, Blue
MNIST	Modified National Institute of Standards and Technology
DNN	Deep Neural Network
FC	Fully Connected

Liste des tableaux

Liste des figures

Liste des figures

Liste des figures

Introduction Générale

1 Introduction

L'essor des réseaux de neurones convolutionnels (CNN) a révolutionné le domaine de la reconnaissance et de la classification des i mages. Leur capacité à extraire automatiquement des caractéristiques complexes et à apprendre des représentations hiérarchiques des données a conduit à des avancées significatives dans divers domaines, tels que la vision par ordinateur, la reconnaissance vocale et la traduction automatique. Cependant, malgré leur succès indéniable dans le traitement des données visuelles, l'application des CNN à des données non-image reste un domaine de recherche relativement peu exploré.

Les données non-image, telles que les séries temporelles, les données textuelles, les données tabulaires, et les signaux, présentent des défis uniques pour l'utilisation des CNN. Contrairement aux images, ces types de données ne possèdent pas de structure spatiale explicite que les convolutions peuvent facilement exploiter. Par conséquent, la transformation adéquate de ces données en une forme compatible avec les CNN est cruciale pour tirer parti de la puissance de ces modèles.

Plusieurs méthodes ont été proposées pour adapter les CNN aux données non-image. Parmi ces méthodes, la transformation des données en matrices ou en pseudo-images est couramment utilisée. Cette approche permet d'utiliser directement les architectures CNN existantes sans modification substantielle.

2 Objectif de l'étude :

Dans le cadre de ce mémoire, nous nous proposons d'évaluer et de comparer différentes méthodes de présentation des données non-image dans les architectures CNN. Nous examinerons comment ces transformations impactent les performances des modèles et quelles sont les meilleures pratiques pour différentes types de données n on-image. Notre objectif est de fournir une analyse approfondie des avantages et des limitations de chaque méthode, ainsi que des recommandations pour les praticiens souhaitant appliquer les CNN à des données non-image.

3 Aperçu de la structure du mémoire :

Pour ce faire, nous diviserons notre étude en plusieurs parties. Nous commencerons par une revue de la littérature sur les CNN et leur application aux données non-image. Ensuite, nous détaillerons les différentes méthodes de transformation des données, en expliquant leur fonctionnement et leurs implications pour l'apprentissage profond. Nous présenterons ensuite une série d'expériences empiriques pour évaluer les performances de ces méthodes sur divers jeux de données non-image.

Enfin, nous discuterons des résultats obtenus, des défis rencontrés et des perspectives futures pour l'application des CNN aux données non-image.

Ce mémoire vise à combler le fossé entre les applications traditionnelles des CNN aux données visuelles et leur potentiel inexploité pour les données non-image. En fournissant une évaluation rigoureuse des méthodes de transformation des données, nous espérons contribuer à une meilleure compréhension et à une adoption plus large des CNN dans des domaines où ils ne sont pas encore couramment utilisés.

CHAPITRE 01:

FONDEMENTS THEORIQUES

1.1 Introduction aux réseaux de neurones convolutifs(CNN)

Les réseaux de neurone convolutif sont des réseaux d'apprentissage profond inspirés du cortex visuel [Hubel& Wiesel 1962]. Ces réseaux ont été utilisés dans les systèmes de recommandation [Ying et al. 2018], en traitement du langage naturel [Kim 2014], et en vision par ordinateur [Krizhevsky et al. 2012]. Leur exploitation en vision par ordinateur a connu un grand succès grâce à leurs caractéristiques inspirées des systèmes visuels naturels.

En vision par ordinateur, le processus de classification par les méthodes d'apprentissage classiques est basé sur 2 étapes principales : l'extraction des caractéristiques et l'apprentissage. Ces caractéristiques sont considérées comme des handcrafted features à cause de l'effort manuel et nécessaire dans l'étude des attributs discriminants. Les méthodes utilisées extraient ces caractéristiques d'une manière non supervisée. Cette séparation entre les modules d'extraction et de classification peut nuire la tâche de classification si certains attributs discriminants ont été négligés dans la phase de l'extraction [5].

Contrairement aux méthodes d'apprentissage classiques, les CNN réalisent implicitement le processus d'extraction des caractéristiques à travers les couches de convolution, où les premières couches représentent les caractéristiques simples. Ensuite, ces caractéristiques sont combinées pour former d'autres qui sont plus complexes dans les couches profondes. Cette spécificité a rendu les CNN un bon outil pour la classification des données non structurées comme les images et les textes. Malgré ces avantages, les CNN profonds risquent le problème de sur-apprentissage sur les volumes limités de données, car ils sont plus adaptés aux grands volumes à cause du problème de sur-apprentissage [Keshari et al. 2018].

Les CNN ont prouvé leur efficacité par rapport aux réseaux de neurones profonds classiques en termes de complexité temporelle et spatiale. Ces réseaux sont caractérisés par leur stratégie de partage des paramètres. Contrairement à un DNN, où les couches adjacentes sont fortement connectées, dans un CNN, chaque neurone de la couche courante est connecté seulement à un sous ensemble de neurones de la couche précédente [2].

Les CNN sont considérés comme des méthodes de stabilisation structurelle qui permettent de réduire les problèmes de surapprentissage à travers l'optimisation du nombre des paramètres.

1.2 Historique

Les réseaux de neurones convolutifs sont inspirés du cortex visuel. Le cortex visuel est la partie de cerveau responsable du traitement des informations provenant de l'œil.

En 1962, les chercheurs [Hubel& Wiesel 1962] ont inséré des électrodes dans des parties spécifiques du cortex visuel d'un chat afin de mesurer l'activation lors que le chat observe quelques formes de base. Ils ont remarqué que les cellules simples répondent seulement aux barres horizontales en bas d'une image. Tandis que les cellules complexes sont caractérisées par une invariance spatiale, où ils peuvent répondre à ces barres dans différents emplacements dans l'image. Cette invariance est assurée par la combinaison des sorties des cellules simples [2].

En fonction de ces hypothèses, [Fukushima 1980] a développé un modèle (figure 1.1) composé de deux types de cellules neuronales : les cellules simples (S) et complexes (C). Les cellules S sont activées à la détection des formes basiques, tandis que les cellules C combinent les activations des cellules S [7]. L'idée est de transformer les concepts biologiques introduits précédemment à des concepts mathématiques pour modéliser la tâche de reconnaissance visuelle des formes. Ce modèle a été exploité pour la reconnaissance des formes dans une approche non supervisée.

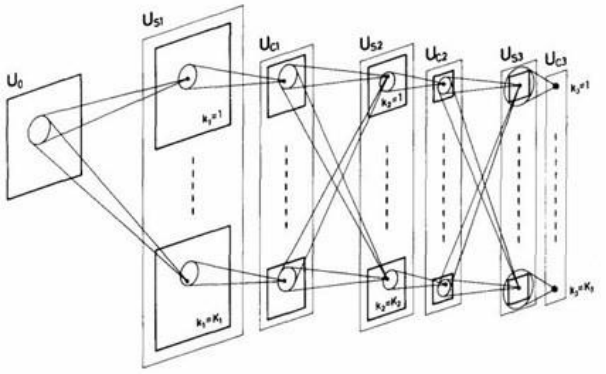

Figure 1.1: La structure du modèle proposé par[Fukushima 1980].

En 1998, [LeCun et al. 1998] ont introduit le réseau de neurones convolutif qui est basé sur l'architecture proposée par Fukushima, où ils ont exploité la méthode de rétropropagation afin d'accomplir une tâche de classification su pervisée. Leur modèle a été testé sur la base d'apprentissage MNIST spécialisée en classification des caractères manuscrits [7]. Au début des années 2000, la recherche sur les réseaux CNN a stagné à cause de la puissance insuffisante des processeurs et des capacités des mémoires internes limités pour les besoins de tels algorithmes. Durant cette période, les algorithmes d'apprentissage automatique classiques ont été largement exploités grâce à leurs exigences raisonnables en termes de complexité de calcul et espace de stockage [7]. En 2012, l'architecture AlexNet de type CNN a réussi le meilleur taux d'erreur dans l'état de l'art sur la base d'apprentissage ImageNet [Krizhevsky et al. 2012]. Cette bonne performance et la capacité des GPUs dans l'optimisation de la complexité temporelle ont encouragé la communauté de l'intelligence artificielle à proposer d'autres variantes optimisées de l'architecture CNN.

1.3 L'architecture générale d'un réseau de neurones convolutif

Un réseau de neurone convolutif est composé de trois types de couches : couche de convolution, couche de pooling et couche entièrement connectée. La figure 1.2 illustre l'architecture d'un CNN.

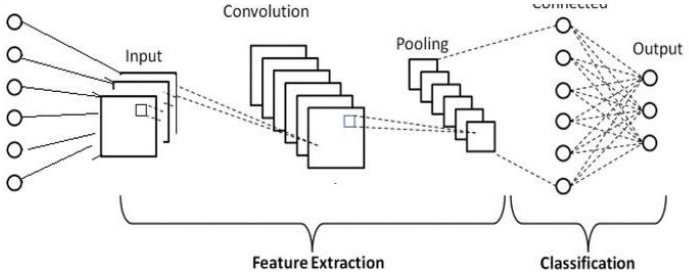

Figure 1.2: l'architecture générale d'un réseau de neurones convolutif

1.3.1 Couche de convolution

La couche de convolution est définie par le bloc de construction principal d'un CNN. Le but de cette couche est d'extraire implicitement les caractéristiques pertinentes des images en entrée durant l'apprentissage. Cette couche effectue une opération de convolution entre deux matrices, la première représente une sous partie des données en entrée (champ réceptif) et la deuxième représente un filtre qui contient les paramètres d'apprentissage. Une opération de convolution génère une troisième matrice référencée par la carte des caractéristiques. La figure 1.3 illustre une opération d'une convolution qui est réalisée par un produit scalaire entre le filtre et un champ réceptif. Ensuite, les résultats du produit sont additionnés pour produire un seul résultat présenté sous forme d'une case dans la carte des caractéristiques. Enfin, le filtre des poids est glissé par un pas S sur le reste des champs réceptifs de la matrice en entrée, et cette opération est répétée pour tous les autres champs [7].

Dans une convolution, la taille de la nouvelle carte des caractéristiques N(t+1) est calculée en fonction de 4 hyper-paramètres : la taille de l'ancienne carte des caractéristiques ou la matrice en entrée N(t) , la taille du filtre F, la valeur du pas S et la valeur de la marge P (équation 1.1).

La marge représente des valeurs nulles qui entourent la matrice en entrée. Cette marge empêche le filtre de dépasser le cadre de cette matrice. L'application de Nc filtres sur les données en entrées résulte une carte des caractéristiques d'une taille de N(t+1) × N(t+1) × Nc , où Nc est sa profondeur [7]. La concaténation des cartes des caractéristiques forme une couche de convolution.

$$N^{t+1} = \frac{N^t - F + 2P}{S} - 1 \qquad (1.1)$$

S

Le processus d'une convolution illustre la stratégie de réduction de dimensionnalité d'un CNN, où chaque case (neurone) de la carte de caractéristique courante est connectée seulement à un sous ensemble des neurones en entrées (champs récepteurs). En plus, l'application du même filtre sur toute la carte des caractéristiques lui permet de découvrir les attributs précédemment détectés dans différentes zones de l'image.

À la fin de chaque opération de convolution, la fonction d'activation ReLu est appliquée sur la couche de convolution résultante afin d'améliorer la généralisation [7].

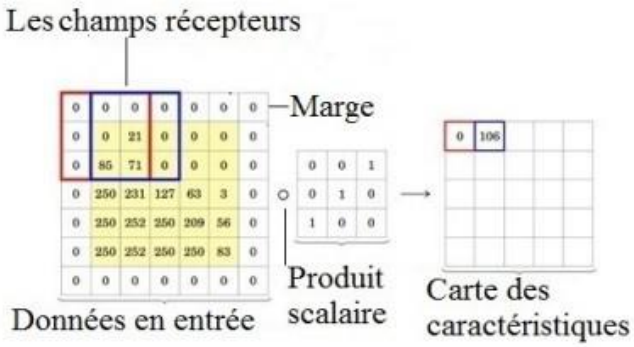

Figure 1.3: Une opération de convolution.

1.3.2 Couche de pooling

Le rôle de la couche de pooling est de réduire la dimensionnalité des couches de convolution résultantes. Le but de cette réduction est d'améliorer la précision par la sélection des attributs dominants. En plus, l'optimisation du nombre des paramètres permet de réduire la taille du modèle et d'optimiser la complexité temporelle [7].

La taille de la matrice résultante de l'opération de pooling est calculée par l'équation 1.1 avec P = 0. Il existe deux types d'opérations pooling : Max-pooling et Avg-pooling. L'opération de Max-pooling renvoie la valeur maximale du champ réceptif tandis que l'opération Avg-pooling renvoie la moyenne des valeurs. Max-pooling est la forme la plus utilisée dans la majorité des architectures de type CNN (figure 1.4) [7]

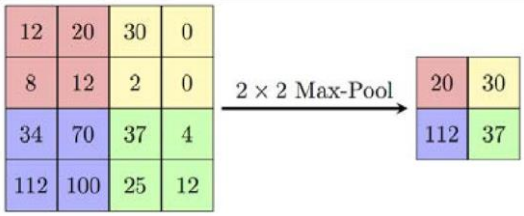

Figure 1.4: Une opération de Max-pooling.

1.3.3 Couche entièrement connectée

Dans un CNN, les couches entièrement connectées (FC) ont la même structure qu'un MLP. Le but de ces couches est d'apprendre les combinaisons non linéaires entre les caractéristiques extraites par les couches de convolution. Le résultat de la dernière couche de convolution [N, N, Nc] est aplatit dans un vecteur de taille [N × N × Nc]. Ce vecteur présente la couche d'entrée à l'ensemble des couches entièrement connectées. En classification s upervisée, l a d ernière couche est utilisée pour la prédiction en se basant sur la fonction d'activation Softmax [7].

1.4 Les types d'apprentissage

L'apprentissage dans un réseau de neurones convolutif peut être effectué de deux façons apprentissage à partir des paramètres initialisés aléatoirement (training from scratch) ou apprentissage par transfert [10].

1.4.1 Apprentissage à partir des initialisations aléatoires

Les réseaux de neurones convolutif sont basés sur la méthode de rétropropagation. Le but principal des CNN est de réduire l'erreur de la fonction du coût par l'ajustement des filtres. Ces filtres représentent les paramètres w de l'apprentissage. Comme nous l'avons mentionné précédemment, les CNN sont caractérisés par le partage des paramètres. Cette spécificité permet de réduire le nombre des paramètres dans une couche de convolution et d'optimiser la complexité temporelle et spatiale. Ce partage et la représentation des paramètres dans des filtres exigent d'adapter la fonction de rétropropagation sur les couches de convolution et de pooling [10].

L'apprentissage dans un CNN commence par une propagation vers l'avant pour calculer la valeur de la fonction du coût en fonction des entrées. Ensuite, les filtres i nitialisés aléatoirement sont ajustés par un processus de rétropropagation. Ce processus est répété pour un certain nombre d'itérations jusqu'à atteindre le critère d'arrêt. Le critère peut dépendre d'un nombre fixe d'itérations, arrêt prématuré, ou une convergence [10].

1.4.2 Apprentissage par transfert et fine-tuning

L'apprentissage par transfert est une méthode d'apprentissage automatique qui permet de réutiliser un modèle développé précédemment pour l'apprentissage d'une tâche A dans une autre tâche B. Ces tâches peuvent être similaires ou différentes selon la nature du processus de l'apprentissage transféré [10].

La recherche en apprentissage transféré a commencé depuis les années 1995, où ils étaient inspirés du comportement des humains dans leur méthode d'apprentissage basée sur la connaissance acquise précédemment. Ces méthodes sont catégorisées en : apprentissage par transfert inductif, tranductif, et non supervisé. Dans la méthode de transfert inductif, les tâches source et cible sont différentes et appartiennent au même domaine. Tandis qu'en apprentissage tranductif, les tâches sont similaires et différentes dans la distribution des probabilités dans l'espace des attributs. En apprentissage non supervisé, les données de l'apprentissage ne sont pas catégorisées dans les domaines source et cible [10].

Les méthodes de l'apprentissage par transfert en apprentissage profond appartiennent à la catégorie de l'apprentissage inductif. Elles sont exploitées dans différents domaines comme le traitement du langage naturel et la vision par ordinateur [10].

En apprentissage automatique classique, les algorithmes d'apprentissage sont caractérisés par leur dépendance de la distribution des attributs en entrée, où les données source et cible doivent avoir la même représentation des données. Le changement de la distribution des attributs exige de reprendre l'apprentissage à partir du début. Contrairement à ces algorithmes, en apprentissage profond, il est possible de transférer la connaissance à partir des modèles formés précédemment,

où la distribution des paramètres (poids) des réseaux DL offre cette possibilité. En vision par ordinateur, l'apprentissage par transfert à partir des réseaux CNN a connu un grand succès grâce à leur nature hiérarchique. Les premières couches représentent des caractéristiques générales comme les filtres de Gabor. Ils permettent de détecter des formes basiques (courbes et bordures). En revanche, les couches profondes permettent de modéliser des caractéristiques plus complexes et liées au domaine d'application de la base d'apprentissage. Les caractéristiques communes des premières couches offre la possibilité d'effectuer un apprentissage transféré entre différentes tâches [10].

L'apprentissage par transfert est l'une des méthodes proposées dans l'état de l'art pour résoudre cette limitation. Cette technique est généralement utilisée quand le volume de données de la tâche cible est limité. Elle permet de réutiliser les premières couches des modèles générés à partir des grands volumes de données. L'apprentissage par transfert à partir des modèles entrainés sur la base d'apprentissage ImageNet a été largement exploité dans différents domaines grâce à son grand volume de données (15 millions) et son nombre important de catégories (22 000) [10].

L'apprentissage par transfert en CNN peut être utilisé sous forme de trois méthodes : (a) l'exploitation du modèle source comme un module d'extraction de caractéristiques, (b) transférer un sous ensemble de couches et réajuster le reste, et (c) transférer et réajuster tous les couches.

1.4.2.1 Module d'extraction des caractéristiques

Cette stratégie est considérée comme une hybridation entre les algorithmes d'apprentissage automatique et les réseaux DL, où des réseaux de type CNN sont exploités pour l'extraction des caractéristiques et les algorithmes ML pour la classification. La figure 1.5 illustre ce processus, généralement, toutes les couches du modèle source sont transférées au modèle cible. Ensuite, la dernière couche est supprimée. Les données de la base d'apprentissage cible sont passées ensuite au modèle. Cela permet de générer une base d'apprentissage structurée sous forme d'attributs et d'instances. Enfin, cette base est classifiée par un algorithme de type ML [7].

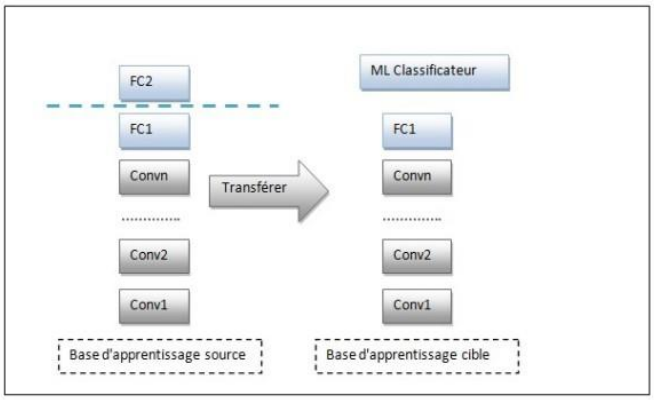

Figure 1.5: L'utilisation des réseaux de neurones convolutifs pour l'extraction des caractéristiques.

1.4.2.2 Fine Tuning

Comme nous l'avons discuté auparavant, les premières couches permettent de représenter des caractéristiques générales, tandis que les couches profondes sont liées au domaine d'application source. Afin d'adapter ces couches au domaine d'application cible, un sous ensemble de couches profondes (convolutions et entièrement connecté) est réajusté par un processus d'apprentissage (figure 1.6) [7].

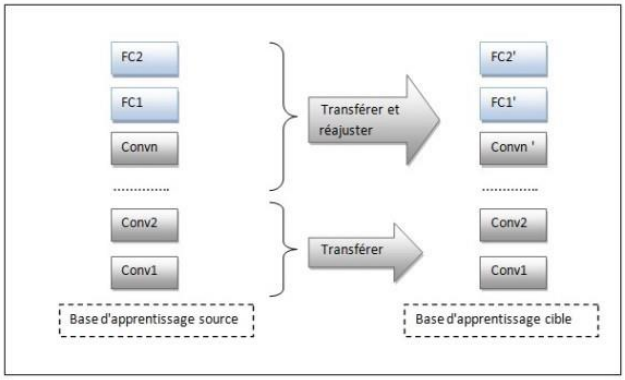

Figure 1.6: Le processus de fine-tuning dans un réseau de neurones convolutif.

1.5 Revue de la littérature sur l 'utilisation des CNN pour la classification des données non images

Il existe une quantité croissante de recherches sur l'utilisation des réseaux de neurones convolutionnels (CNN) pour la classification de données non images. Voici un aperçu de certaines études et de leurs résultats significatifs [11] :

1. Zhang et al. (2015) :

L'étude de Zhang et al. (2015) a proposé une méthode novatrice pour utiliser des réseaux de neurones convolutionnels (CNN) dans la classification de séquences temporelles. Leur approche consistait à appliquer des filtres convolutifs pour extraire des motifs discriminatifs à partir des séries temporelles, permettant ainsi de capturer les relations temporelles complexes dans les données [11].

Les résultats de leur étude ont démontré d'excellentes performances de leur approche pour la prédiction de séries temporelles. En exploitant la capacité des CNN à apprendre des caractéristiques hiérarchiques à différents niveaux d'abstraction, ils ont pu obtenir des modèles de classification robustes et précis pour les données séquentielles. Cette approche a ouvert de nouvelles perspectives dans l'application des CNN aux données non-images, en particulier pour la classification de séquences temporelles. Les résultats prometteurs obtenus par Zhang et al.(2015) ont contribué à stimuler la recherche dans ce domaine et ont montré le potentiel des CNN pour traiter efficacement ce type de données [11].

2. Conneau et al. (2016) :

L'étude de Conneau et al. (2016) a présenté un modèle novateur de réseaux de neurones convolutionnels (CNN) pour la classification de texte. Ce modèle exploitait des représentations vectorielles de mots pré-entraînées pour améliorer la performance de la classification de textes courts [11].

En utilisant des représentations vectorielles de mots pré-entraînées, le modèle de CNN de Conneau et al. a pu capturer des informations sémantiques et syntaxiques riches des mots, ce qui a permis d'améliorer la compréhension du texte par le modèle.

Les résultats de leur étude ont montré que leur modèle de CNN dépassait les approches traditionnelles pour la classification de textes courts. En exploitant efficacement les représentations vectorielles de mots et en utilisant des techniques de CNN adaptées au traitement du langage naturel, leur modèle a pu obtenir des performances supérieures dans la classification de texte [11].

3. Lee et al. (2017) :

L'étude de Lee et al. (2017) a exploré l'application des réseaux de neurones convolutionnels (CNN) à la classification de signaux audio. Leur approche consistait à transformer les formes d'ondes audio en images spectrogrammes, permettant ainsi d'appliquer les CNN données audio de manière efficace [11].

En convertissant les signaux audio en spectrogrammes, qui sont des représentations visuelles des fréquences temporelles des signaux audio, les chercheurs ont pu exploiter les capacités des CNN pour traiter des données sous forme d'images. Cela leur a permis d'appliquer des opérations de convolution et de pooling pour extraire des caractéristiques pertinentes des spectrogrammes[11]

Les résultats de leur étude ont montré des performances prometteuses pour la classification de signaux audio en utilisant des CNN sur les spectrogrammes. Cette approche a ouvert de nouvelles perspectives dans l'analyse et la classification de données audio, avec des applications potentielles dans la reconnaissance vocale, la musique, la surveillance acoustique,etc [11].

4. Tang et al. (2018) :

L'étude de Tang et al. (2018) a investigué l'utilisation des réseaux de neurones convolutionnels (CNN) pour la classification de données génomiques. Dans leur recherche, ils ont utilisé des séquences d'ADN comme entrées pour prédire des interactions protéine-protéine, une tâche importante en bioinformatique [11].

En appliquant des CNN aux séquences d'ADN, les chercheurs ont pu capturer des motifs et des relations complexes dans les données génomiques, ce qui a permis à leur modèle de prédire avec précision les interactions protéine-protéine Les résultats de leur étude ont montré que le modèle CNN présentait de bonnes performances pour la prédiction des interactions entre protéines. En exploitant efficacement les caractéristiques des séquences d'ADN à l'aide des CNN, leur approche s'est avérée prometteuse pour la prédiction et la compréhension des ces interactions[11].

1.6 limitation des approches existantes et besoins pour de nouvelles méthodes

Malgré les progrès réalisés dans l'utilisation des réseaux de neurones convolutionnels(CNN)pour la classification de données non images, certaines limitations persistent, ce qui soulève la nécessité de nouvelles méthodes et approches[13] :

1.Interprétabilité :

Les modèles CNN sont souvent considérés comme des "boîtes noires" en raison de leur complexité. Il est difficile d'expliquer les prédictions et les décisions prises par ces modèles, ce qui peut poser des problèmes en termes de confiance et d'acceptation dans certaines applications critiques [13].

2.Taille des données et surapprentissage :

Les CNN peuvent nécessiter des ensembles de données massifs pour être efficaces, ce qui peut être contraignant dans des domaines où les données sont rares. De plus, les CNN peuvent être sujets au surapprentissage, en particulier lorsque les données d'entraînement sont limitées [13].

3.Transfert de connaissances:

Les CNN nécessitent souvent une quantité importante de données pour l'entraînement. Il est donc important de développer des techniques de transfert de connaissances pour tirer parti des modèles pré-entraînés sur des tâches similaires et les adapter à de nouveaux ensembles de données plus petits [13].

4.Adaptabilité aux différentes structures de données :

Les CNN ont été initialement conçus pour traiter des données spatiales comme les images. Il est nécessaire de développer des architectures CNN adaptées à des types de données non structurées tels que des séquences temporelles, du texte ou des données génomiques [13].

5.Interopérabilité et généricité:

Il est important de concevoir des modèles CNN qui puissent être utilisés de manière efficace et générale, indépendamment du domaine d'application spécifique, tout en garantissant des performances optimales.

Pour adresser ces limitations, de nouvelles approches sont nécessaires, telles que le développement de modèles CNN plus interprétables, l'exploration de techniques d'apprentissage

par transfert plus efficaces, la conception de CNN adaptés à des types de données spécifiques et l'intégration de notions de robustesse et de fiabilité dans les modèles CNN. Ces avancées pourraient ouvrir de nouvelles perspectives pour l'utilisation des CNN dans la classification de données non images [13].

1.7 Conclusion :

Ce chapitre mis en évidence les réseaux de neurones convolutionnels CNN, leurs structure et leurs utilisation dans la classification des données notamment les données non image . La représentation des données non-images dans les CNN pose plusieurs défis en termes de conversion des données dans un format adapté aux CNN.

CHAPITRE 02:

METHODOLOGIE PROPOSEE

2.1 Introduction

Après avoir décrit théoriquement dans le chapitre précédent les réseaux de neurones CNN et les principes de classification des données non images, on va essayer de présenter les données de différentes manières on se concentrant sur la visualisation de l'aspect des images fournies à notre modèle CNN .

2.2 Présentation de la génération de pseudo-images

La génération de pseudo-images dans un réseau de neurones convolutionnel (CNN) est une technique permettant de créer des images synthétiques à partir de données non-image. Cette méthode vise à transformer des vecteurs de données en images qui peuvent être traitées par des CNN de manière similaire aux images réelles.

Le processus commence par la conversion des vecteurs en matrices de taille (H*L). Ensuite, ces matrices doivent être adaptées aux dimensions d'entrée du modèle d'apprentissage. Les tailles courantes pour ces images sont (32*32), (64*64), ou (128*128) pixels.

Pour que les pseudo-images soient compatibles avec le modèle, un redimensionnement est nécessaire. Ce redimensionnement peut se faire de deux manières :

1. **Avec rééchantillonnage :** Lorsque le rééchantillonnage est activé, l'image est redimen-sionnée en utilisant des techniques telles que l'interpolation bicubique. Cela permet de maintenir la qualité et les détails de l'image même après le redimensionnement.

2. **Sans rééchantillonnage :** Si le rééchantillonnage est désactivé, l'image est simplement recadrée aux dimensions souhaitées sans ajustement supplémentaire. Cela peut être plus rapide mais peut aussi entraîner une perte de qualité ou de détails.

Les tableaux 2.1 et 2.2 présente les différents cadres d'image utilisés lors des testes.

Test	Cadre
1	1x19321
2	400x50
3	1x19321
4	142x142
5	1x19321
6	19321x1
7	400x50
8	500x40

Test	Cadre
1	1x19321
2	133x50
3	81x81
4	1x19321
5	133x50
6	81x81
7	1x19321
8	133x50
9	81x81

Tableau 2.1 : Différents cadres d'images utilisés jeu de données curatedTCGAData.

Test	Cadre
1	1x5000
2	5000x1
3	50x100
4	100x50
5	250x25
6	71x71
7	1x5000
8	5000x1
9	50x100
10	100x50
11	250x25
12	71x71
13	1x5000
14	5000x1
15	50x100
16	100x50
17	250x25
18	71x71

Tableau 2.2 : Différents cadres d'images utilisés jeu de données TCGA.rnaseq et gisette

La figure 2.1 présente une taille d'image 32 avec deux différents cadres (50x100) et (1x5000).
La figure 2.2 présente une taille d'image 64 avec deux différents cadres (71x71) et (250x25).
La figure 2.3 présente une taille d'image 128 avec deux différents cadres (142x142) et (19321x1).

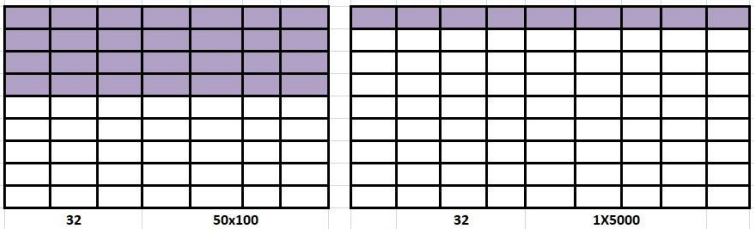

Figure 2.1: taille d'image 32 avec différents cadres d'images.

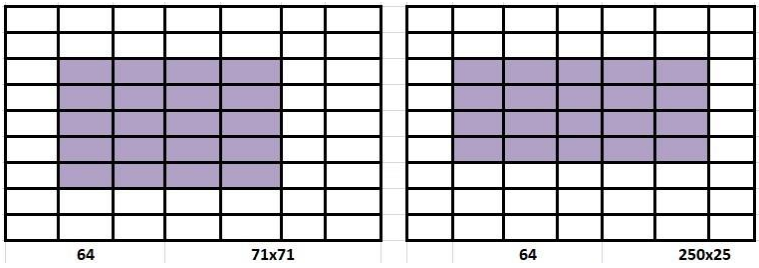

Figure 2.2: taille d'image 64 avec différents cadres d'images.

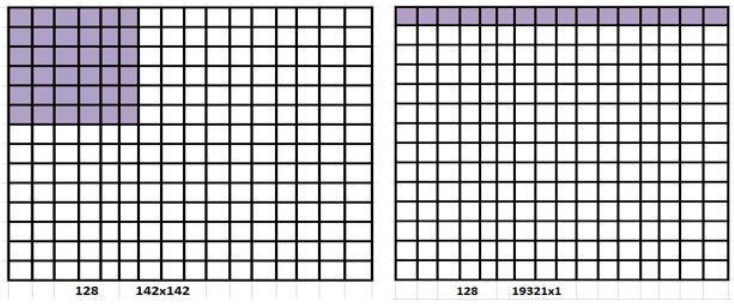

Figure 2.3: taille d'image 128 avec différents cadres d'images.

Voici quelque exemple d'images obtenues :

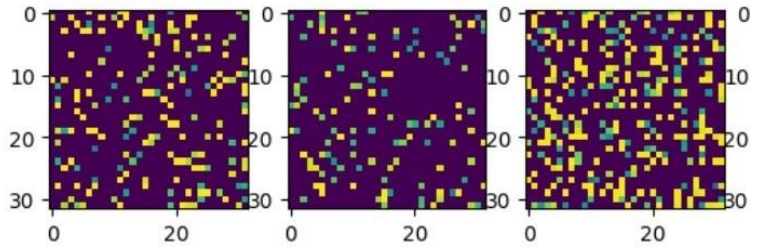

Figure 2.4: image de taille 32 cadre (100x50) resamp= false (dataset gisette) .

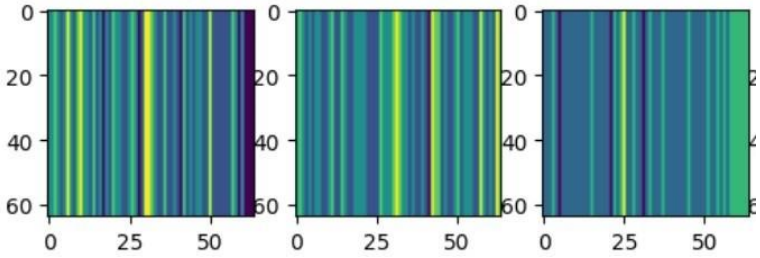

Figure 2.5: image de taille 64 cadre (1x19321) resamp=true (dataset curatedTCGAData) .

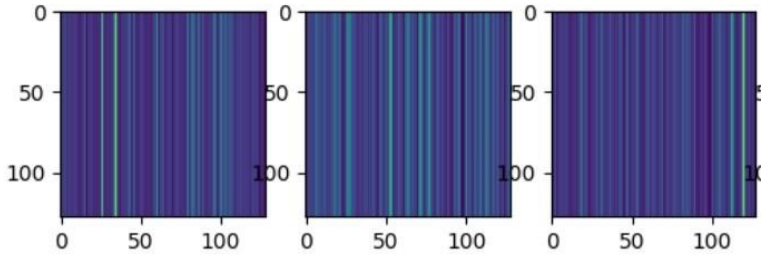

Figure 2.6: image de taille 128 cadre (1x5000) resamp=False (dataset TCGA.rnaseq) .

La figure 2.4 présente les images obtenues avec une taille d'image 32 , un cadre (100x50) et un rééchantillonnage resamp =false (le jeu de données utilisé dans le test c'est Gisette). Dans la figure 2.5 la taille d'image était 64 , un cadre (1x19321) et un rééchantillonnage resamp=true(le jeu de données utilisé dans le test c'est curatedTCGAData) . pour la figure 2.6 on a une taille d'image 128 , un cadre (1x5000) et un rééchantillonnage resamp=false (le jeu de données utilisé dans le test c'est TCGA.rnaseq).

- ✓ **Génération d'image RGB**

Lors des essais, nous avons également généré des images en couleur (RGB). Le principe de cette génération est de créer une image couleur à partir de trois matrices distinctes, représentant les canaux rouge (R), vert (G) et bleu (B).

1. Création des matrices RGB : Les données d'entrée, sous forme de vecteurs, sont réparties sur ces trois matrices. Chaque valeur de ces matrices est normalisée pour se situer dans une plage de 0 à 255, correspondant aux niveaux d'intensité des couleurs.

2. Empilement des matrices : Une fois les matrices normalisées, elles sont empilées ensemble pour former un tableau RGB. Ce tableau combine les informations des trois canaux de couleur, créant ainsi une représentation complète de l'image couleur.
3. Génération et redimensionnement de l'image : À partir de ce tableau RGB, une image est crée. Ensuite, pour s'assurer que l'image est compatible avec les dimensions d'entrée du modèle CNN, elle est redimensionnée à la taille spécifiée (par exemple, 32x32, 64 x 64 ou 128x128 pixels). Ce redimensionnement peut utiliser une méthode de rééchantillonnage, telle que l'interpolation bicubique, pour maintenir la qualité et les détails de l'image, ou être effectué sans rééchantillonnage pour une conversion plus rapide.

Voici quelque exemple d'images obtenues :

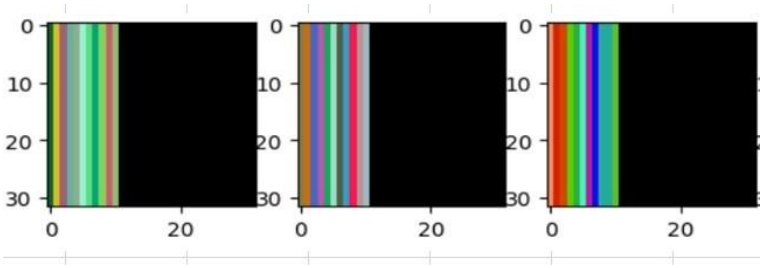

Figure 2.7 : image de taille 32 cadre (1x19321) resamp=false (dataset curatedTCGAData)

.

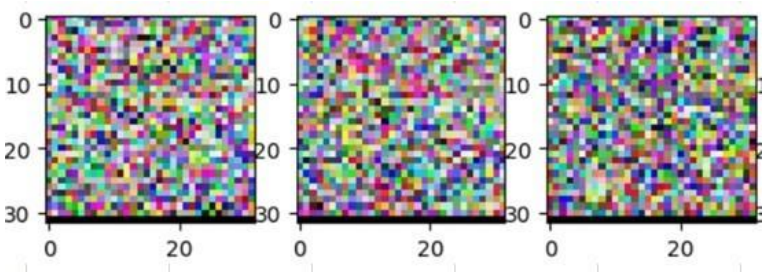

Figure 2.8 : image de taille 32 cadre (133x50) resamp=false (dataset curatedTCGAData) .

La figure 2.7 présente les images RGB obtenues avec une taille d'image32, un cadre (1x19321) et un rééchantillonnage resamp =false (le jeu de données utilisé dans le test c'est curatedTCGAData).

Dans la figure 2.8 la taille d'image était 32, un cadre (133x50) et un rééchantillonnage resamp=true (le jeu de données utilisé dans le test c'est curatedTCGAData) .

pour la figure 2.10 on a une taille d'image 128, un cadre (50x100) et un rééchantillonnage resamp=false (le jeu de données utilisé dans le test c'est Gisette).

Exemple sur rééchantillonnage :

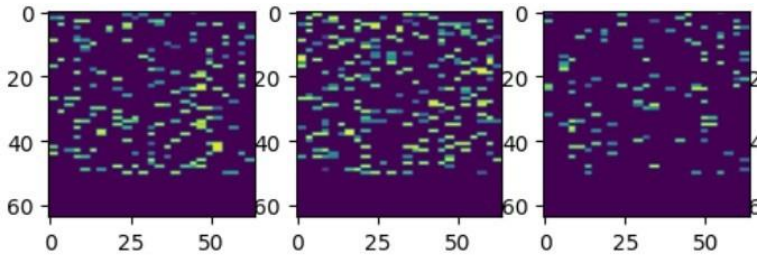

Figure 2.9 : image de taille 64 cadre (250x25) resamp= false (dataset Gisette)

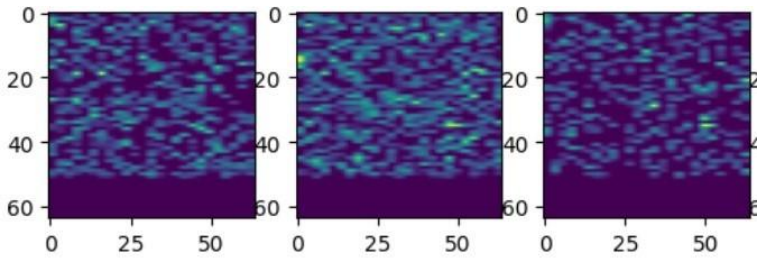

Figure 2.10 : image de taille 64 cadre (250x25) resamp= True (dataset Gisette) .

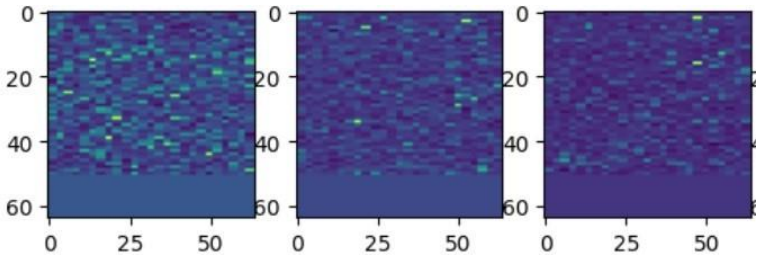

Figure 2.11 : image de taille 64 cadre (250x25) resamp=False (dataset TCGA.rnaseq) .

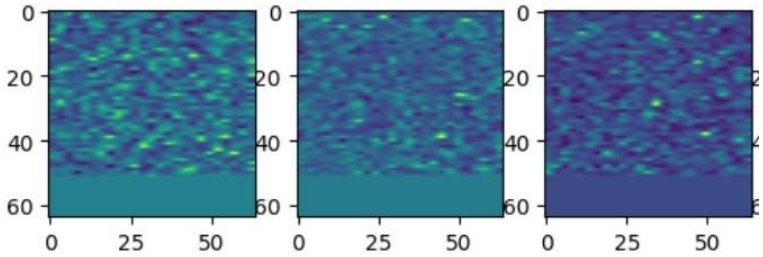

Figure 2.12 : image de taille 64 cadre (250x25) resamp=True (dataset TCGA.rnaseq) .

2.3 Description de l'architecture CNN adaptée à la classification des données non image

L'architecture du réseau de neurones conventionnel CNN est une amélioration de l'architecture de réseau de neurones standard. Dans cette dernière les données sont d'abord transmises à la couche d'entrée, puis elles se déplacent vers le niveau caché et en sortent. Les couches sont entièrement connectées et il n'y a pas de connexion entre les nœuds de la même couche [11].

Le CNN a obtenu des résultats remarquables dans des domaines tels que la classification des images et l'analyse du langage [4].

La structure d'un CNN :

(1) Une ou plusieurs couches convolutives

(2) Regroupement des couches en haut

(3) Couches entièrement connectées

(4) Couches de décrochage servant de couches de régularisation

Avec cette structure, le CNN peut tirer parti de la structure bidimensionnelle des données d'entrée (Le réseau peut utiliser une image comme entrée).
Ainsi, nous évitons l'extraction de fonctionnalités compliquées et la reconstruction de données inutiles des algorithmes de reconnaissance traditionnels. La puissance de modélisation peut être augmentée et de sorte que le niveau de difficulté du traitement manuel accru des données peut être réduit par le regroupement, le partage des poids et la connectivité peu fréquente. [3]CNN peut apprendre d'une grande quantité de données non étiquetées à différents niveaux de fonctionnalité. Par conséquent, les utilisations possibles de CNN dans des domaines différents sont polyvalentes [3].

L'architecture d'une méthode similaire à celle proposée est illustrée à la figure 3.7

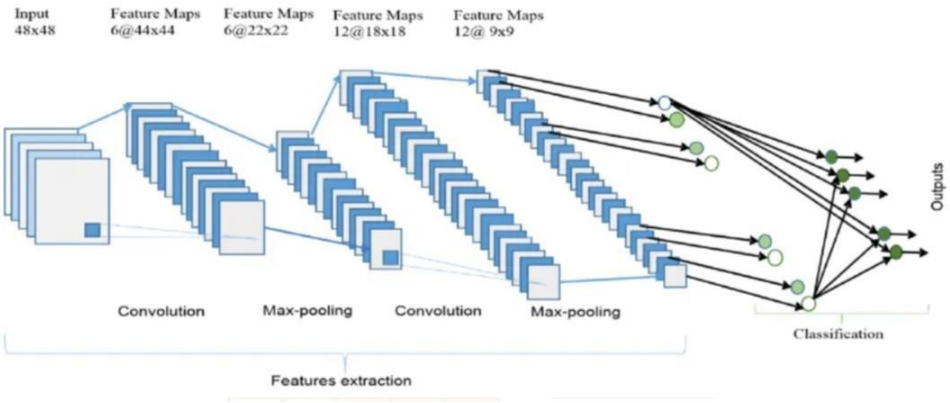

Figure 2.13 : Exemple d'architecture d'un model séquentiel CNN .

Dans la programmation Python, le type séquentiel est le type de modèle le plus couramment utilisé. C'est le moyen le plus simple de créer un modèle CNN dans Keras. Cela nous permet de construire le modèle couche par couche. Les couches sont ajoutées au modèle à l'aide de la fonction add (). Pour ce modèle, il y a trois Convolution et trois Pooling suivis d'un calque Flatten qui est généralement utilisé comme connexion entre Convolution et les couches Dense. Les couches Dense sont souvent utilisées pour les couches de sortie. L'activation employée est « Softmax » qui donne la probabilité pour chaque classe et donne un total de 1. Le modèle fait une prédiction basée sur la classe avec la probabilité la plus élevée [17].

Le résumé du modèle est affiché dans le tableau 3.3 ci-dessous.

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 30, 30, 32)	320
max_pooling2d_12 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_13 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_13 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_14 (Conv2D)	(None, 4, 4, 128)	73856
max_pooling2d_14 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten_4 (Flatten)	(None, 512)	0
dense_8 (Dense)	(None, 128)	65664
dense_9 (Dense)	(None, 7)	903

Total params: 159239 (622.03 KB)
Trainable params: 159239 (622.03 KB)
Non-trainable params: 0 (0.00 Byte)

Tableau 2.3 : résumé du modèle CNN

2.4 Explication des techniques de prétraitement et de transformation des données

Le prétraitement des données est une étape essentielle du deep learning pour le Big Data. Cela implique de nettoyer, transformer et organiser les données avant de les alimenter dans un modèle d'apprentissage en profondeur. Cette étape est cruciale car elle peut avoir un impact significatif sur la précision et l'efficacité du modèle [3].

L'un des principaux objectifs du prétraitement des données est d'améliorer la qualité des données. Cela implique d'identifier et de supprimer toutes les valeurs aberrantes, les valeurs manquantes ou les données bruitées qui pourraient affecter négativement les performances du modèle. Par exemple, en reconnaissance d'images, si certaines images sont floues ou ont une faible résolution, elles peuvent ne pas être utiles pour le modèle. En supprimant ces points de données,

nous pouvons améliorer la qualité globale de l'ensemble de données et garantir que le modèle est formé sur des données de haute qualité [21].

2.4.1 Les techniques de prétraitement et de transformation des données

Le prétraitement et la transformation des données non images dans un réseau de neurones convolutionnels (CNN) nécessitent des étapes spécifiques pour adapter ces données à l'architecture et aux exigences du CNN [11]. Voici les étapes générales à suivre pour prétraiter et transformer des données non images dans un CNN [12] :

1. **Lecture des données :** Charger et lire les données non images à partir de leur source, qu'il s'agisse de texte, de séquences numériques, de signaux, etc.

2. **Vectorisation des données** : Convertir les données non structurées en vecteurs pour les rendre compatibles avec les modèles CNN .Par exemple, pour le texte on peut utiliser des techniques comme la vectorisation de mots (Word Embeddings) pour représenter les mots sous forme de vecteurs [11].

3. **Normalisation :** Mettre à l'échelle les données numériques pour les ramener à une plage standard, par exemple en utilisant la normalisation Min-Max ou la normalisation Z-score. Cela peut améliorer la convergence du modèle et la performance globale [7].

4. **Réduction de dimensionnalité** : Pour les données de haute dimensionnalité, il peut être utile d'appliquer des techniques de réduction de dimensionnalité [11].

5. **One-Hot Encoding** : Pour les données catégorielles, appliquer le codage one-hot pour représenter les catégories sous forme de vecteurs binaires [11].

6. **Traitement des valeurs manquantes** : Gérer les valeurs manquantes de manière appropriée en utilisant des techniques comme l'imputation de données ou en les traitant comme des valeurs spéciales [11].

7. **Transformation en format d'entrée du CNN** : Adapter les données prétraitées à la forme d'entrée attendue par le CNN, qui est généralement un tensor en 3D (hauteur, largeur, canaux) pour les images [5]. Pour les données non images, cela peut nécessiter une transformation appropriée en fonction de leur structure.

8. **Augmentation des données (si nécessaire)** : Pour augmenter la taille de l'ensemble de données et améliorer la généralisation du modèle [10], appliquer des transformations telles que la rotation, le retournement, le recadrage, etc ., aux données non images si cela est pertinent.

Il est important de noter que l'utilisation de CNN pour des données non images peut nécessiter des adaptations spécifiques en fonction du type de données et de la tâche à réaliser.

En résumé, le prétraitement et la transformation des données non images dans un réseau de neurones convolutionnels requièrent une attention particulière pour adapter les données à l'architecture du CNN et garantir des performances optimales dans des tâches de classification, de prédiction ou d'analyse de données non images.

2.4.2 Etapes d'entrainement du modèle CNN et évaluation des performances

L'entraînement et l'évaluation d'un modèle CNN pour des données non images impliquent plusieurs étapes spécifiques pour garantir de bonnes performances. Voici les étapes générales à suivre pour entraîner et évaluer un modèle CNN sur des données non images :

1. **Prétraitement des données :**Prétraitez les données non images en utilisant des techniques telles que l'encodage des textes, la normalisation des données numériques, la transformation des séquences, etc., pour les adapter à l'architecture du CNN [3].

2. **Construction du modèle CNN :** Concevez l'architecture du modèle CNN en tenant compte de la nature des données non images et en choisissant les couches appropriées pour représenter et extraire les caractéristiques pertinentes des données [7].

3. **Entraînement du modèle :** Divisez les données en ensembles d'entraînement et de validation. Entraînez le modèle CNN sur les données d'entraînement en utilisant des techniques d'optimisation telles que la rétropropagation du gradient pour ajuster les poids du réseau [3].

4. **Évaluation du modèle :** Évaluez les performances du modèle sur l'ensemble de validation en mesurant des métriques telles que la précision, le rappel, la F1-score, etc. Analysez les résultats pour identifier d'éventuels problèmes de surajustement ou de sous-ajustement du modèle [3].

5. **Optimisation du modèle :**Optimisez les hyperparamètres du modèle CNN, tels que le taux d'apprentissage, la taille du lot, le nombre d'époch, etc., pour améliorer les performances du modèle [10].

6. **Inférence et prédiction :**Une fois le modèle entraîné et évalué, utilisez-le pour effectuer des prédictions sur de nouvelles données non images et évaluer sa capacité à généraliser.

En suivant ces étapes et en ajustant les paramètres du modèle CNN en fonction des caractéristiques des données non images, il est possible d'obtenir un modèle performant et précis pour des tâches de classification, de prédiction ou d'analyse de données non images [11].

2.5 Conclusion

La transformation des données non images dans un CNN est essentielle pour garantir des performances optimales du modèle, améliorer sa capacité à généraliser, et permettre une extraction efficace de s ca ractéristiques pe rtinentes de s do nnées no n im ages. En adaptant les techniques de prétraitement en fonction des caractéristiques spécifiques des données non images, on peut maximiser l'efficacité et la précision du modèle CNN dans des tâches variées de classification, de prédiction ou d'analyse de données non images.

CHAPITRE 03:

Mise en œuvre, résultats et discussion

3.1 Introduction

Dans ce chapitre, on va discuter de l'implémentation, l'énumérant , les outils, les fonctions, les modèles, les ensembles de données utilisés et les protocoles de test jusqu'à ce que le but de notre étude soit confirmé. Un ensemble de mesures supervisées telles que le rappel, la précision, le f1score et la fonction de perte ont été utilisés. Pour affiner les résultats, un scénario complet d'une étude comparative avec différents ensembles de données, modèles est utilisé pour améliorer la performance du modèle.

3.2 Jeux de données utilisés

3.2.1 curatedTCGAData

le jeu de données curatedTCGAData est un très grand ensemble de données collecté par Le Cancer Genome Atlas (TCGA) qui contient 20000 échantillons de tumeurs et d'échantillons normaux . les 33 types de cancer était selectionées pour l'etude suivant des critères précis notamment :

✓ Mauvais pronostic

✓ Incidence globale sur la santé publique

✓ Tumeur primaire non traitée

Dans ce projet le jeu de données est utilisé pour étudier l'efficacité de l'algorithme de classification pour détecter les différents types de gènes qui présentes une anomalie ou une variation et la classer dans le type de cancer correspondant .
Notre jeu de données est composé de 19321 attributs (échantillons) . les types de cancer sont présenter par 7 classes : BRCA , HNSC , KIRC , LGG , LUAD , LUSC , THCA [6].

Le type de cancer	**Le nom scientifique complet**
BRCA	Breast Invasive Carcinoma (Carcinome invasif du sein)
HNSC	Head And Neck Squamous Cell Carcinoma (Carcinome épidermoïde de la tête et du cou)
KIRC	Kidney Renal Clear Cell Carcinoma (Carcinome rénal à cellules claires)
LGG	Brain Lower Grade Glioma (Gliome cérébral de bas grade)
LUAD	Lung Adenocarcinoma (adénocarcinome pulmonaire)
LUSC	Lung Squamous Cell Carcinoma (Carcinome épidermoïde pulmonaire)
THCA	Thyroid Carcinoma (carcinome thyroïdien)

Tableau 3.1: les classes du jeu de données curatedTCGAData .

Types de cancer	BRCA	HNSC	KIRC	LGG	LUAD	LUSC	THCA
Nombres	1093	520	533	516	515	501	501

Tableau 3.2: le nombre d'échantillon du jeu de données curatedTCGAData.

3.2.2 TCGA.rnaseq :

Un sous-ensemble des données du TCGA , Les données d'exemple ont été générées à partir d'une sélection aléatoire de 200 échantillons de chacun des types de cancer [1]:

BRCA, KIRC et COAD (total de 600 échantillons).

Ensuite, les 5 000 gènes présentant la plus grande variation standard ont été sélectionnés [14].

Types de cancer	BRCA	COAD	KIRC
Nombres	200	200	200

Tableau 3.3 : le nombre d'échantillon du jeu de données TCGA.rnaseq .

3.2.3 GISETTE

L'ensemble de données se compose de trois ensembles : formation, validation et test avec 6000, 1000 et 6500 observations respectivement. L'ensemble de données comprend un total de 5000 caractéristiques, dont 2500 sans pouvoir prédictif. L'ordre des caractéristiques et des modèles a été aléatoire [16].

Le problème de reconnaissance de chiffres manuscrits dans cet ensemble de données concernant la séparation des chiffres ' 4' e t ' 9', fait référence à une tâche de classification dans le domaine de la vision par ordinateur. L'objectif est d'entraîner un modèle d'apprentissage automatique pour distinguer de manière précise et fiable entre les chiffres manuscrits '4 ' et '9', qui sont souvent confondus en raison de leur similitude visuelle [16].

Pour résoudre ce problème, on utilise généralement des techniques d'apprentissage automatique, telles que les réseaux de neurones convolutionnels (CNN), qui sont bien adaptés à la reconnaissance d'images. Ces modèles peuvent être entraînés à reconnaître les caractéristiques distinctives des chiffres '4' et '9', comme la présence ou l'absence d'une boucle, la forme du trait
vertical, etc.

Une fois entraîné, le modèle peut être utilisé pour prédire correctement si un chiffre manuscrit donné est un '4' ou un '9'. Il peut être intégré dans des applications de reconnaissance optique de caractères (OCR), de tri automatique de courrier, de détection de fraudes, etc., où la précision de la reconnaissance des chiffres est essentielle [16].

Le nombre d'attributs 5000 présenter par 2 classes.

Dataset	**Nombre d'attributs**	**Nombre de lignes**	**Nombre de classe**
curatedTCGAData	19321	4179	07
TCGA.rnaseq	5OO1	600	03
Gisette	2500	2500	02

Tableau 3.4 : Tableau récapitulatif des ensembles des données.

3.3 Choix du langage de programmation

Python est un langage de programmation interprété à usage général, facile à apprendre et à utiliser principalement parce qu'il est axé sur la lisibilité. Ses structures de données de haut niveau intégrées, associées à un typage dynamique et à une liaison dynamique, le rendent très attrayant pour le développement rapide d'applications ainsi que pour une utilisation en tant que langage de script ou de liaison pour lier des composants existants. La syntaxe simple et facile à apprendre de Python met l'accent sur la lisibilité et réduit les coûts de maintenance du programme [17].

Python prend en charge les modules et les paquets qui favorisent la modularité du programme et la réutilisation du code. L'interpréteur Python et la vaste bibliothèque standard sont disponibles gratuitement sous forme source ou binaire pour toutes les principales platesformes et sont libres de redistribuer [17].

3.4 Configuration de l'environnement expérimental

Un cadre de deeplearning open source a été appliqué pour simplifier la mise en œuvre de modèles complexes et de grande taille et pour implémenter et optimiser le modèle que nous proposons.

• TensorFlow :

TensorFlow est une plate-forme open source complète pour la construction d'applications d'apprentissage automatique, c'est une bibliothèque mathématique symbolique qui utilise le flux de données et la programmation différentielle pour effectuer une variété de tâches axées sur la formation et l'inférence des réseaux neuronaux profonds. Permet aux novices et aux experts de créer des modèles d'apprentissage automatique pour ordinateur de bureau, mobile, Web ou Cloud en utilisant une variété d'outils, de bibliothèques et de ressources communautaires . et avoir plusieurs wrappers en plusieurs langages tels que Python, Java, C++. Les utilisations les plus populaires de TensorFlow(sa applications.) : détection vidéo, reconnaissance d'images et de voix et applications de texte [19].

• Keras :

Keras est une API conçue pour les humains, pas pour les machines. Keras adhère aux meilleures pratiques pour réduire la charge cognitive : elle offre des API cohérentes et simples, minimise le nombre d'actions utilisateur requises dans les cas d'utilisation courants et fournit des commentaires clairs et utiles lorsque des erreurs utilisateur se produisent. Il comprend également une documentation complète et des guides de développement. Keras résout de nombreux problèmes différents comme la classification des images à l'aide de différentes architectures telles que VGG16, Xception, ResNet, Inception [18].

• SkLearn :

Sklearn (scikit-learn) est une bibliothèque Python qui fournit une variété d'algorithmes d'apprentissage automatique supervisés et non supervisés [20]. Il est basé sur certaines technologies comme NumPy, Pandas et matplotlib. Les principaux cas d'utilisation de cette bibliothèque peuvent être divisés en 6 catégories [20]:

Les principaux cas d'utilisation de cette bibliothèque peuvent être divisés en 6 catégories:

1. Prétraitement (normalisation min-max)
2. Régression (régression logistique)
3. Classification (CNN)
4. Regroupement (moyennes K)
5. Sélection du modèle
6. Réduction de la dimension

• Google Colaboratory

Google Colab ou Colaboratory est un service cloud fourni par Google (gratuit). Il est basé sur Jupyter Notebook et destiné à l'apprentissage et à la recherche automatiquement. Cette plateforme vous permet de former des modèles de machine learning directement dans le cloud, avec : Aucune configuration requise, accès gratuit à GPU et partage facile [17].

3.5 Protocole de test :

Dans notre recherche, nous avons développé un protocole de test pour faire une enquête pour prouver son efficacité et sa qualité dans le protocole en commençe par la préparation des données passant à l'étape la plus essentielle qui est la génération de pseudo-images puis la configuration CNN, la phase de formation et enfin la phase de test.

3.5.1 les différentes étapes du protocole :

- La première étape consiste à préparer les données afin qu'elles puissent s'intégrer au modèle pour continuer les essais.

- Deuxièmement, l'étape la plus importante est la génération de pseudo-images. L'objectif était de mettre l'accent sur la visualisation de l'aspect des images fournies à CNN, tout en étant présentées dans différentes tailles et cadres.

- La troisième étape était la configuration CNN pour traiter les entrées avec les différentes formes utilisées, l'importation des bibliothèques nécessaires, l'ajout de couches denses.

- Puis la phase de formation.

- Enfin la phase de test pour évaluer le modèle puis le comparer.

La figure 3.1 montre les étapes du protocole de test.

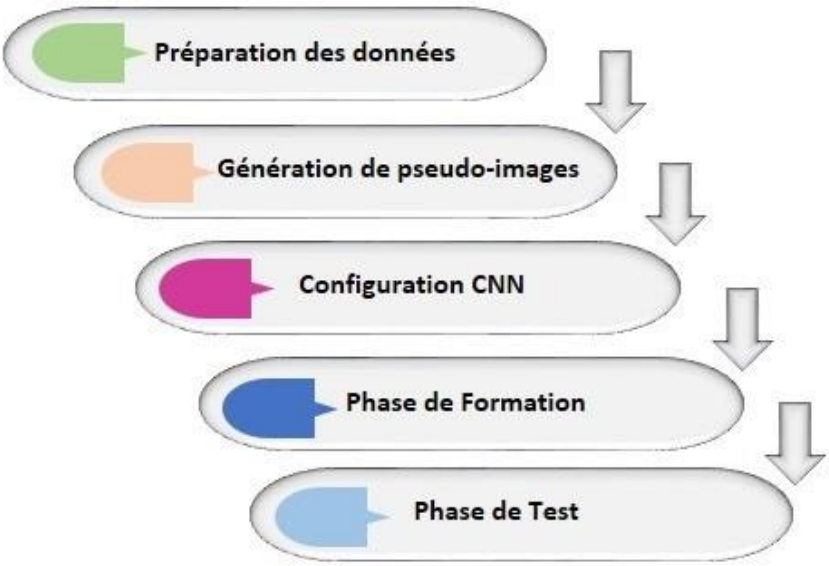

Figure 3.1:Etapes du Protocole de test.

3.5.1.1 Préparation des données

Pour curatedTCGAData :

➢ on a divisé l'ensemble de données en fichier test (20%) et fichier train (80%).

➢ Création d'un cadre de données avec des étiquettes multi-classes (BRCA , HNSC , KIRC , LGG , LUAD , LUSC , THCA).

➢ Deuxième codage d'étiquette (0, 1, 2, 3, 4, 5, 6) étiquettes multi-classes (BRCA , HNSC , KIRC , LGG , LUAD , LUSC , THCA).

Pour TCGA.rnaseq :

➢ On a divisez l'ensemble de données en fichier test(20%) et fichier train (80%).

➢ Création d'un cadre de données avec des étiquettes multi-classes (BRCA , COAD , KIRC).

➢ Deuxième codage d'étiquette (0, 1, 2,) étiquettes multi-classes (BRCA , COAD , KIRC).

Pour GISETTE :

➢ On a divisez l'ensemble de données en fichier test(20%) et fichier train (80%).

3.6 Résultat et discussion

Plusieurs expériences ont été menées à l'aide de différents ensembles de données, tailles d'images, images en couleurs et classificateurs pour s'assurer que les résultats obtenus correspondent à nos hypothèses. Pour ce faire, plusieurs phases de test ont été réalisées, à l'ordre suivant :

1. Le premier test consistait à examiner l'impact de la variation de la taille des images pour chaque ensemble de données. Ce test visait à identifier les différences pouvant découler de ces variations.
2. Dans le deuxième test, on a essayé de générer des images RGB (en couleurs) pour chaque ensemble de données avec la variation de tailles d'images.
3. Le troisième test consiste à comparer notre modèle CNN aux classificateurs Random Forest , Arbre de decision et Ada-Boost . Ce test visait à discerner toute variation significative du rendement et des résultats entre notre modèle et les modèles cités.

En menant systématiquement ces expériences et tests, nous nous sommes assurés que les résultats obtenus fourniraient des preuves est fiables pour soutenir nos hypothèses.

3.6.1 Test N°1 :

L'idée du premier test était d'utiliser des images et des trames de tailles différentes avec les trois jeux de données. Nous avons utilisé (80%) de l'ensemble de données pour la formation du CNN et les (20%) restants ont été utilisés pour les tests.

✐ Les paramètres de notre modèle :

Epochs number : **10**
Batch_size : **32**
Random state : **42**
Optimizer : **Adam**

3.6.1.1 Résultats curatedTCGAData :

Les tableaux suivants présentent les résultats du test n°1 avec reasmp = false et les tests avec reasmp = true

Le premier tableau (tableau 3.4)**: resamp = False**

test	description	Accuracy:	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x19321/32/false	0.8469	0,8469	0,8469	0.8469	0,4621	1	19321	19321
2	400x50/32/false	0.8804	0,8804	0.8804	0.8804	0,2898	400	50	20000
3	1x19321/64/false	0.8852	0,8852	0.8852	0.8852	1,1655	1	19321	19321
4	142x142/64/false	0.9211	0,9211	0.9211	0.9211	1,3929	142	142	20164
5	1x19321/128/false	0.8636	0,8636	0.8636	0.8636	5,2274	1	19321	19321
6	19321x1/128/false	0.9091	0,9091	0.9091	0.9091	5,5499	19321	1	19321
7	400x50/128/false	0.9306	0,9306	0.9306	0.9306	3,0956	400	50	20000
8	500x40/128/false	0.9450	0,9450	0.9450	0.9450	5,0873	500	40	20000

Tableau 3.5: Résultats du test N°1 curatedTCGAData cadre (32X32,64x64,128x128) resamp=false.

La figure 3.2 présente un graphique de précision et la figure 3.3 présente un graphique de temps d'exécution.On a fait une première comparaison entre les différentes tailles d'images avec resamp =false , et on a tirer les observations suivantes :

1. La meilleure précision était dans la combinaison (500x40/128/false) ou la taille du cadre est (500x40) et la taille d'image est 128 mais le temps d'exécution est élevé.
2. une bonne précision dans(142x142/64/false) ou la taille du cadre est (142x142) et la taille d'image 64 avec un temps meilleur que la précision précédente.

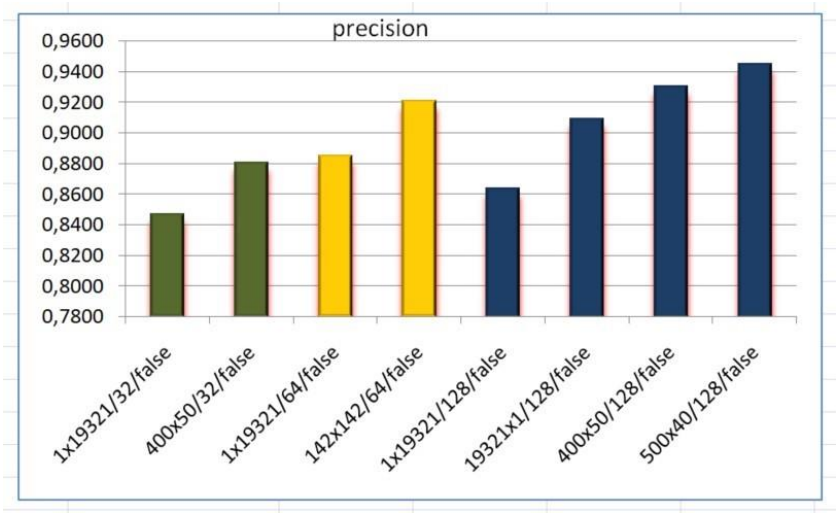

Figure 3.2: Graphe de précision test N°1 du jeu de données curatedTCGAData resamp=false

Figure 3.3: Graphe du temps d'éxécution test N°1 du jeu de données curatedTCGAData resamp=false

Le deuxiéme tableau(tableau 3.6) : resamp = True

test	description	Accuracy:	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x19321/32/true	0,5813	0,5813	0,5813	0,5813	0,6786	1	19321	19321
2	400x50/32/true	0,8803	0,8803	0,8803	0,8803	0,27	400	50	20000
3	1x19321/64/true	0,8229	0,8229	0,8229	0,8229	0,7572	1	19321	19321
4	142x142/64/true	0,8923	0,8923	0,8923	0,8923	0,7653	142	142	20164
5	1x19321/128/true	0,6962	0,6962	0,6962	0,6962	5,697	1	19321	19321
6	19321x1/128/true	0,7799	0,7799	0,7799	0,7799	2,9419	19321	1	19321
7	400x50/128/true	0,9426	0,9426	0,9426	0,9426	5,3624	400	50	20000
8	500x40/128/true	0,933	0,933	0,933	0,933	5,2644	500	40	20000

Tableaue 3.6:Résultats du testN°1 curatedTCGAData cadre (32X32, 64x64, 128x128) resamp=true.

Comme le démontre la figure 3.4 et la figure 3.5 concernant respectivement la précision et le temps d'exécution on a tiré les observations suivante:

1. la meilleure précision est dans (400x50/128/true) de la trame (400x50) et la taille d'image 128 mais le temps d'exécution est élevé.
2. une bonne précision dans (142x142/64/true) de la trame (142x142) avec un temps d'exécution plus court.
3. le meilleurs temps d'exécution était dans (400x50/32/true) de le trame (400x50) avec une bonne précision .

le choix est pour la précision dans la combinaison (142x142/64/true) ou la taille du cadre est (142x142) et la taille d'image est 64(le choix du cadre est carré).

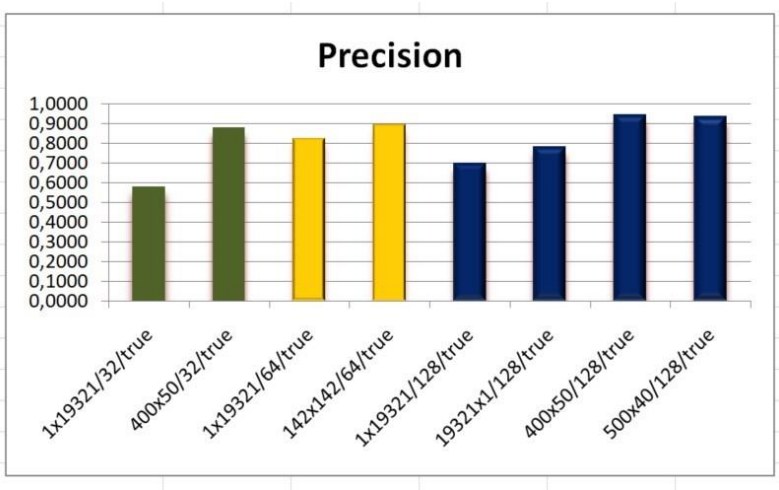

Figure 3.4:Graphe de précision test N°1 du jeu de données curatedTCGAData resamp=true.

Figure 3.5:Graphe du temps d'exécution test N°1 du jeu de données curatedTCGAData resamp=true.

Observation général : pour le data set **curatedTCGAData** la meilleure précision dans un meilleur temps d'exécution est quand le cadre est carré et la taille d'image est 64 .
Pour le ré échantillonnage le resamp=false est le choix idéal dans notre premier test .

La matrice de confusion pour ce choix (Figure 3.6)

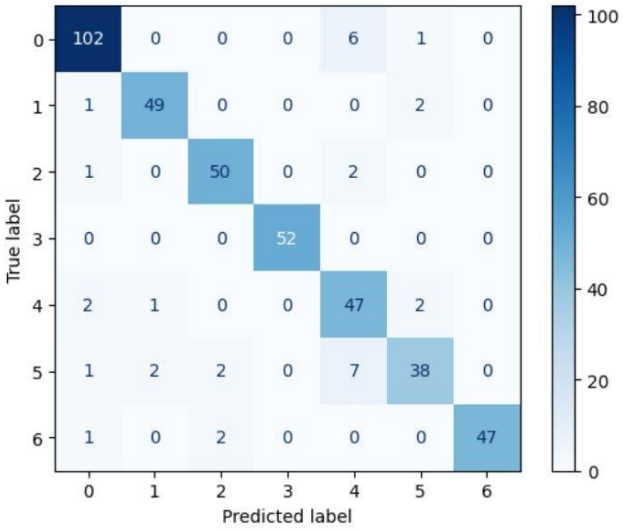

Figure 3.6 : matrice de confusion cadre (142x142) taille d'image 64 resamp=false (curatedTCGAData).

3.6.1.2 Résultats TCGA.rnaseq:

Resultats du test n°1 resump=false : Le tableau 3.7 présente les résultats du test n°1 du jeu de données TCGA.rnaseq avec resamp= false.

test	description	Accuracy:	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x5000/32/false	0,9666	0,9666	0,9666	0,9666	0,2302	1	5000	5000
2	5000x1/32/false	0,9166	0,9166	0,9166	0,9166	0,2093	5000	1	5000
3	50x100/32/false	0,9000	0,9000	0,9000	0,9000	0,2025	50	100	5000
4	100x50/32/false	0,9500	0,9500	0,9500	0,9500	0,2319	100	50	5000
5	250x25/32/false	0,9166	0,9166	0,9166	0,9166	0,1759	250	25	6250
6	71x71/32/false	0,9166	0,9166	0,9166	0,9166	0,2198	71	71	5041
7	1x5000/64/false	0,9500	0,9500	0,9500	0,9500	0,2896	1	5000	5000
8	5000x1/64/false	0,9333	0,9333	0,9333	0,9333	0,7999	5000	1	5000
9	50x100/64/false	0,9453	0,9453	0,9453	0,9453	0,2361	50	100	5000
10	100x50/64/false	0,9166	0,9166	0,9166	0,9166	0,2321	100	50	5000
11	250x25/64/false	0,9333	0,9333	0,9333	0,9333	0,2352	250	25	6250
12	71x71/64/false	0,9500	0,9500	0,9500	0,9500	0,3178	71	71	5041
13	1x5000/128/false	0,9667	0,9667	0,9667	0,9667	0,796	1	5000	5000
14	5000x1/128/false	0,9667	0,9667	0,9667	0,9667	0,7682	5000	1	5000
15	50x100/128/false	0,8500	0,8500	0,8500	0,8500	0,7616	50	100	5000
16	100x50/128/false	0,9500	0,9500	0,9500	0,9500	0,8635	100	50	5000
17	250x25/128/false	0,9423	0,9423	0,9423	0,9423	0,6373	250	25	6250
18	71x71/128/false	0,9333	0,9333	0,9333	0,9333	0,7771	71	71	5041

Tableau 3.7Résultats du testN°1 TCGA.rnaseq cadre (32X32, 64x64, 128x128) resamp=false

Comme le démontre la figure 3.7 et la figure 3.8 concernant respectivement le graphe de précision et le graphe du temps d'exécution et aprés comparaison entre les différentes tailles d'image et resamp = false on a tiré les observations suivante:

1. toutes les tailles d'images et cadres ont donnés une bonne précision .
2. la meilleure précision avec un temps d'exécution idéal est dans la combinaison (1x5000/32/false) ou la taille du cadre est (1x5000) et la taille d'image est 32.
3. une précision similaire avec la même taille du cadre précédent mais la taille d'image est 128.
4. On peut choisir aussi la combinaison (5000x1/128/false) ou la combinaison (100x50/32/false) une très bonne précision et un bon temps d'exécution.

Le choix est pour le cadre (1x5000) (rectangulaire) et la taille d'image 32 et resamp=false.

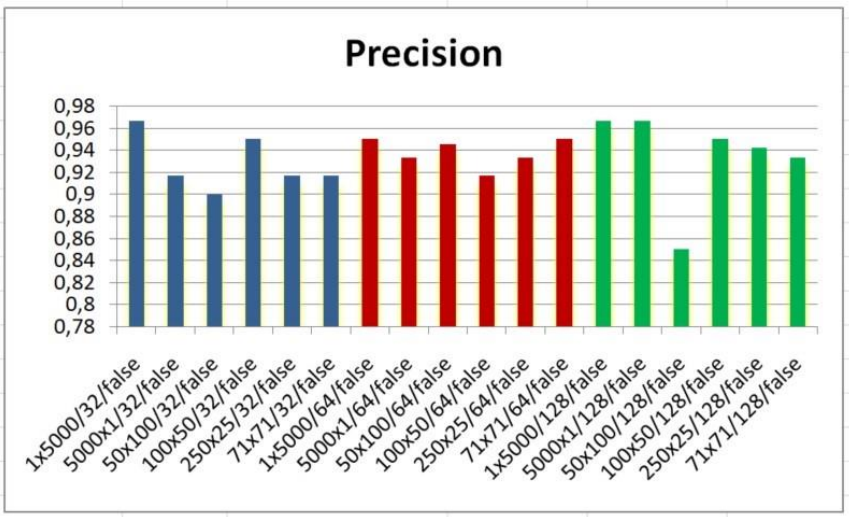

Figure 3.7:Graphe de précision test N°1 du jeu de données TCGA.rnaseq resamp=false.

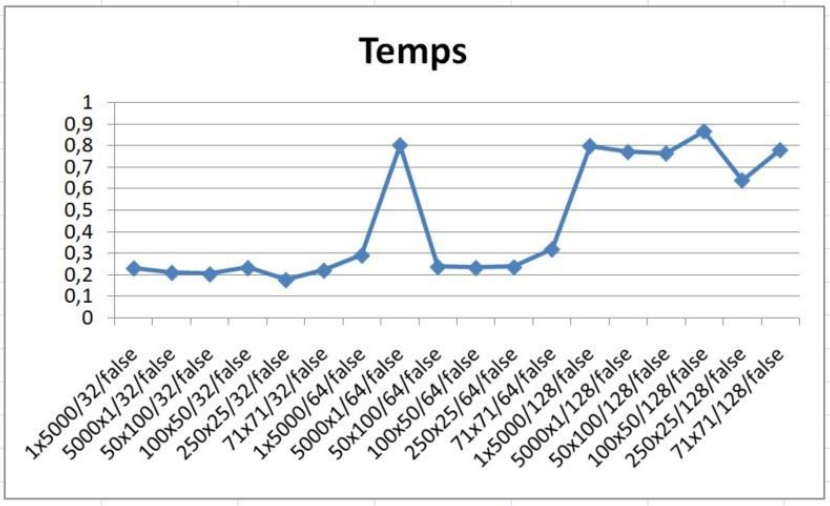

Figure 3.8:Graphe du temps d'exécution test N°1 du jeu de données TCGA.rnaseq resamp=false.

Resultats du test n°1 resump=true :

Le tableau 3.8 présente les résultats du test n°1 du jeu de données TCGA.rnaseq avec resamp = true

test	description	Accuracy:	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x5000/32/true	0,6500	0,6500	0,6500	0,6500	0,1896	1	5000	5000
2	5000x1/32/true	0,6833	0,6833	0,6833	0,6833	0,2081	5000	1	5000
3	50x100/32/true	0,9166	0,9166	0,9166	0,9166	0,2488	50	100	5000
4	100x50/32/true	0,7500	0,7500	0,7500	0,7500	0,1862	100	50	5000
5	250x25/32/true	0,8333	0,8333	0,8333	0,8333	0,214	250	25	6250
6	71x71/32/true	0,8456	0,8456	0,8456	0,8456	0,2074	71	71	5041
7	1x5000/64/true	0,8500	0,8500	0,8500	0,8500	0,2132	1	5000	5000
8	5000x1/64/true	0,9142	0,9142	0,9142	0,9142	0,2096	5000	1	5000
9	50x100/64/true	0,8833	0,8833	0,8833	0,8833	0,2671	50	100	5000
10	100x50/64/true	0,9666	0,9666	0,9666	0,9666	0,5189	100	50	5000
11	250x25/64/true	0,9333	0,9333	0,9333	0,9333	0,2732	250	25	6250
12	71x71/64/true	0,9000	0,9000	0,9000	0,9000	0,4951	71	71	5041
13	1x5000/128/true	0,8666	0,8666	0,8666	0,8666	1,5173	1	5000	5000
14	5000x1/128/true	0,6666	0,6666	0,6666	0,6666	0,6164	5000	1	5000
15	50x100/128/true	0,9666	0,9666	0,9666	0,9666	1,1173	50	100	5000
16	100x50/128/true	0,9700	0,9700	0,9700	0,9700	1,4949	100	50	5000
17	250x25/128/true	0,9166	0,9166	0,9166	0,9166	1,7115	250	25	6250
18	71x71/128/true	0,9833	0,9833	0,9833	0,9833	0,7706	71	71	5041

Tableau 3.8:Résultats du testN°1 TCGA.rnaseq cadre (32X32, 64x64, 128x128) resamp=true.

Comme le démontre la figure 3.10 et la figure 3.11 concernant respectivement le graphe de précision et le graphe du temps d'exécution et aprés comparaison entre les différentes tailles d'image et resamp = true on a tiré les observations suivante:

1. La meilleure précision et dans la combinaison (100x50 /128 /true) avec un bon temps d'exécution .
2. Moins de précision dans la combinaison (1x5000/32/true) et la combinaison (5000x1/128/true) donc c'est un choix à éviter .
3. Le temps d'execution était bon pour toutes les combinaisons ce qui favorise le choix de ce test avec plusieurs combinaisons .

Le meilleur choix pour ce test c'est le cadre (100x50) et taille d'image 128 avec resamp=true.

Le rééchantillonnage avec ces deux valeurs (true et false) donne une très bonne précision pour ce jeu de données (**TCGA.rnaseq**).
Le choix pour ce jeu de données était le resamp=true,un cadre(100x50)et une taille d'image 128.

La matrice de confusion (Figure 3.9).

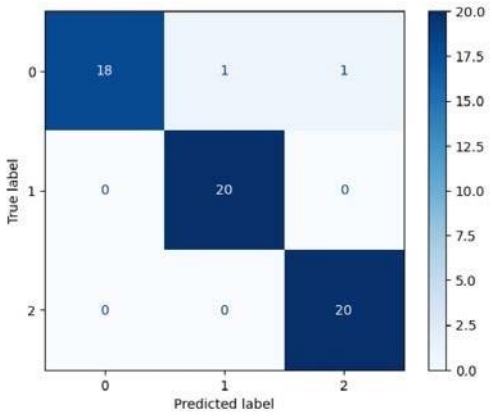

Figure 3.9 : Matrice de confusion cadre (1x5000) taille d'image 32 resamp=false (TCGA.rnaseq).

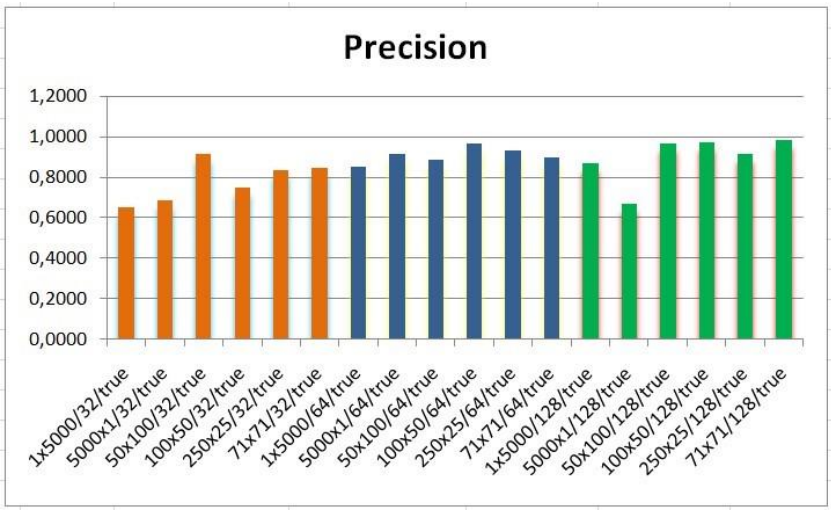

Figure 3.10: Graphe du précision test N°1 du jeu de données TCGA.masq resamp=true

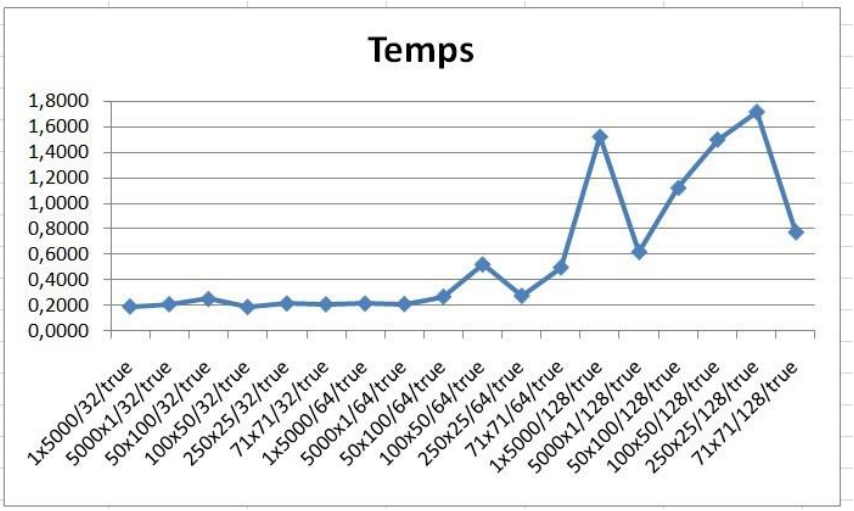

Figure 3.11: Graphe du temps d'exécution test N°1 du jeu de données TCGA.masq resamp=true

3.6.1.3 Résultats GISETTE:

Resultats du test n°1 resump=false:

Le tableau 3.9 présente les résultats du test n°1 du jeu de données Gisette avec resamp=false.

test	description	Accuracy:	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x5000/32/false	0,7543	0,7543	0,7543	0,7543	0,4255	1	5000	5000
2	5000x1/32/false	0,7285	0,7285	0,7285	0,7285	0,3915	5000	1	5000
3	50x100/32/false	0,8914	0,8914	0,8914	0,8914	0,2704	50	100	5000
4	100x50/32/false	0,8686	0,8686	0,8686	0,8686	0,4025	100	50	5000
5	250x25/32/false	0,8843	0,8843	0,8843	0,8843	0,4435	250	25	6250
6	71x71/32/false	0,8900	0,8900	0,8900	0,8900	0,4067	71	71	5041
7	1x5000/64/false	0,6643	0,6643	0,6643	0,6643	1,4029	1	5000	5000
8	5000x1/64/false	0,6971	0,6971	0,6971	0,6971	1,4014	5000	1	5000
9	50x100/64/false	0,9000	0,9000	0,9000	0,9000	1,4279	50	100	5000
10	100x50/64/false	0,8771	0,8771	0,8771	0,8771	1,4015	100	50	5000
11	250x25/64/false	0,9186	0,9186	0,9186	0,9186	1,4176	250	25	6250
12	71x71/64/false	0,8914	0,8914	0,8914	0,8914	2,2864	71	71	5041
13	1x5000/128/false	0,7971	0,7971	0,7971	0,7971	4,3331	1	5000	5000
14	5000x1/128/false	0,7542	0,7542	0,7542	0,7542	5,3194	5000	1	5000
15	50x100/128/false	0,9157	0,9157	0,9157	0,9157	10,3716	50	100	5000
16	100x50/128/false	0,9171	0,9171	0,9171	0,9171	5,2374	100	50	5000
17	250x25/128/false	0,9343	0,9343	0,9343	0,9343	5,3473	250	25	6250
18	71x71/128/false	0,8829	0,8829	0,8829	0,8829	5,2556	71	71	5041

Tableau 3.9:Résultats du test N°1 gisette cadre (32X32, 64x64, 128x128) resamp=false.

Comme le démontre la figure 3.12 et la figure 3.13 concernant respectivement le graphe de précision et le graphe du temps d'exécution et aprés comparaison entre les différentes tailles d'image et resamp = false on a tiré les observations suivante:

1. La meilleure précision est dans la combinaison (250x25/128/false) mais le temps d'exécution est élevé .
2. Une bonne précision avec un temps d'exécution court dans la combinaison (250x25/64 /false).
3. Un temps d'exécution énorme dans la combinaison (50x100/128/false) ce qui paralyse le travail de notre modèle

Le choix est pour le cadre (250x25) avec une taille d'image 64 et resamp=false

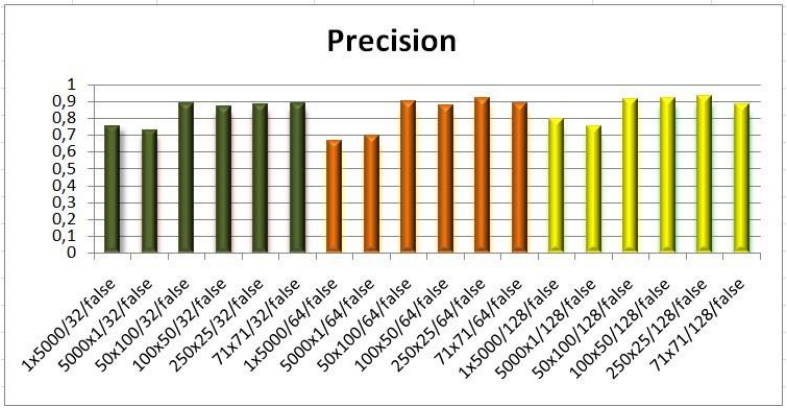

Figure 3.12: Graphe de précision test N°1 du jeu de données gisette resamp=false.

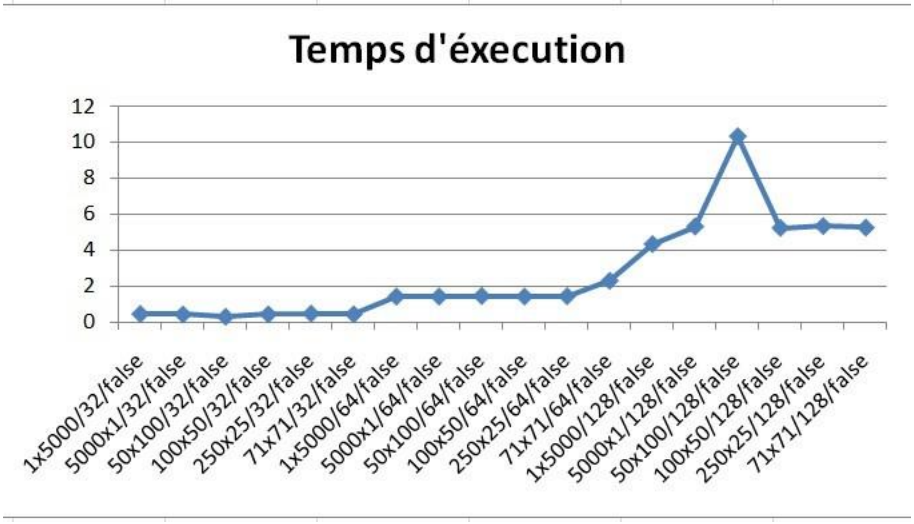

Figure 3.13 : Graphe du temps d'exécution test N°1 du jeu de données gisette resamp=false.

Resultats du test n°1 resump=true :

Le tableau 3.10 présente les résultats du test N°1 avec resamp= true pour le jeu de données Gisette.

test	description	Accuracy	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x5000/32/true	0,7114	0,7114	0,7114	0,7114	0,4401	1	5000	5000
2	5000x1/32/true	0,7234	0,7234	0,7234	0,7234	1,5191	5000	1	5000
3	50x100/32/true	0,8971	0,8971	0,8971	0,8971	0,4154	50	100	5000
4	100x50/32/true	0,8986	0,8986	0,8986	0,8986	0,4393	100	50	5000
5	250x25/32/true	0,9171	0,9171	0,9171	0,9171	0,9337	250	25	6250
6	71x71/32/true	0,8914	0,8914	0,8914	0,8914	0,6350	71	71	5041
7	1x5000/64/true	0,7571	0,7571	0,7571	0,7571	2,7376	1	5000	5000
8	5000x1/64/true	0,7857	0,7857	0,7857	0,7857	1,1072	5000	1	5000
9	50x100/64/true	0,8971	0,8971	0,8971	0,8971	1,1347	50	100	5000
10	100x50/64/true	0,9285	0,9285	0,9285	0,9285	1,3913	100	50	5000
11	250x25/64/true	0,9228	0,9228	0,9228	0,9228	1,3862	250	25	6250
12	71x71/64/true	0,9114	0,9114	0,9114	0,9114	1,328	71	71	5041
13	1x5000/128/true	0,8142	0,8142	0,8142	0,8142	10,4938	1	5000	5000
14	5000x1/128/true	0,8085	0,8085	0,8085	0,8085	5,6139	5000	1	5000
15	50x100/128/true	0,8700	0,8700	0,8700	0,8700	5,452	50	100	5000
16	100x50/128/true	0,8785	0,8785	0,8785	0,8785	7,2722	100	50	5000
17	250x25/128/true	0,9314	0,9314	0,9314	0,9314	6,577	250	25	6250
18	71x71/128/true	0,9257	0,9257	0,9257	0,9257	10,4612	71	71	5041

Tableau 3.10 : Résultats du testN°1 gisette cadre (32X32, 64x64, 128x128) resamp=true.

Comme le démontre la figure 3.15 et la figure 3.16 concernant respectivement le graphe de précision et le graphe du temps d'exécution et aprés comparaison entre les différentes tailles d'image et resamp = true on a tiré les observations suivante:

1. Une meilleure précision dans les combinaisons(100x50/64/true) et (250x25/64/true) avec un bon temps d'exécution.
2. Le cadre (250x25) avec les tailles d'image 32 et 64 donne une très bonne précision avec un meilleure temps d'exécution .

3. Le cadre (71x71) avec une taille d'image 128 donne une très bonne précision mais un temps d'exécution énorme donc un choix à éviter . par contre ce même cadre avec une taille d'image 64 donne un temps d'exécution idéal pour notre model .

Le choix pour ce test peut être soit un cadre de (250x25)avec une taille d'image 32/64 ou un cadre 71x71 avec une taille d'image 64.

Observation générale :

Le choix du cadre 250x25 est idéal avec les deux valeurs(true et false) du rééchantilonnage et une taille d'image 64.
La matrice de confusion(Figure 3.14).

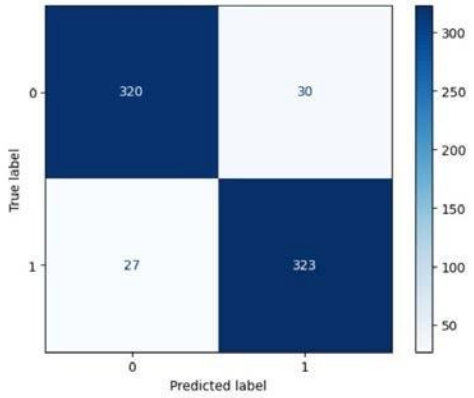

Figure 3.14 : matrice de confusion cadre(250x25) taille d'image 64 resamp=true(gisette).

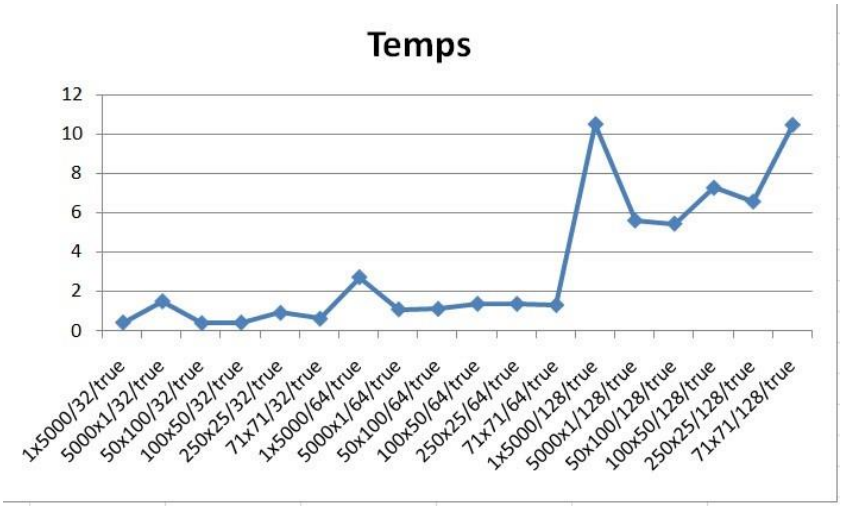

Figure 3.15 : Graphe de précision test N°1 du jeu de données gisette resamp=true.

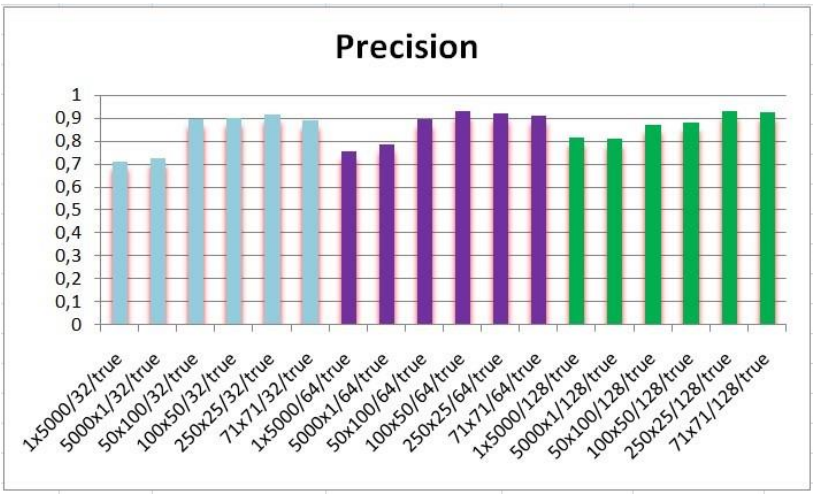

Figure 3.16 : Graphe du temps d'exécution test N°1 du jeu de données gisette resamp=true.

3.6.2 Test N°2 :

L'idée du 2eme test était de générer des images RGB à partir des ensembles de données. on a comme entrée un tableau , des dimensions (HxL) avec la variation de tailles d'image et un indicateur de rééchantillonnage . dans ce test on a chercher a prouvez que notre modèle est capable d'avoir des résultats satisfaisants pour chaque jeu de données alors un développement dans ces domaines.

✎ Les paramètres de notre modèle :

Epochs number : **25**
Batch_size : **32**
Random state : **42**
Optimizer : **Adam**

3.6.2.1 Résultats curatedTCGAData :

Le tableau 3.10 présente les résultats du test N°1 avec resamp=true pour le jeu de données curatedTCGAData avec resamp=false.

test	description	Accuracy:	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x19321/32/false	0,4904	0,4904	0,4904	0,4904	0,1881	1	19321	19321
2	133x50/32/false	0,5909	0,5909	0,5909	0,5909	0,2039	133	50	6650
3	81x81/32/false	0,5287	0,5287	0,5287	0,5287	0,1906	1	19321	6561
4	1x19321/64/false	0,3541	0,3541	0,3541	0,3541	0,2282	1	19321	19321
5	133x50/64/false	0,7441	0,7441	0,7441	0,7441	0,2913	133	50	6650
6	81x81/64/false	0,2607	0,2607	0,2607	0,2607	0,1761	1	19321	6561
7	1x19321/128/fals	0,3445	0,3445	0,3445	0,3445	0,2164	1	19321	19321
8	133x50/128/false	0,2607	0,2607	0,2607	0,2607	0,3307	133	50	6650
9	81x81/128/false	0,2607	0,2607	0,2607	0,2607	0,1998	1	19321	6561

Tableau 3.11 : Résultats du testN°2 curatedTCGAData cadre (32X32,64x64,128x128) resamp=false.

Comme le démontre le graphe de précision figure 3.17 et le graphe du temps d'éxecution figure 3.18 et aprés comparaison on a eu les résultats suivants:

1. La meilleure combinaison était (133x50/64/false) , un cadre de taille (133X50) et une image de taille 64 avec un bon temps d'exécution.
2. Le cadre carré (81X81) avec une taille d'image 128 à donner une mauvaise précision donc ce choix est à éviter.
3. Une précision moyenne dans la combinaison (133x50/32/false) .

Si on choisi à convertir une image en couleur pour ce jeu de données alors le meilleur cadre sera (133X50) avec une taille d'image 64 et un ré échantillonnage resamp=false .

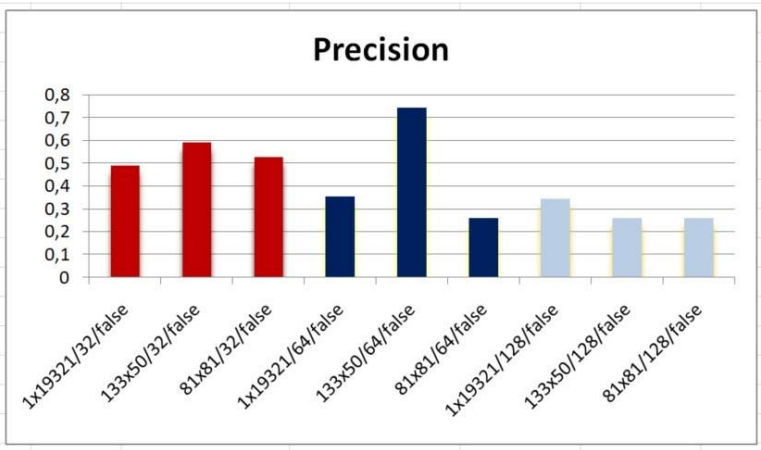

Figure 3.17 : Graphe de précision test N°2 du jeu de données curatedTCGAData resamp=false.

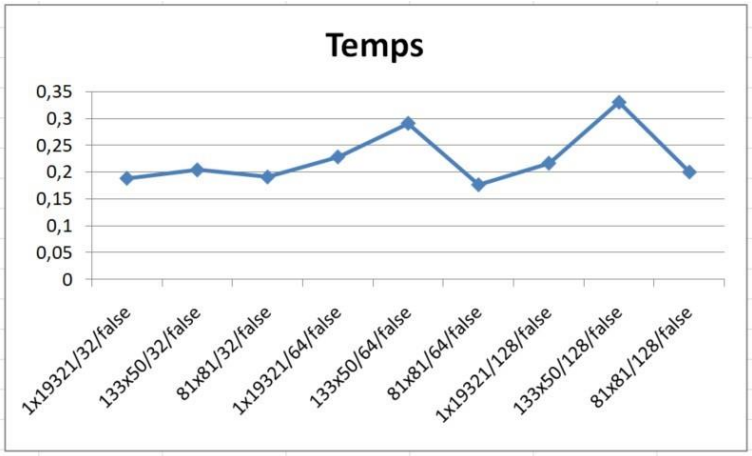

Figure 3.18: Graphe du temps d'exécution test N°2 du jeu de données curatedTCGAData resamp=false.

Le tableau 3..12 présente les résultats du test N°2 resamp=true :

test	description	Accuracy:	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x19321/32/true	0,4976	0,4976	0,4976	0,4976	0,1901	1	19321	19321
2	133x50/32/true	0,5382	0,5382	0,5382	0,5382	0,2032	133	50	6650
3	81x81/32/true	0,5334	0,5334	0,5334	0,5334	0,1757	81	81	6561
4	1x19321/64/true	0,4449	0,4449	0,4449	0,4449	0,3284	1	19321	19321
5	133x50/64/true	0,4665	0,4665	0,4665	0,4665	0,1858	133	50	6650
6	81x81/64/true	0,7416	0,7416	0,7416	0,7416	0,1959	81	81	6561
7	1x19321/128/true	0,2607	0,2607	0,2607	0,2607	0,2536	1	19321	19321
8	133x50/128/true	0,2607	0,2607	0,2607	0,2607	0,2771	133	50	6650
9	81x81/128/true	0,3657	0,3657	0,3657	0,3657	0,1978	81	81	6561

Tableau 3.12 : Résultats du testN°2 curatedTCGAData cadre (32X32,64x64,128x128) resamp=true.

On comparant les résultats obtenus sur la précision(figure 3.19) et le temps d'exécution (figure 3.20) on a tiré les observations suivantes:

1. Une meilleure précision dans la combinaison (81x81/64/true) avec un très bon temps d'exécution
2. Le choix du cadre (81x81) avec les tailles d'image 32 et 128 ne donne pas les résultats souhaités

Le cadre carré (81x81) peut être choisi dans ce test pour ce jeu de données avec une taille d'image 64 et resamp=true

Observation général : Le choix du cadre carré (81x81) avec une taille d'image 64 et un rééchantillonnage de valeur true est idéal pour ce test .

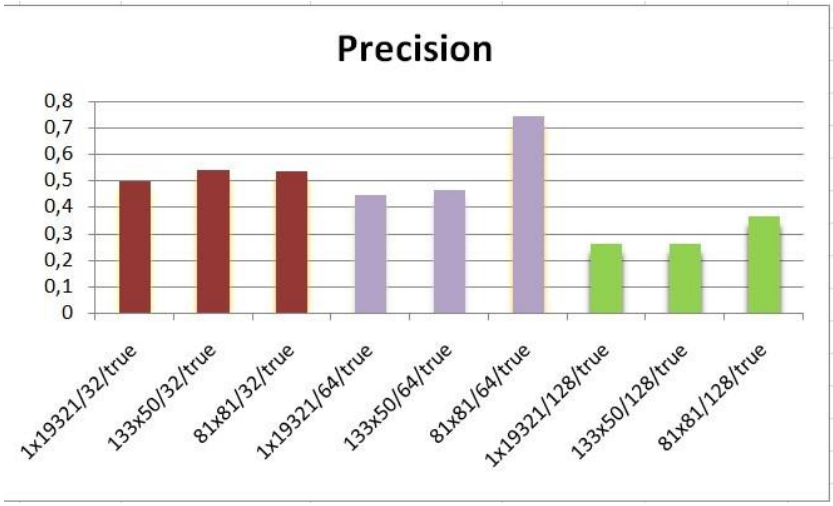

Figure 3.19: Graphe de précision test N°2 du jeu de données curatedTCGAData resamp=true.

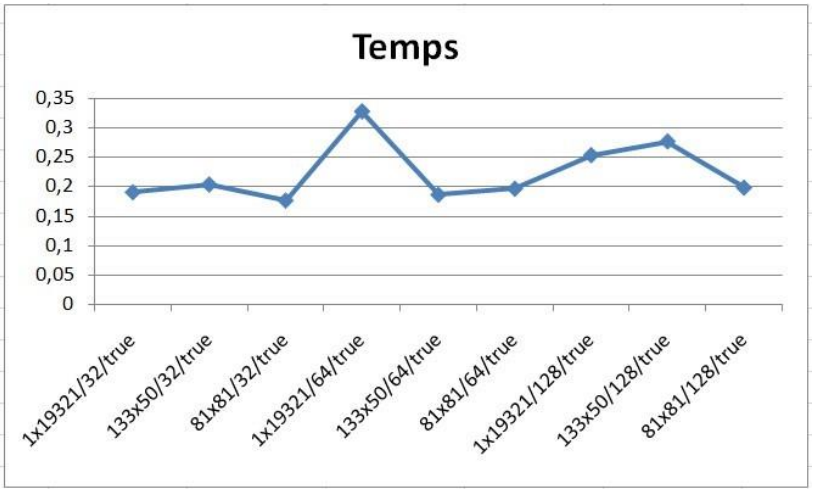

Figure 3.20 : Graphe du temps d'exécution test N°2 du jeu de données curatedTCGAData resamp=true.

3.6.2.2 Résultats TCGA.rnaseq :

Le tableau 3.13 présente les résultats du test n°2 du jeu de données TCGA.rnaseq avec resamp = false

test	description	Accuracy	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x5000/32/false	0,6833	0,6833	0,6833	0,6833	0,1864	1	5000	5000
2	50x100/32/false	0,5666	0,5666	0,5666	0,5666	0,1745	50	100	5000
3	100x50/32/false	0,6500	0,6500	0,6500	0,6500	0,1976	100	50	5000
5	71x71/32/false	0,6666	0,6666	0,6666	0,6666	0,2565	71	71	5041
6	1x5000/64/false	0,7333	0,7333	0,7333	0,7333	0,1894	1	5000	5000
7	50x100/64/false	0,6500	0,6500	0,6500	0,6500	0,1698	50	100	5000
8	100x50/64/false	0,7500	0,7500	0,7500	0,7500	0,1163	100	50	5000
9	71x71/64/false	0,7333	0,7333	0,7333	0,7333	0,1872	71	71	5041
9	1x5000/128/false	0,5333	0,5333	0,5333	0,5333	0,1277	1	5000	5000
10	50x100/128/false	0,7833	0,7833	0,7833	0,7833	0,123	50	100	5000
11	100x50/128/false	0,7500	0,7500	0,7500	0,7500	0,1279	100	50	5000
12	71x71/128/false	0,8333	0,8333	0,8333	0,8333	0,1692	71	71	5041

Tableau 3.13 : Résultats du testN°2 TCGA.rnaseq cadre (32X32, 64x64, 128x128) resamp=false.

Aprés comparaison des résultats de précision commele démontre la figure 3.21 et du temps d'éxecution démontré par la figure 3.22 on a tiré les observations suivantes:

1. La meilleure précision dans la combinaison (71x71/128/false) avec un temps d'exécution idéal.
2. Une bonne précision dans la combinaison (50x100/128/false) .
3. On va pas choisir la taille d'image 32 avec la taille du cadre (50x100) a cause de la précision qui n'aide pas à obtenir les résultats souhaités .

Le choix dans ce test pour ce jeu de données sera une taille de cadre (71x71) un cadre carré avec une taille d'image 128.

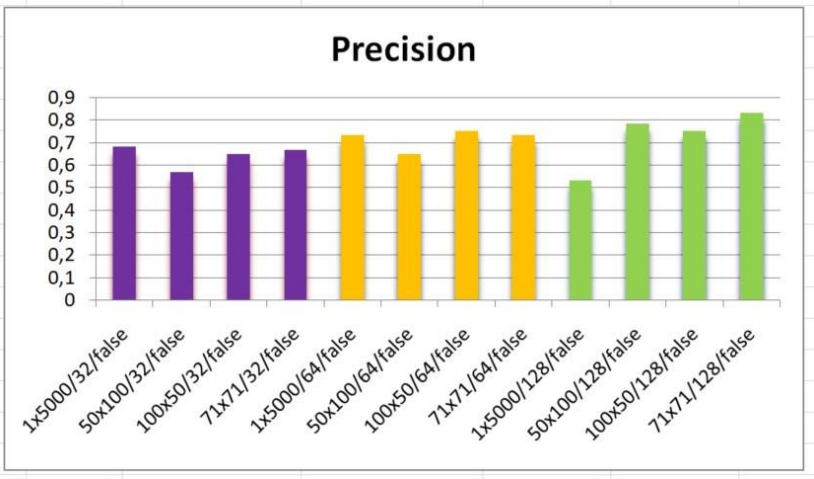

Figure 3.21 : Graphe de précision test N°2 du jeu de données TCGA.rnaseq resamp=false.

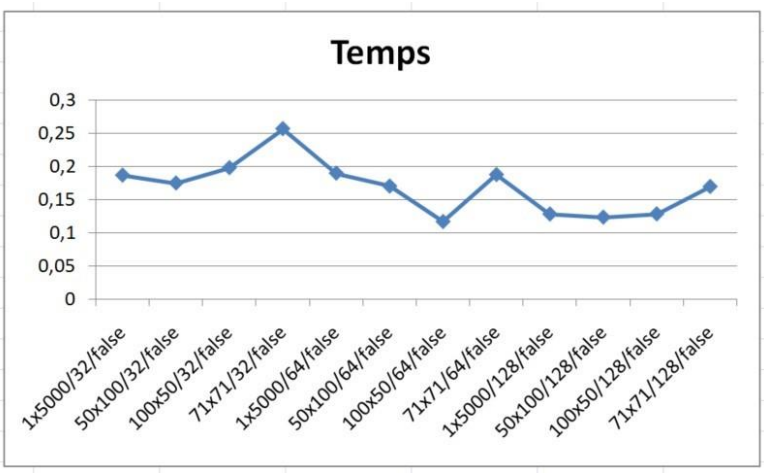

Figure 3.22: Graphe du temps d'exécution test N°2 du jeu de données TCGA.rnaseq resamp=false.

Le tableau 3.14 présente les résultats du test N°2 avec resamp=true

test	description	Accuracy	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x5000/32/true	0,5656	0,5656	0,5656	0,5656	0,1583	1	5000	5000
2	50x100/32/true	0,5500	0,5500	0,5500	0,5500	0,1578	50	100	5000
3	100x50/32/true	0,8000	0,8000	0,8000	0,8000	0,2546	100	50	5000
5	71x71/32/true	0,6666	0,6666	0,6666	0,6666	0,1952	71	71	5041
6	1x5000/64/true	0,5333	0,5333	0,5333	0,5333	0,1943	1	5000	5000
7	50x100/64/true	0,6333	0,6333	0,6333	0,6333	0,1273	50	100	5000
8	100x50/64/true	0,7833	0,7833	0,7833	0,7833	0,1846	100	50	5000
9	71x71/64/true	0,7563	0,7563	0,7563	0,7563	0,1255	71	71	5041
9	1x5000/128/true	0,3666	0,3666	0,3666	0,3666	0,1963	1	5000	5000
10	50x100/128/true	0,8245	0,8245	0,8245	0,8245	0,2162	50	100	5000
11	100x50/128/true	0,8833	0,8833	0,8833	0,8833	0,1959	100	50	5000
12	71x71/128/true	0,6500	0,6500	0,6500	0,6500	0,2451	71	71	5041

Tableau 3.14 : Résultats du testN°2 TCGA.rnaseq cadre (32X32, 64x64, 128x128) resamp=true.

On a fait une compraison des résultats obtenus sur la précision démontrés par la figure 3.23 et sur le temps d'exécution démontrés par la figure 3.24, on a eu les observations suivantes:

1. Une meilleure précision dans la combinaison (100x50/128/true) avec un temps d'exécution optimal .

Le temps d'exécution était court pour toutes les combinaisons avec une variation de précision

Le choix pour ce test un cadre de taille (100x50) avec une taille d'image 128 et un resamp=true.

Observation général: La taille d'image 128 avec un rééchantillonnage true ou false donne une meilleure précision. .

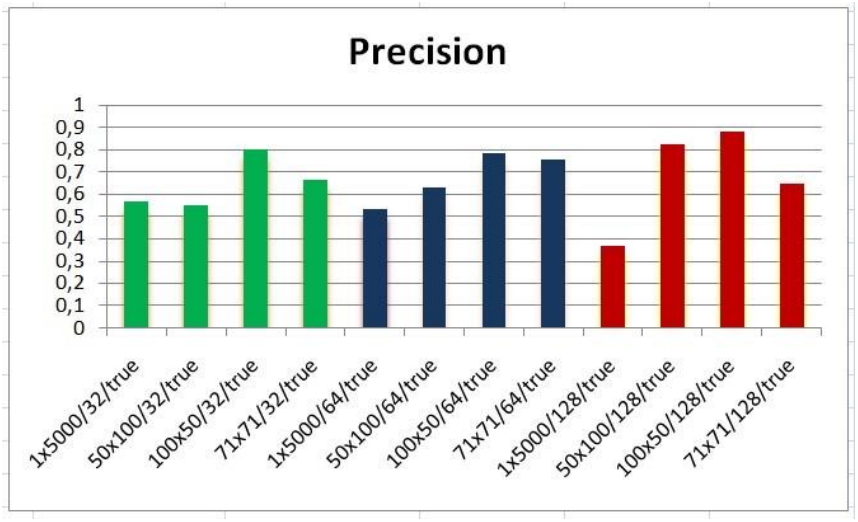

Figure 3.23 : Graphe de précision test N°2 du jeu de données TCGA.rnaseq resamp=true.

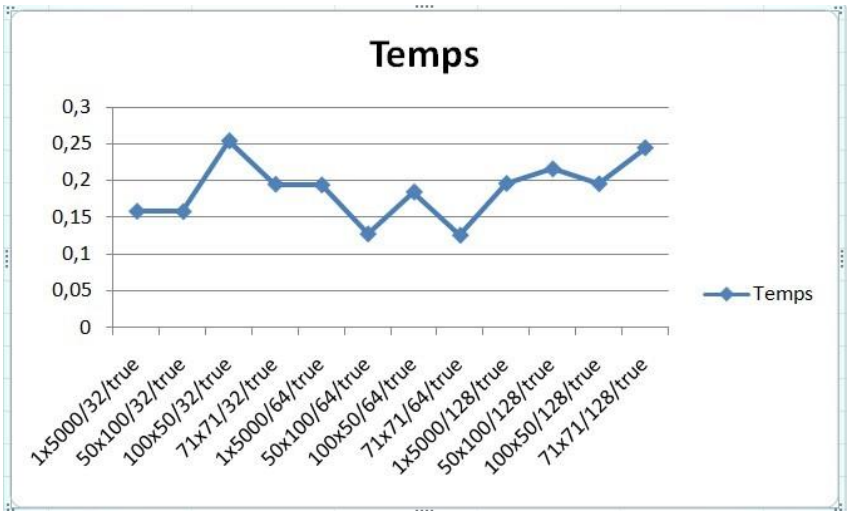

Figure 3.24: Graphe du temps d'exécution test N°2 du jeu de données TCGA.rnaseq resamp=true.

3.6.2.3 Résultats GISETTE :

Le tableau 3.15 présente les résultats du test n°2 du jeu de données GISETTE avec resamp = false.

test	description	Accuracy	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x5000/32/false	0,7157	0,7157	0,7157	0,7157	0,4538	1	5000	5000
2	50x100/32/false	0,8357	0,8357	0,8357	0,8357	0,4498	50	100	5000
3	100x50/32/false	0,8471	0,8471	0,8471	0,8471	0,4361	100	50	5000
5	71x71/32/false	0,8557	0,8557	0,8557	0,8557	1,5191	71	71	5041
6	1x5000/64/false	0,6342	0,6342	0,6342	0,6342	2,7091	1	5000	5000
7	50x100/64/false	0,8557	0,8557	0,8557	0,8557	1,962	50	100	5000
8	100x50/64/false	0,8785	0,8785	0,8785	0,8785	2,6768	100	50	5000
9	71x71/64/false	0,8528	0,8528	0,8528	0,8528	1,3874	71	71	5041
9	1x5000/128/false	0,7557	0,7557	0,7557	0,7557	10,5021	1	5000	5000
10	50x100/128/false	0,8985	0,8985	0,8985	0,8985	5,2336	50	100	5000
11	100x50/128/false	0,8785	0,8785	0,8785	0,8785	5,7442	100	50	5000
12	71x71/128/false	0,8857	0,8857	0,8857	0,8857	10,3877	71	71	5041

Tableau 3.15 : Résultats du testN°2 gisette cadre (32X32, 64x64, 128x128) resamp=false.

Aprés comparaison des résultats de précision comme le démontre la figure 3.25 et du temps d'éxecution démontré par la figure 3.26 on a tiré les observations suivantes:

1. Une bonne précision dans la combinaison (50x100/128/false) mais le temps d'exécution est élevé.
2. Un temps d'exécution énorme qui va causer des problèmes dans notre modèle dans la combinaison (71x71/128/false) et (1 x5000/128/false).
3. une précision qu'on peut choisir dans les combinaisons (100x50/64/false) et (100x50/32/false) et (71x71/32/false) avec un bon temps d'exécution.

donc le choix sera multiples pour ce test : la taille du cadre (100x50) avec la taille d'image 32 ou 64 ; et la taille du cadre (71x71) avec la taille d'image 32.

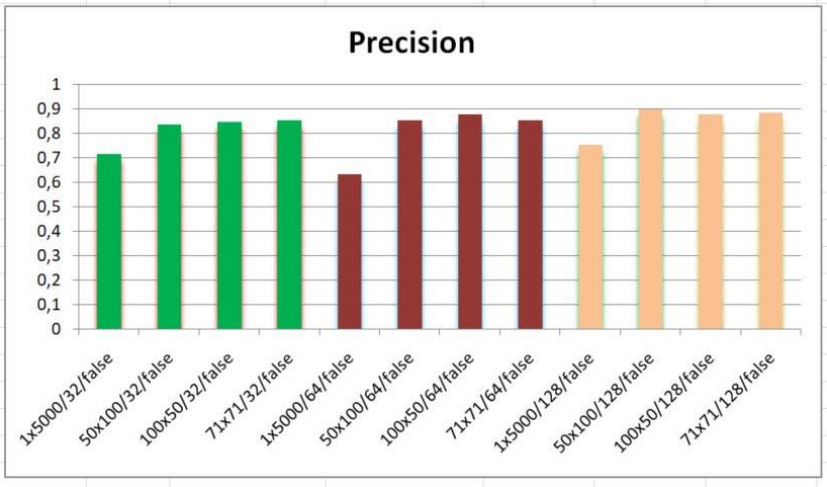

Figure 3.25 : Graphe de précision test N°2 du jeu de données gisette resamp=false.

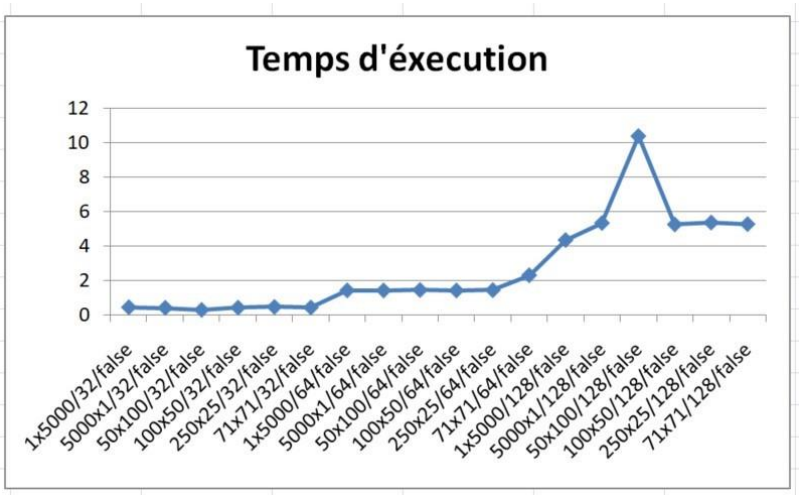

Figure 3.26 : Graphe du temps d'exécution test N°2 du jeu de données gisette resamp=false.

Le tableau 3.16 présente les résultats du test N°2 avec resamp=true

test	description	Accuracy	Precision	Recall	F1-score	Temps	L	H	L x H
1	1x5000/32/true	0,6671	0,6671	0,6671	0,6671	0,8604	1	5000	5000
2	50x100/32/true	0,8500	0,8500	0,8500	0,8500	0,4397	50	100	5000
3	100x50/32/true	0,8443	0,8443	0,8443	0,8443	0,4438	100	50	5000
5	71x71/32/true	0,8286	0,8286	0,8286	0,8286	0,4301	71	71	5041
6	1x5000/64/true	0,6914	0,6914	0,6914	0,6914	1,3996	1	5000	5000
7	50x100/64/true	0,8414	0,8414	0,8414	0,8414	1,4132	50	100	5000
8	100x50/64/true	0,8671	0,8671	0,8671	0,8671	2,7718	100	50	5000
9	71x71/64/true	0,8586	0,8586	0,8586	0,8586	1,394	71	71	5041
9	1x5000/128/true	0,6557	0,6557	0,6557	0,6557	5,263	1	5000	5000
10	50x100/128/true	0,8743	0,8743	0,8743	0,8743	5,8321	50	100	5000
11	100x50/128/true	0,8814	0,8814	0,8814	0,8814	10,3673	100	50	5000
12	71x71/128/true	0,8571	0,8571	0,8571	0,8571	7,8221	71	71	5041

Tableau 3.16 : Résultats du testN°2 gisette cadre (32X32, 64x64, 128x128) resamp=true.

Aprés comparaison des résultats de précision comme le démontre la figure 3.27 et du temps d'éxecution démontré par la figure 3.28 on a tiré les observations suivantes:

1. une meilleure précision dans la combinaison (100x50/64/true) avec un bon temps d'exécution.
2. le même cadre (100x50) avec une taille d'image 128 donne une bonne précision mais avec un temps d'exécution très élevé.

Le choix du cadre (100x50) avec une taille d'image 64 donne de bon résultats pour notre méthode.

On remarque que le cadre (100x50) avec la taille d'image 64 sera choisi avec les deux valeurs du rééchantillonnage car il donne une bonne précision. .

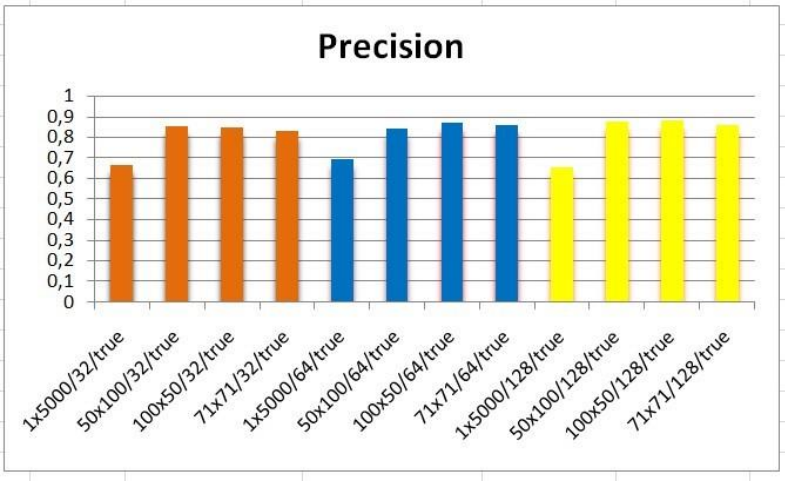

Figure 3.27 : Graphe de précision test N°2 du jeu de données gisette resamp=true.

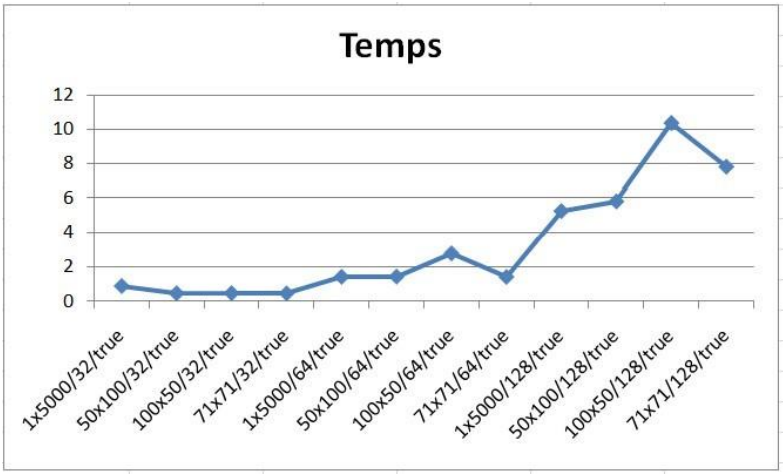

Figure 3.28 : Graphe du temps d'exécution test N°2 du jeu de données gisette resamp=true.

Suite aux résultats du test N°1 et N°2 on a fait les courbes d'apprentissage de chaque ensemble de données : curatedTCGAData(figure 3.29), TCGA.rnaseq(figure 3.30), Gisette (figure 3.31).

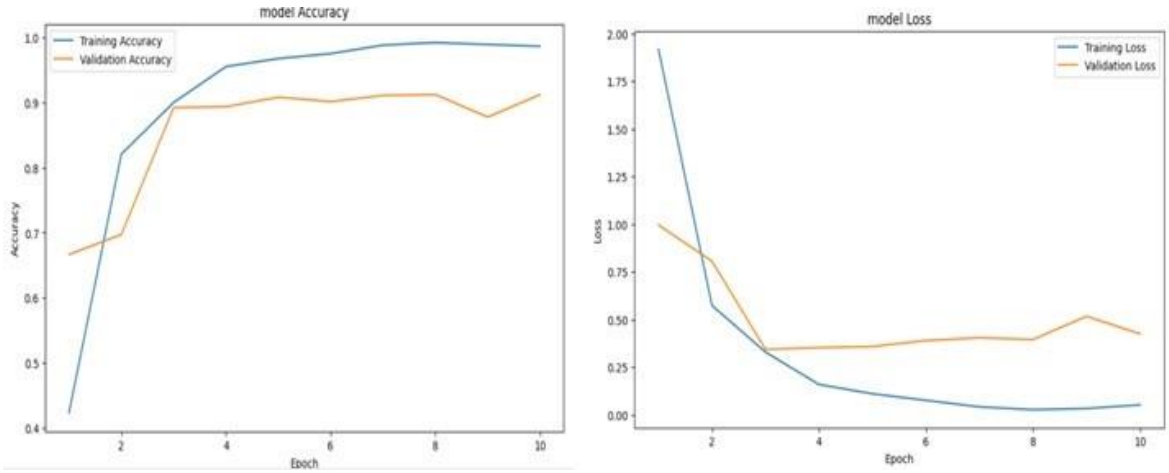

Figure 3.29: la fonction ‘accuracy ‘ et ‘ loss’ pour le jeu de données curatedTCGAData .

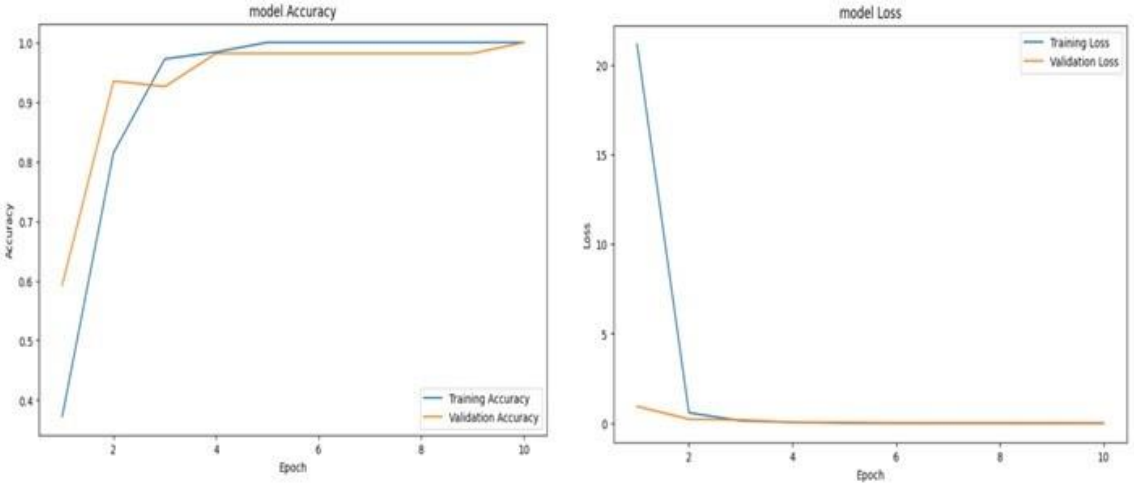

Figure 3.30: la fonction ‘accuracy ‘ et ‘ loss’ pour le jeu de données TCGA.rnaseq .

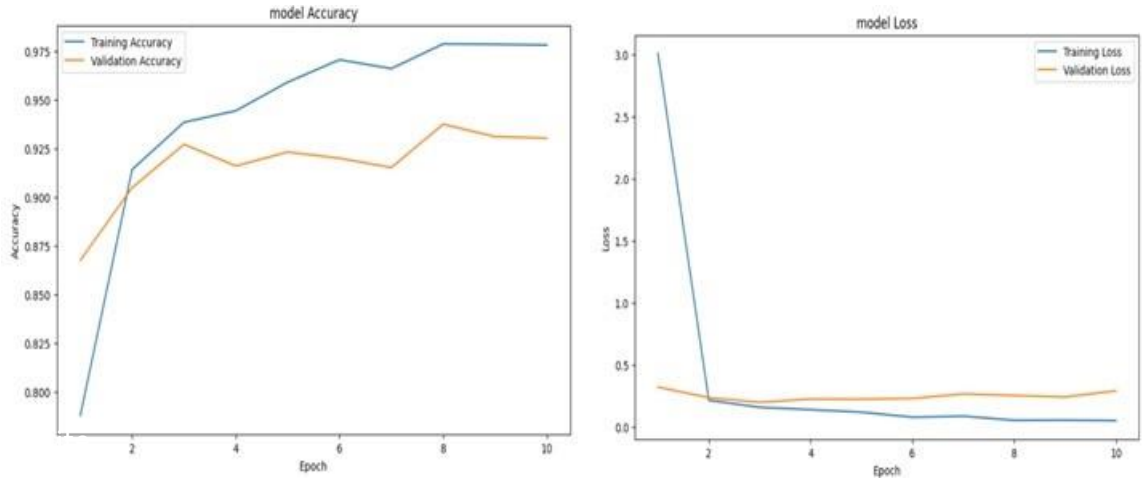

Figure 3.31: la fonction ‘accuracy ‘ et ‘ loss’ pour le jeu de données Gisette .

3.6.3 Test N°3 :

Dans ce test on a essayé de comparer les résultats de notre modèle avec d'autres classificateurs tel que : Random Forest , Arbre de decision et Ada-Boost

Les arbres de décision, AdaBoost et RandomForest sont des algorithmes d'apprentissage automatique populaires utilisés pour la classification et la régression.

1. **Arbres de décision:**
   Les arbres de décision sont des modèles d'apprentissage supervisé qui utilisent une structure arborescente pour prendre des décisions basées sur des règles.
   Chaque nœud de l'arbre représente une caractéristique (ou attribut) du jeu de données, chaque branche représente une valeur possible de cette caractéristique, et chaque feuille représente une classe ou une valeur de prédiction [22].

2. **Ada-Boost (Adaptive Boosting):**
   AdaBoost est un algorithme d'ensemble qui combine plusieurs modèles faibles (généralement des arbres de décision peu profonds) pour former un modèle fort.
   Cet algorithme fonctionne en ajustant itérativement le poids des observations mal classées à chaque étape, ce qui permet de mettre l'accent sur les exemples difficiles et d'améliorer la performance globale du modèle.
   Les modèles faibles sont ensuite pondérés en fonction de leur performance lors de l'agrégation pour former un modèle de prédiction final plus puissant [22].

3. **Random Forest**
   Random Forest est un autre algorithme d'ensemble qui combine plusieurs arbres de décision pour former un modèle robuste et performant.
   Contrairement à Ada-Boost qui met l'accent sur les observations mal classées, Random Forest utilise un processus de bagging (bootstrap aggregating) pour former de multiples arbres de décision sur des sous-ensembles aléatoires du jeu de données [22].

Après comparaison entre notre méthode et les différents classificateurs on a eu les résultats démontré par le tableau 3.17.

méthode	precision curatedTCGAData	précision TCGA.rnaseq	précision gisette
conversion en pseudo-image	0,9211	0,9700	0,9228
Arbre de decision	0,8500	0,8700	0,8900
Random forest	0,9056	0,9134	0,9000
Ada -Boost	0,4952	0,8500	0,8700

Tableau 3.17 : Résultats du testN°3 compaison entre les classificateurs et la conversion en pseudo-image.

Les graphique suivant présente les résultats des tests (précision) pour une comparaison entre notre modèle et les trois classificateurs choisi en utilisant les trois jeux de données et en choisissant pour chaque ensemble de données la meilleurs combinaison qui nous à donner une meilleure précision dans un temps d'exécution optimal .

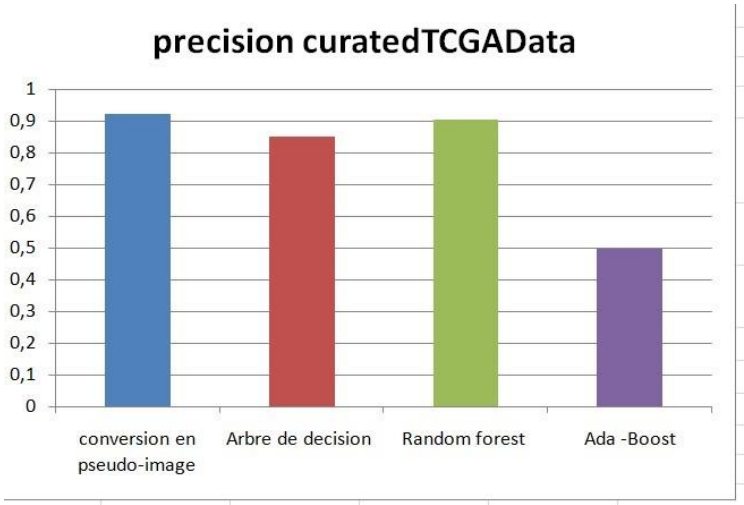

Figure 3.32: Graphe de précision test N°3 du jeu de données curatedTCGAData .

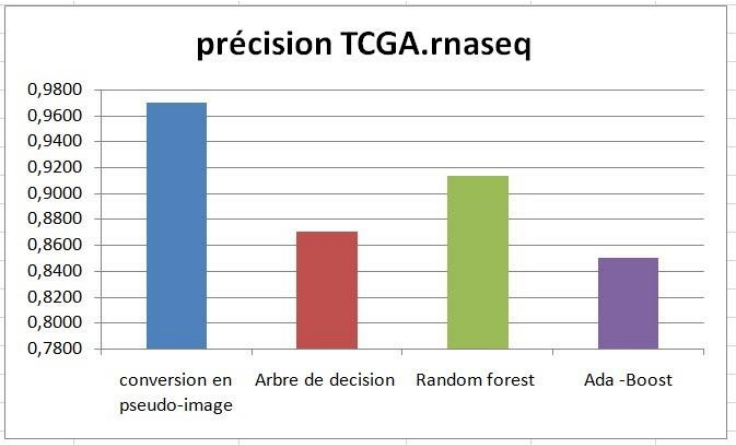

Figure 3.33 : Graphe de précision test N°3 du jeu de données TCGA.rnaseq .

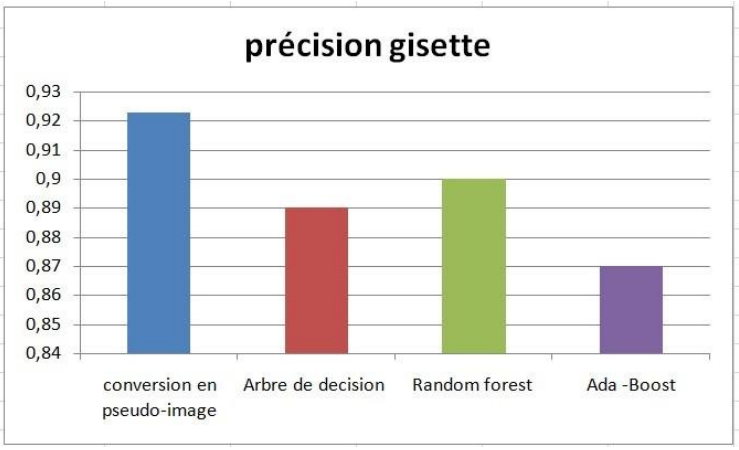

Figure 3.34 : Graphe de précision test N°3 du jeu de données gisette .

les résultats de précision démontrés par les figures 3.32 , 3.33 et 3.34 nous a donnés les observations suivantes:

Notre méthode utilisée pour convertir les données de l'ensemble de données curatedTCGAData en peudo-image a donner de très bons résultats par rapports aux autres classificateurs (comme le montre la figure 3.32).

- ✓ Pour une combinaison de (142x142 /64/false) une taille d'image 64 et un rééchantillonnage de valeur false on a eu une précision de 92% on utilisant notre méthode par contre en utilisant les autres classificateurs on a eu pour : arbre de décision 85% , Rondom Forest 90% , Ada-Boost 49 %
- ✓ Pour l'ensemble de données TCGA.rnaseq on choisisant la combinaison (100x50/128 /true) une taille d'image 128 et un resamp= true on a eu une précision de 97% on utilisant notre méthode (comme le montre la figure 3.33)

  On utilisant les classificateurs on a eu : arbre de décision 87%,Rondom Forest 91 %, Ada-Boost 85 %
- ✓ Pour l'ensemble de données GISETTE on choisisant la combinaison (250x25/64/true) une taille d'image 64 et un resamp= true on a eu une précision de 92% on utilisant notre méthode (comme le montre la figure la figure 3.34)

On utilisant les classificateurs on a eu : arbre de décision 89% ,Rondom Forest 90% , Ada-Boost 87 %

Sur la base de ces résultats, nous avons conclu que notre modèle CNN donne des prédictions meilleures que les autres classificateurs (arbre de décision , Rondom Forest, Ada-Boost).

De plus, la comparaison entre ces modèles utilisant les trois (03) ensembles de données a confirmé que notre méthode de conversion de données en pseudo-image est plus fiable et précis sur des ensembles de données différents.

Discussion :

Comme notre thèse se concentre sur l'impact de la représentation des données non images sur les CNN, nous avons tiré des observations intéressantes des différents tests menés. dans le test N°1, nous avons constaté que la variation de tailles d'images , du cadre d'image et du rééchantillonnage a un impact sur les résultats de chaque jeu de données . pour le jeu de données « curatedTCGAData » qui contient des données énormes avec une tailles très intéressante on a eu une meilleure combinaison (142x142/64/false) alors un cadre carré avec une taille d'image 64 représente mieux cet ensemble de données . par contre pour le jeu de données « TCGA.rnaseq » qui est un sous ensemble du premier avec moins de données on a eu une autre combinaison (100x50 /128 /true), même pour le troisième jeu de données « Gisette » qui est un domaine différent donc une nature de données différente on a eu une autre combinaison (250x25/64/true) .On a continuer nos tests ; le test N°2 c'était sur la conversion des données en images couleurs(RGB) alors une représentation différente avec des résultats différents : une image est créer a partir d'un tableau RGB , le cadre carré(81x81 ou 71x71) et la taille d'image 64 et le rééchantillonnage false ont donner une meilleure précision pour nos jeux de données . on à utiliser les courbes d'apprentissage pour les trois (03) jeux de données curatedTCGAData , TCGA.rnaseq et Gisette pour montrer que notre modèle CNN à une précision d'entraînement idéale et la perte de données est faible et cela était confirmé d'après les résultats obtenus .

Ensuite le test N°3 visait à comparer les résultats de notre modèle avec les classificateurs (Random Forest , Arbre de decision et Ada-Boost) pour prouvez l'efficacité de ce dernier.

Les résultats obtenus ont montré que notre modèle CNN pour les trois (03) jeux de données avais une meilleure précision que les autres classificateurs (92% curatedTCGAData , 97% TCGA.rnaseq , 92% Gisette)

- ✓ 92% curatedTCGAData , pour les autres classificateurs : arbre de décision 85% , Rondom Forest 90% , Ada-Boost 49 %
- ✓ 97% TCGA.rnaseq , pour les autres classificateurs : arbre de décision 87% Rondom Forest 91%, Ada-Boost 85 %
- ✓ 92% Gisette , pour les autres classificateurs : arbre de décision 89% , Rondom Forest 90% , Ada-Boost 87 %

3.7 Conclusion:

Ce chapitre était consacré à la mise en œuvre, aux résultats et à la discussion. Nous avons discuté du protocole de test, qui décrivait les étapes essentielles suivies pour tester notre modèle. Ces étapes comprenaient la préparation des données, la génération de pseudo-images, la configuration CNN, la phase de formation et les tests.

Nous avons effectué un total de trois(03) tests pour confirmer ou réfuter nos hypothèses. Chaque test nous a fourni des renseignements précieux. A partir des tests nous avons choisi la combinaison qui va mieux avec chaque jeu de données alors une meilleure précision dans la représentation des données avec moins de perte.On a remarqué que les images RGB ne donne pas de meilleures résultats que les images niveau de gris , et consomme plus de temps de calcul.

On a conclu après comparaison que notre modèle à fourni des résultats meilleurs que d'autre classificateurs (Random Forest ,Arbre de decision et Ada-Boost).

Conclusion Générale

L'ère du big data a radicalement transformé le paysage de l'analyse des données, introduisant des défis et des opportunités sans précédent. Les méthodes traditionnelles d'analyse, telles que la régression linéaire, les arbres de décision et les techniques de clustering, bien qu'efficaces dans des contextes variés, peinent à gérer la complexité, la richesse et la dimensionnalité élevée des données contemporaines[1]. Dans ce contexte, les réseaux de neurones convolutionnels (CNN) ont émergé comme une solution puissante, offrant des capacités de traitement et d'apprentissage profondément adaptées aux données structurées de type image.

Ce mémoire s'est attelé à explorer le potentiel des CNN pour la classification des données non-image, avec un accent particulier sur les données génomiques. Ces dernières, en raison de leur nature complexe et de leur dimensionnalité élevée, représentent un cas d'étude idéal pour évaluer l'efficacité des CNN au-delà de leur application traditionnelle aux images. Nous avons investigué diverses méthodes de transformation des données non-image en formats compatibles avec les architectures CNN, et nous avons testé ces approches sur des bases de données génomiques.

Nos résultats expérimentaux montrent que, malgré les défis associés à la transformation des données non-image, les CNN peuvent surpasser les méthodes traditionnelles dans certains contextes. Les performances obtenues démontrent que les CNN peuvent effectivement capturer et exploiter les structures complexes présentes dans les données génomiques, offrant ainsi une précision et une robustesse accrues. Cependant, nos études ont également mis en lumière certaines limitations et défis, notamment en termes de prétraitement des données et d'optimisation des architectures CNN pour des types de données spécifiques.

En conclusion, ce mémoire a contribué à élargir les horizons de l'application des CNN, en démontrant leur potentiel pour l'analyse des données non-image, notamment les données génomiques. Nos travaux offrent des perspectives nouvelles et prometteuses pour les chercheurs et les praticiens, en soulignant l'importance d'adapter et d'optimiser les techniques d'apprentissage profond aux caractéristiques uniques des bigdata. Nous espérons que cette étude incitera à une adoption plus large des CNN dans des domaines variés, facilitant ainsi des avancées significatives dans l'analyse et l'exploitation des données complexes et volumineuses.

Les recherches futures pourraient se concentrer sur l'optimisation des méthodes de transformation des données non-image, ainsi que sur le développement de nouvelles architectures CNN spécialement conçues pour ces types de données. Par ailleurs, l'intégration des CNN avec d'autres techniques d'apprentissage machine et d'analyse de données pourrait offrir des solutions hybrides encore plus puissantes et flexibles, adaptées aux défis posés par le big data.

Bibliographie

[1] Faster capital , BIg data analytics, mise à jour 25 avril 2024 https://fastercapital.com/fr/mots-cle/big-data.html.

[2] Cs231n Convolutional Neural Networks for vision recognition , corse web site, https://cs231n.github.io/convolutional-networks/

[3] T. Guo, J. Dong, H. Li, Y. Gao, Simple convolutional neural network on image classification, in: 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), pp. 721–724.

[4] J. Brownlee, Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python, Machine Learning Mastery, 2018. Google-Books-ID: o5qnDwAAQBAJ.

[5] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9.

[6] Marcel Ramos [aut, cre] (), Levi Waldron [ctb], Lucas Schiffer [ctb], Ludwig Geistlinger [ctb], Valerie Obenchain [ctb], Martin Morgan [ctb], Curated Data From The Cancer Genome Atlas (TCGA) as MultiAssayExperiment Objects , Version 1.24.1, February 15, 2024

[7] D. Stutz, Understanding Convolutional Neural Networks, Seminar Report, 2014.

[8] R. K. Singh, M. SivaBalakrishnan, Feature Selection of Gene Expression Data for Cancer Classification: A review, in: 2nd International Symposium on Big Data and Cloud Computing, pp. 52–57.

[9] Sean Davis, M. Ramos, TCGA utility functions for data management, Version 1.23.3, February 17, 2024

[10] S. Indolia, A. K. Goswami, S. P. Mishra, P. Asopa, Conceptual Under standing of Convolutional Neural Network- A Deep Learning Approach, Procedia Computer Science 132 (2018) 679–688.

[11] Anuraganand Sharma, Dinesh Kumar, Non-image Data Classification with Convolutional Neural Networks , Preprint submitted to arXiv.org ,Cs cv 08 July 2020.

[12] Sharma A*, Vans E, Shigemizu D, Boroevich KA, Tsunoda T*, DeepInsight: a methodology to transform a non-image data to an image for convolution neural network architecture, Scientific Reports, 9:11399, pp. 1-7, 2019. https://www.nature.com/articles/s41598-019-47765-6

[13] Sharma A, Lysenko A, Boroevich KA, Vans E, & Tsunoda T. DeepFeature: feature selection in nonimage data using convolutional neural network, Briefings in Bioinformatics, Volume 22, Issue 6, November 2021, bbab297. https://doi.org/10.1093/bib/bbab297

[14] Carolyn Hutter, Robert Kruger,Harmony turk,Jean C.Zenklusen , TGCA Legacy : Multi-Omic Studies in Cancer, September 27-29 , 2018 Washington,DC,USA https://www.cell.com/pb-assets/consortium/pancanceratlas/pancani3/index.html

[15] alok-ai-lab/pyDeepInsight, https://github.com/alok-ai-lab/pyDeepInsight

Bibliographie

[16] UC Irvine Machine learninig Repository, Gisette dataset, https://archive.ics.uci.edu/dataset/170/gisette.

[17] v.poulailleau, Traduction française de la documentation Python , https://github.com/python/python-docs-fr detection in wireless sensor networks: a survey Journal of Network and Computer Applications (2011) Wood, Mark and Erlinger, Michael. Intrusion detection message exchange requirements.

[18] Getting started with Keras , https://keras.io/getting_started/

[19] TensorFlow Core, keras , https://www.tensorflow.org/guide/keras?hl=fr

[20] Scikit-learn machine learning in python, https://scikit-learn.org/stable/

[21] Faster capital, l'importance du prétraitement des données dans le deeplearning pour le Big data, https://fastercapital.com/fr/sujet/l'importance-du-pr%C3%A9traitement-des-donn%C3%A9es-dans-le-deep-learning-pour-le-big-data.html#:~:text=Le%20pr%C3%A9traitement%20des%20donn%C3%A9es%20peut,donn%C3%A9es%20comportant%20des%20valeurs%20manquantes.

[22] Tech Target Le MAGIT , les différences entre arbre de décision , Random forest et gradient Boosting, https://www.lemagit.fr/conseil/Les-differences-entre-arbre-de-decision-Random-Forest-et-Gradient-Boosting

ملخص

مع التطور التكنولوجي الهائل و كمية البيانات الضخمة أصبحت الأساليب التقليدية لتحليل البيانات مثل الانحدار الخطي وأشجار القرار وأساليب التجميع محدودة رغم فاعليتها و ذلك نظرا للعدد الهائل من البيانات الغنية بالمعلومات لا سيما غير المنظمة منها ، المعقدة و التي تتميز بأبعاد عالية . واستجابة لهذه التحديات أظهرت الشبكات العصبية التلافيفية كفاءة ملحوظة في تصنيف البيانات المصورة و بإمكانيات واعدة لتحليل البيانات غير المصورة.تركز هذه الأطروحة على تقييم طرق عرض البيانات غير المصورة في بنية الشبكات العصبية التلافيفية و على وجه الخصوص، نحن مهتمون بتطبيق هذه الشبكات على البيانات الجينية و التي تعتبر أمثلة نموذجية للبيانات الضخمة والغنية بالمعلومات. بحيث يجب تحويل البيانات إلى شكل متوافق مع الشبكات العصبية التلافيفة للاستفادة من قدرات هذه النماذج.أجرينا سلسلة من الاختبارات لتقييم أداء الشبكات العصبية التلافيفة على قواعد البيانات الجينية، و قد أظهرت تجاربنا أنه على الرغم من التحديات الكامنة في تحويل البيانات غير المصورة إلى تنسيقات مكيفة مع هذه الشبكات فمن الممكن تحقيق نتائج جيدة جدا قد تتجاوز تلك الخاصة بالطرق التقليدية .سلسة التجارب و النتائج المتحصل عليها سلطت الضوء على فوائد وقيود مناهج تحويل البيانات المختلفة، و التي ستساهم لا شك في توضيح الرؤى و مساعدة الباحثين والممارسين في هذا المجال على تطوير و تحسين أداء هذه النماذج. تقدم هذه الأطروحة تحليلاً مفصلاً لتقنيات عرض البيانات غير المصورة و التي أظهرت قدرتها على تحقيق نتائج ممتازة متجاوزة الطرق التقليدية لتحليل البيانات و تعقيداتها. نأمل أن تساهم دراستنا في اعتماد الشبكات العصبية التلافيفة على نطاق أوسع لتحليل البيانات الضخمة، وبالتالي فتح آفاق جديدة لتحليل البيانات الضخمة المعقدة و الغنية بالمعلومات .كلمات مفتاحية : البيانات الضخمة ، الشبكات العصبية التلافيفية, الانحدار الخطي ،أشجار القرار، أساليب التجميع

Abstract

With the technological evolution and explosion of big data (big data), traditional methods of data analysis, such as linear regression, decision trees and clustering methods, are facing unprecedented challenges. These methods, while effective in many contexts, show their limitations when dealing with massive amounts of data, especially those that are unstructured, complex, information-rich and characterized by high dimensionality. In response to these challenges, convolutional neural networks (CNN), which have demonstrated remarkable efficiency in classifying image data, have promising potential for the analysis of non-image data. This thesis focuses on the evaluation of non-image data presentation methods in the CNN architecture. In particular, we are interested in the application of CNN to genomic data, which are typical examples of large and information-rich data. The proper transformation of this data into a form compatible with CNN is essential to leverage the capabilities of these models. We conducted a series of tests to evaluate the performance of CNN on genomic databases. Our experiments show that, despite the challenges inherent in transforming non-image data into formats adapted to CNN , it is possible to achieve significant performance, sometimes surpassing those of traditional methods. Our results highlight the benefits and limitations of different data transformation approaches, providing valuable insights for researchers and practitioners in the field. In conclusion, this thesis provides a detailed analysis of non-image data presentation techniques for CNN, demonstrating their potential to overcome the limitations of traditional data analysis methods. We hope that our study will contribute to a wider adoption of CNN for big data applications, thus opening up new perspectives for the analysis of large and complex data.

Keywords : Big data , linear regression, decision trees , clustering ,CNN ,

Résumé

Avec l'évolution technologique et l'explosion des données massives (big data), les méthodes traditionnelles d'analyse des données, telles que la régression linéaire, les arbres de décision et les méthodes de clustering, sont confrontées à des défis sans précédent. Ces méthodes, bien qu'efficaces dans de nombreux contextes, montrent leurs limites lorsqu'elles doivent traiter des quantités massives de données, en particulier celles qui sont non structurées, complexes, riches en informations et caractérisées par une dimensionnalité élevée. En réponse à ces défis, les réseaux de neurones convolutionnels (CNN), qui ont démontré une efficacité remarquable dans la classification des données image, présentent un potentiel prometteur pour l'analyse des données non-image.Ce mémoire se concentre sur l'évaluation des méthodes de présentation des données non-image dans l'architecture des CNN. En particulier, nous nous intéressons à l'application des CNN aux données génomiques, qui sont des exemples typiques de données de grande dimension et riches en informations. La transformation adéquate de ces données en une forme compatible avec les CNN est essentielle pour tirer parti des capacités de ces modèles.Nous avons mené une série de tests pour évaluer les performances des CNN sur des bases de données génomiques. Nos expériences montrent que, malgré les défis inhérents à la transformation des données non-image en formats adaptés aux CNN, il est possible de réaliser des performances significatives, surpassant parfois celles des méthodes traditionnelles. Nos résultats mettent en lumière les avantages et les limitations de différentes approches de transformation des données, offrant des insights précieux pour les chercheurs et les praticiens dans le domaine.En conclusion, ce mémoire fournit une analyse détaillée des techniques de présentation des données non-image pour les CNN, démontrant leur potentiel pour surmonter les limitations des méthodes traditionnelles d'analyse de données. Nous espérons que notre étude contribuera à une adoption plus large des CNN pour des applications de big data, ouvrant ainsi de nouvelles perspectives pour l'analyse des données complexes et de grande dimension.Mots-clés : Big Data , Régression linéaire, , Arbres de décision, Clustering , CNN