UNIVERSITÉ DU QUÉBEC À MONTRÉAL

OUTILS DE CLASSIFICATION TAXONOMIQUE BASÉS SUR DES RÉSEAUX NEURONAUX POUR ANALYSE DE DONNÉES MÉTAGÉNOMIQUES

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

VINCENT THERRIEN

JUILLET 2026

REMERCIEMENTS

Je tiens à remercier Monsieur Vladimir Makarenkov pour ses précieux conseils et son accompagnement tout au long du projet ainsi que Madame Virginie Calderon pour m'avoir fait découvrir la richesse de la bioinformatique.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

ACRONYMES

AA Apprentissage automatique (*Machine Learning*).

AE Autoencodeur.

AP Apprentissage profond (*Deep learning*).

CNN Réseau neuronal convolutionnel (*Convolutional Neural Network*).

DAE Autoencodeur débruiteur (*Denoising autoencoder*).

GAN Réseau neuronal antagoniste génératif (*Generative Adversarial Network*).

IA Intelligence artificielle.

MLP Perceptron multicouche (*Multi-Layer Perceptron*).

ReLU Unité de rectification linéaire (*Rectified Linear Unit*).

RN Réseau neuronal (*Neural Network*).

RNN Réseau neuronal récurrent (*Recurrent Neural Network*).

SSM Modèle à espace d'états (*State Space Model*).

NOTATION

Nombres

a scalaire.

a vecteur.

A matrice.

A tenseur.

Unités

o octet (groupe de 8 bits).

s seconde.

$FLOPS$ Nombre d'opérations sur nombre à virgule flottante par seconde.

Symboles chimiques

A Adénosine.

C Cytosine.

G Guanine.

N Nucléotide non défini.

T Thymine.

Préfixes

k *kilo*, 10^3.

M *méga*, 10^6.

G *giga*, 10^9.

T *tera*, 10^{12}.

Ki *kibi*, 2^{10}.

Mi *mébi*, 2^{20}.

Gi *gibi*, 2^{30}.

Ti *tibi*, 2^{40}.

GLOSSAIRE

bibliothèque Dans le contexte du séquençage d'ADN, désigne un ensemble de fragments d'ADN préparés en vue du séquençage (*library*).

cénancêtre Espèce la plus récente que deux taxons ont en commun. Aussi appelé dernier ancêtre commun ou DAC (*last common ancestor*, *LCA*).

contig Séquence d'ADN continue assemblée à partir de fragments séquencés séparément qui appartenaient à la même séquence avant le processus de séquençage.

hyperparamètre Valeur numérique configurée avant l'entraînement d'un modèle d'apprentissage automatique qui détermine comment ses paramètres sont ajustés durant l'entraînement.

indel Contraction de insertion / délétion, un type de modification de l'ADN qui consiste à substituer ou éliminer une ou plusieurs nucléotides dans une séquence.

lecture Dans le contexte du séquençage d'ADN, désigne une séquence brute d'un fragment d'ADN obtenue avec une technologie de séquençage (*read*).

métagénome Ensemble du matériel génétique d'un microbiome (Roy *et al.*, 2024).

métagénomique Étude des métagénomes (Roy *et al.*, 2024).

microbiome Ensemble de microorganismes vivants dans un habitat donné ainsi que leurs activités et propriétés physio-chimiques (e.g. métabolites, éléments génétiques mobiles, virus, etc.) (Berg *et al.*, 2020).

microbiote Ensemble de microorganismes vivants dans un habitat donné (e.g. bactéries, archées, protistes, fonges, algues, etc.) (Berg *et al.*, 2020).

paramètre Valeur numérique contenue dans un modèle d'apprentissage automatique ajustée durant l'entraînement qui détermine le comportement du modèle lors de l'inférence.

RÉSUMÉ

Le séquençage d'ADN rapide rend possible de nouvelles applications en biologie et en médecine, mais l'analyse d'un volume de plus en plus grand de données génomiques requiert des ressources informatiques et un temps de calcul massifs, d'où l'importance de développer des outils d'analyse efficaces.

La métagénomique étudie le matériel génétique prélevé dans les microbiomes, des communautés de microorganismes qui vivent dans des habitats très variés comme le sol, l'eau, le système digestif humain et d'autres environnements. Une meilleure compréhension des microbiomes est nécessaire pour améliorer notre capacité à les caractériser, ce qui aiderait à détecter les pathogènes plus efficacement ou mieux analyser les sols en agriculture, par exemple. Une méthode répandue pour analyser des métagénomes consiste à comparer des séquences d'ADN prélevées dans un environnement avec des bases de données contenant des génomes de référence. Cette approche est toutefois lente et limitée par la qualité des bases de données. Une technique plus récente et plus flexible consiste à entraîner des réseaux neuronaux pour identifier les espèces présentes dans un métagénome, mais elle requiert un temps d'entraînement important et comporte des difficultés en lien avec l'interprétation des résultats.

Ce mémoire présente de nouvelles techniques de classification taxonomique des séquences d'ADN métagénomique, ce qui consiste à catégoriser les organismes. Les techniques sont basées sur le modèle Mamba, un modèle à espace d'états, ainsi que qu'une variante modifiée par un pré-traitement avec CNN (Mamba-CNN). Sur des ensembles de séquences d'ADN génétiquement diversifiées, Mamba-CNN classifie les données avec une précision moyenne de $88,96\%$, ce qui se compare favorablement à un transformateur de taille similaire ($72,46\%$), le modèle BERTax ($86,88\%$) et les outils MMSeqs 2 ($88,20\%$) ainsi que Kraken 2 ($75,51\%$). La durée d'entraînement des modèles à espace d'états testés est outre moins longue qu'un transformateur de taille similaire, avec $0,164\ s$ pour un pas avec Mamba-CNN contre $0,188\ s$ pour le transformateur avec un nombre similaire de paramètres.

INTRODUCTION

Les technologies de séquençage d'ADN permettent aujourd'hui d'analyser de grandes quantités de matériel génétique. Alors que la première génération de technique de séquençage (à partir des années 1970) était limitée à un faible débit, les technologies de deuxième génération (aussi appelées «nouvelle génération» ou *NGS*, à partir des années 2000) ont grandement augmenté la vitesse de séquençage mais demeuraient limitées à de courtes séquences d'au plus quelques centaines de nucléotides. Les technologies de troisième génération (à partir des années 2010) constituent une autre amélioration qui permet d'analyser des séquences encore plus longues, jusqu'à plusieurs milliers de nucléotides, encore plus rapidement (Satam *et al.*, 2023). Ces technologies ont contribué à améliorer nos connaissances en génomique et mené à des applications en biologie et médecine, par exemple, pour la détection de pathogènes, les diagnostics cliniques, la protection de l'environnement et l'agriculture (Liu *et al.*, 2025). De nombreux outils ont aussi été développés pour analyser les données de séquençage et faciliter le travail des chercheurs (Roy *et al.*, 2024). Ce mémoire s'intéresse en particulier à une application de l'analyse de séquences d'ADN, la classification taxonomique de métagénomes, qui vise à identifier les espèces présentes dans des microbiomes.

La métagénomique se heurte à des problèmes majeurs. Le **volume de données** complique les analyses. Un séquençage conventionnel de métagénome avec des technologies de troisième génération peut générer $10\ Go$ de données par expérience, mais la quantité de données varie selon le type d'expérience (Pérez-Cobas *et al.*, 2020). Les bases de données contiennent aussi beaucoup d'information : RefSeq (Goldfarb *et al.*, 2024) contient des génomes de référence de haute qualité pour plus de $176\ 395$ organismes en avril 2026 et GTDB (Parks *et al.*, 2025) contient un taxonomie exhaustive pour plus de $153\ 000$ espèces de procaryotes. Identifier une espèce en comparant une séquence d'ADN avec chaque séquence d'une base de données de référence entraînerait un temps de calcul inacceptable pour des applications réelles, c'est pourquoi de nombreuses méthodes d'analyse ont été développées pour obtenir des résultats plus rapidement (Roy *et al.*, 2024).

Ce mémoire vise à améliorer des techniques de classification taxonomique de données de séquençage métagénomique, ce qui implique les étapes principales suivantes :

La **génération de données synthétiques** permet de créer des ensembles de données pour entraîner et valider les outils de classification. Les données doivent être générées à partir de séquences de haute qualité

et représenter des échantillons métagénomiques variés pour éviter de surajuster les modèles et d'obtenir des performances non représentatives d'un cas d'utilisation réel.

L'élaboration de **modèles de classification taxonomique** consiste à concevoir des réseaux neuronaux inédits pour améliorer les techniques existantes. Ces modèles doivent offrir des performances supérieures en terme d'utilisation de ressources informatiques (temps de calcul et utilisation de la mémoire) et de classification (précision plus élevée).

L'**entraînement des modèles** consiste à modifier automatiquement les paramètres des modèles pour leur permettre d'effectuer les classifications taxonomiques. Cette étape requiert des ressources intensive et doit s'appuyer sur de bons hyperparamètres pour converger vers une solution efficace. Un entraînement mal configuré peut faire échouer le modèle à apprendre comment résoudre une tâche et ainsi causer un gaspillage de ressources (Liu *et al.*, 2020).

Ce projet consiste à améliorer les techniques d'apprentissage automatique pour classifier des séquences obtenues lors d'expériences de séquençage métagénomique. Après la sélection de modèles existants, le processus d'entraînement est ajusté pour chaque modèle afin de maximiser leurs performances. Les hyperparamètres sont modifiés et différentes techniques d'entraînement sont testées pour déterminer les meilleures manières d'obtenir une classification de haute qualité. Tous les modèles sont ensuite testés avec les mêmes données de test et d'entraînement pour identifier les modèles les plus efficaces. On compare aussi les nouveaux modèles développés dans le cadre de ce projet avec des techniques de classification développées par d'autres équipes de chercheurs pour confirmer que les nouveaux modèles constituent bel et bien des amélioration par rapport aux outils déjà existants.

Ce mémoire présente les contributions principales suivantes :

- Comparaison systématique de différentes architectures de réseaux neuronaux pour la classification taxonomique de données métagénomique, en particulier une comparaison approfondie entre les transformateurs et les modèles à espace d'état (SSM).
- Étude des effets du pré-entraînement et de l'ajustement sur les performances des modèles de classification.
- Étude des effets du prétraitement des données de séquençage (jetonisation et traitement par CNN)

sur les performances des modèles de classification.

- Conception de modèles de réseaux neuronaux inédits spécifiquement développés pour la classification taxonomique.
- Analyse de la consommation de ressources informatique par les modèles d'AA.
- Publication du code source ouvert pour répliquer tous les résultats du projet.

Ce mémoire comporte quatre sections principales. Le chapitre 1 présente un état de l'art des outils d'analyse de données métagénomique ainsi que de l'apprentissage profond, avec un accent particulier sur les réseaux neuronaux dédiés aux analyses de données génomiques. Le chapitre 2 explique la méthodologie du projet : collecte et transformation des données, conception des modèles, entraînement et évaluation. Le chapitre 3 présente les résultats des modèles. Le chapitre 4 discute des résultats et propose des améliorations possibles aux nouveaux modèles.

CHAPITRE 1
ÉTAT DE L'ART

Ce chapitre présente une revue de littérature sur les sujets en lien avec la classification taxonomique de données métagénomiques. Elle discute des technologies de séquençage, bases de données, techniques d'analyses classiques et techniques basées sur l'apprentissage automatique.

1.1 Séquençage d'ADN

Les technologies de séquençage servent à obtenir des séquences de caractères qui indiquent la structure de molécules d'ADN or d'ARN. Le tableau 1.1 présente quelques technologies de séquençage d'ADN (Satam *et al.*, 2023).

Nom de la technologie	Principe de séquençage	Longueur des séquences	Limites
454 pyrosequencing	Synthèse	400 à 1000	Peut contenir des erreurs de délétion et insertion
Ion Torrent	Synthèse	200 à 300	Perte de signal lors du séquençage d'homopolymères
Illumina	Synthèse	36 à 300	Taux d'erreur de séquençage jusqu'à 1% dans certains cas.
SOLiD	Ligation	75	Erreurs de substitution et sous-représentation des régions riches en GC
PacBio SMRT	Synthèse	$10k$ à $25k$	Coût élevé comparé aux autres méthodes
Oxford Nanopore	Détection d'impédance	$10k$ à $30k$	Taux d'erreur de séquençage jusqu'à 15% dans certains cas.

Table 1.1 Survol de technologies de séquençage d'ADN

Il est nécessaire de suivre un protocole expérimental pour séquencer correctement du matériel génétique. Navguire et al (Navgire *et al.*, 2022) présentent les étapes fondamentales suivantes d'un tel protocole :

1. **Collecte d'échantillon** : Le matériel génétique est prélevé dans un environnement à un instant donné et sous des conditions spécifiques. Par exemple, le métagénome de la rhizosphère d'une plante peut être échantillonné à une étape précise de son développement et le microbiome fécal humain peut être échantillonné un temps défini après l'administration d'un vaccin à un patient.
2. **Isolation de l'ADN** : L'échantillon est traité en vue du séquençage, par exemple, en ajoutant des réactifs pour provoquer la lyse des cellules et libérer leur matériel génétique. Dans certaines expériences, comme l'analyse de la diversité d'un microbiome environnemental, il est généralement souhaitable de préserver le maximum de diversité dans le matériel analysé pour éviter de déformer l'abondance de certains organismes. Dans d'autres contextes, comme l'identification d'organismes particulier, il peut être préférable de sélectionner une région spécifique des séquences et d'ignorer le reste du matériel génétique.
3. **Préparation de la bibliothèque** : L'ADN isolé est adapté à une technologie de séquençage particulière, ce qui mène à la création d'une bibliothèque de fragments d'ADN. À la fin de cette étapes, les fragments d'ADN ont des longueurs similaires et comprennent des adapteurs connus aux extrémités $5'$ et $3'$.
4. **Séquençage** : La bibliothèque est traitée par une technologie de séquençage, qui génère des fichiers contenant les lectures des fragments d'ADN. Les données de lecture peuvent ensuite être utilisées pour réaliser une analyse de la composition de l'échantillon. Le séquençage peut prendre quelques minutes pour de petites bibliothèques avec certaines technologies de séquençage comme Oxford Nanopore mais prend généralement plusieurs heures, voire plusieurs jours (Roy *et al.*, 2024).

Il existe deux approches principales pour l'étape 3 (préparation de la bibliothèque). Dans le cadre d'une analyse métagénomique **basée sur les gènes marqueurs** (*marker-gene metagenomics*), des régions spécifiques des métagénomes sont amplifiées et séquencées. Ces régions sont sélectionnées en fonction de leur capacité à identifier des espèces spécifiques. Par exemple, les gènes d'ARN ribosomique 16S ont plusieurs caractéristiques qui les rendent utiles pour la classification : ils comportent des sous-séquences variables et d'autres hautement conservée, ce qui facilite l'identification de liens entre les espèces de bactéries (Johnson *et al.*, 2019). Toutefois, puisque les gènes peuvent être transférés entre certains organismes et que les gènes marqueurs sont relativement courts, cette approche, bien que peu dispendieuse, est limitée à une faible sensitivité (Matchado *et al.*, 2024). Une variante de cette approche consiste à utiliser des séquences plus longues que les gènes marqueurs seuls, ce qui peut améliorer les performances (Benítez-Páez *et al.*, 2022). La seconde approche de préparation d'une bibliothèque est le **séquençage complet du génome** (*whole ge-*

nome sequencing), qui consiste à séquencer tous les fragments présents dans l'échantillon. Cette approche est plus dispendieuse parce qu'elle requiert le séquençage d'un volume beaucoup plus grand de matériel génétique, mais elle permet des analyses métagénomiques plus poussées parce que davantage de données sont disponibles. Un désavantage de cette approche est sa sensibilité à l'ADN dominant dans un échantillon ; par exemple, dans un échantillon de microbiome humain, l'ADN humain peut être surreprésenté et rendre l'ADN microbien plus difficile à analyser (Matchado *et al.*, 2024). Ce phénomène est moins susceptible de survenir dans les analyses basées sur les gènes marqueurs parce que l'ADN humain ne comprend pas les séquences cibles.

1.2 Bases de données

La communauté scientifique maintient des bases de données publiques afin de faciliter la réplication de leurs études et d'améliorer l'état des connaissances en biologie. Cette section en présente quelques-unes qui sont pertinentes pour la classification taxonomique.

1.2.1 Bases de données de séquençage

Les bases de données de séquençage contiennent des séquences biologiques (protéines ou nucléotides) qui décrivent des génomes complets ou simplement des sections de génomes.

- GenBank (Benson *et al.*, 2011) est une base de données ouverte de toutes les séquences de nucléotides publiquement disponibles ainsi que leurs traduction en protéine, produite et maintenue par le NCBI. En 2024, GenBank comprenait plus de $4,7$ milliards de séquences de nucléotides appartenant à plus de $580\ 000$ espèces.
- RefSeq (Goldfarb *et al.*, 2024) est une base de données ouverte de séquences de nucléotides et protéines publiquement disponibles, produite et maintenue par le NCBI. Contrairement à GenBank, RefSeq contient une seule séquence de référence pour chaque molécule biologique et les séquences de RefSeq sont annotées pour leur assigner des informations supplémentaires, comme les fonctions biologiques des séquences. Les séquences de RefSeq sont ainsi moins redondantes et plus fiables que celle de GenBank, mais aussi moins nombreuses. Les données de RefSeq sont enregistrées pour $176\ 395$ organismes en mars 2026.
- Ensembl (Dyer *et al.*, 2025) est un projet qui se concentre principalement sur l'annotation de gé-

nomes de vertébrés. Au lieu d'effectuer des annotations sur des séquences biologiques comme RefSeq, Ensembl les effectue plutôt à même des génomes d'organismes. La base de données contient les génomes de 4700 espèces en mars 2026.

1.2.2 Bases de données taxonomiques

Les bases de données taxonomiques décrivent les liens entre des organismes, ce qui permet d'assigner des organismes à des groupes particuliers. Il existe plusieurs manières de classifier les organismes. Une taxonomie consiste à catégoriser hiérarchiquement et nommer des organismes en groupes nommés taxons. Une taxonomie ne reflète pas nécessairement les liens évolutifs réels entre les espèces. Un arbre phylogénomique est une structure hiérarchique qui cherche à identifier les liens évolutifs entre les espèces aussi précisément que possible.

- NCBI Taxonomy (Schoch *et al.*, 2020) est une base de données taxonomique maintenue par le NCBI. Elle classifie hiérarchiquement tous les organismes ayant au moins une séquence enregistrée dans une base de données du INSDC, qui comprend GenBank. NCBI Taxonomy ne constitue pas un arbre phylogénétique ; il est plutôt un outil de classification taxonomique qui aide à organiser les espèces mais ne représente pas exactement leurs liens évolutifs. Par conséquent, certaines séquences ne sont pas attribuées à des nœuds particuliers dans la hiérarchie taxonomique, ce qui rend leurs relations avec les autres espèces peu claires. La base de données est régulièrement mise à jour, ce qui peut entraîner des problèmes de compatibilité de version entre certains outils.
- GTDB (Parks *et al.*, 2025) est une base de données phylogénétique moderne pour les bactérie et archées maintenue par le Australian Centre for Ecogenomics et l'Université de Queensland. GTDB normalise les rangs taxonomiques par distance évolutive, ce qui signifie que des taxons d'un même niveau taxonomique comprennent une diversité génétique similaire. Cette caractéristique est bénéfique pour les outils de classification parce que la base de données comprend moins d'anomalies que NCBI Taxonomy. Toutefois, GTDB est limité à deux domaines du vivant et ne contient aucune information sur les eucaryotes et les virus.

1.3 Analyse des séquences

Les analyses métagénomiques se divisent en deux principaux types : les analyses taxonomiques et les analyses fonctionnelles.

1.3.1 Analyses taxonomiques

Les analyses taxonomiques consistent à comparer les lectures d'ADN à des bases de données de référence pour identifier les organismes présents dans un échantillon (Navgire *et al.*, 2022). Plus particulièrement, le profilage vise à estimer l'abondance relative des organismes présents dans un échantillon et la classification, à identifier le plus précisément possible les espèces présentes dans un échantillon. La classification peut être effectuée avec des lectures ou encore avec des contigs élaborées à partir des lectures brutes. En général, les analyses taxonomiques s'appuient généralement sur les séquences d'ADN seules et n'utilisent pas d'annotations fonctionnelles. Les analyses taxonomiques peuvent s'appuyer sur plusieurs outils.

1.3.1.1 Kraken2

Kraken (Wood *et al.*, 2014) est une suite d'outils de classification, quantification et visualisation métagénomique. La suite inclut quatre logiciels (Lu *et al.*, 2022) :

- **Kraken** classifie les lectures d'ADN à partir d'une base de données de K-mers en prenant, en général, $K = 31$. Les K-mers de l'échantillon sont comparés à la base de données pour identifier des correspondances exactes et déterminer le cénancêtre des génomes qui contiennent les mêmes K-mers. L'avantage de cet algorithme est que même si une espèce présente dans l'échantillon n'est pas représentée dans la base de donnée, il est possible de lui assigner groupe taxonomique plus général (ex. la famille au lieu du genre) (Wood *et al.*, 2014). Néanmoins, comme Kraken emmagasine des K-mers entier, il requiert beaucoup de mémoire et offre de faibles performances (Lu *et al.*, 2022).
- **KrakenUniq** est une variante de Kraken qui distingue (1) les correspondances entre les lectures de l'échantillon avec la base de données et (2) les correspondances uniques de K-mers. Cela lui permet de consider la couverture de plusieurs régions différentes des génomes de référence et évite les biais en faveur des régions hautement conservées, mais augmente la quantité de mémoire à utiliser (Lu *et al.*, 2022).
- **Kraken 2** modifie l'algorithme original de Kraken pour utiliser une correspondance statistique entre

les lectures et la base de données au lieu de correspondances exactes, ce qui entraîne deux bénéfices : (1) Kraken 2 emmagasine un sous-ensemble de K-mers, ce qui diminue la taille de la base de données de 85%, et (2) Kraken 2 est plus résilient aux erreurs de séquençages et aux indels. Cette variante est recommandée pour les analyses de microbiome (Lu *et al.*, 2022).

- **Kraken2Uniq** combine les correspondances statistiques de Kraken 2 et le traçage de régions distinctes de KrakenUniq. Cette variante est recommandée pour l'identification de pathogènes parce qu'elle s'avère plus sensible à des espèces précises mais requiert des ressources informatiques plus grandes (Lu *et al.*, 2022).

La suite logicielle Kraken contient par ailleurs KrakenTools et Pavian, des outils d'analyse statistique et de visualisation des résultats. Des outils tiers sont généralement requis pour réaliser une analyse complète. Par exemple, avec de courtes lectures d'ADN produites par des appareils de la plateforme Illumina (100 à 250 nucléotide par lecture), il est d'abord nécessaire de former des contigs avec un logiciel comme Bowtie (Langmead et Salzberg, 2012) (Lu *et al.*, 2022).

1.3.1.2 bracken

Bracken (Lu *et al.*, 2017) est un logiciel de profilage taxonomique basé sur Kraken. Kraken n'est pas conçu pour mesurer les abondances relatives de groupes taxonomiques présents dans un échantillon ; il sert plutôt à identifier des espèces précises. Comme plusieurs séquences d'ADN sont communes à plusieurs espèces à cause de régions hautement conservées ou de transferts horizontaux, Kraken assigne aux lectures correspondantes des taxons génériques, ce qui rend les résultats impropres à une analyse d'abondance. Bracken modifie les résultats de Kraken en redistribuant les lectures à travers un arbre phylogénétique par une méthode probabiliste, ce qui les rend apte à une analyse d'abondances.

1.3.1.3 MetaPhlAn 3

MetaPhlAn 3 (Beghini *et al.*, 2021) est un logiciel de classification taxonomique qui utilise une approche basée sur les gènes spécifiques à des groupes d'organismes, c'est-à-dire des gènes qui se trouvent universellement dans un groupe taxonomique donné et qui sont universellement absents à l'extérieur de ce groupe. Au lieu d'indexer des génomes entiers, MetaPhlAn 3 aligne uniquement les gènes marqueurs avec une base de données de référence d'environ $1,1$ millions de gènes en utilisant Bowtie. Cette approche permet d'assi-

gner avec une grande précision les lectures contenant des gènes marqueurs à un groupe taxonomique, mais les lectures ne comprenant pas de tels gènes (la majorité des lectures, dans la plupart des cas) ne peuvent pas être assigné à un taxon.

1.3.1.4 mOTUs3

mOTUs3 (Ruscheweyh *et al.*, 2021) est un logiciel de classification taxonomique qui utilise des gènes marqueurs métagénomiques. Ces marqueurs sont définis d'une manière similaire à la base de données utilisée par MetaPhlAn 3, mais au lieu d'être générés exclusivement à partir de génomes de référence, ils sont aussi générés à partir de métagénomes assemblés. Cela permet à mOTUs3 de regrouper des lectures dans des groupes taxonomiques qui n'ont pas de génome de référence officiel (dans un tel cas, Kraken 2 leur assigne un groupe taxonomique plus élevé et MetaPhlAn 3 ne peut pas faire de classification).

1.3.1.5 sourmash

Sourmash (Irber *et al.*, 2024) est un logiciel de profilage taxonomique qui estime les abondances de groupes taxonomiques en comparant des jeux de données compressés. Le logiciel est basé sur l'algorithme de compression avec pertes FracMinHash, qui préserve un sous-ensemble restreint des K-mers présents dans les lectures d'ADN nommé signature. Des signatures peuvent être comparées par similarité de Jaccard sans effectuer de comparaisons intensives des séquences afin d'estimer les compositions taxonomiques d'échantillons.

1.3.1.6 Centrifuge

Centrifuge (Kim *et al.*, 2016) est un logiciel de classification taxonomique qui utilise la transformation de Burrows-Wheeler pour créer une base de données optimisée pour la recherche de séquences et l'indice de Ferragina-Manzini pour optimiser la recherche. Au lieu d'effectuer des correspondances avec des K-mers d'une longueur prédéfinie comme Kraken 2, Centrifuge recherche les correspondances les plus longues possibles entre les lectures séquencées et la base de données de référence.

1.3.1.7 MetaMaps

MetaMaps (Dilthey *et al.*, 2019) est un logiciel de classification taxonomique spécialisé pour les longues séquences. Il fonctionne en deux étapes. MetaMaps utilise d'abord un algorithme de mappage basé sur la minimisation pour approximer les endroits dans une base de données de référence où les lectures pourraient avoir des correspondances. Ensuite, il utilise un algorithme d'espérance-maximisation pour évaluer la qualité des correspondances potentielles et identifier les taxons d'où proviennent les lectures.

1.3.1.8 MMSeqs2

MMSeqs2 (Kallenborn *et al.*, 2025) est une suite logicielle conçue pour la recherche et le groupement rapide dans les bases de données de protéines et de nucléotides. Il utilise trois étapes pour effectuer des alignements efficacement : (1) pré-filtrage des correspondances potentielles par K-mers, (2) alignements sans indels et (3) alignement avec indels de Smith-Waterman. Le module MMSeqs2 Taxonomy permet d'effectuer des classifications taxonomiques en traduisant les lectures d'ADN selon les six cadres de lecture et en comparant des séquences d'acides aminés. Cette approche permet de reconnaître des liens évolutifs entre des séquences qui ont divergé au niveau des séquences de nucléotides mais qui ont conservé la même structure de protéine, ce qui constitue un avantage qui n'est pas accessible aux techniques précédemment présentées.

1.3.1.9 BugSeq

BugSeq (Fan *et al.*, 2021) est un logiciel de classification taxonomique spécialisé pour les lectures nanopores (c-à-d longues séquences). Il combine des outils d'alignement de lectures et de recherche de cénancêtre basé sur un processus bayésien. Contrairement aux autres outils, qui sont livrés comme des programmes autonomes, BugSeq est un pipeline qui cascade plusieurs logiciels.

1.3.1.10 MEGAN et MEGAN-LR

MEGAN et MEGAN-RL (Huson *et al.*, 2018) sont des logiciels d'analyse de métagénome respectivement conçus pour les analyses des lectures courtes et longues. Un peu comme MMSeqs2, ces outils sont basés sur une analyse de séquences de protéines traduites à partir des cadres de lecture des séquences de nucléotides. Un aspect important de MEGAN est sa capacité à corriger les décalage dans les cadres de lecture

pour permettre de corriger les séquences de protéines malgré les indels.

1.3.2 Analyses fonctionnelles

Les analyses fonctionnelles consistent à étudier des fonctions particulières, comme les gènes de résistance aux antimicrobiens (Navgire *et al.*, 2022). Elles se divisent en deux étapes. La prédiction de gènes sert à déterminer quelles séquences d'ADN correspondent à un gène d'intérêt et l'annotation fonctionnelle, à y assigner une fonction particulière. Les analyses fonctionnelles s'appuient sur les séquences mais aussi sur des bases de données qui décrivent les fonctions de gènes. Comme ce mémoire se concentre sur les techniques de classification taxonomique, ces techniques sont ici présentées brièvement.

- BLAST (Altschul *et al.*, 1997) est un logiciel qui trouve des régions de similarité entre des séquences de protéines ou de nucléotides grâce à la minimisation basée sur les K-mers. Il s'agit d'un outil plus ancien et moins efficace que MMSeqs2, mais il demeure largement utilisé, entre autre grâce à son intégration au serveur du NCBI qui permet d'effectuer des requête contre des bases de données.
- DIAMOND (Buchfink *et al.*, 2015) est un logiciel d'alignement adapté aux données métagénomiques qui rapporte un temps d'exécution jusqu'à $20\ 000$ fois plus rapide que BLAST pour les séquences courtes avec un degré de sensibilité similaire à BLAST. Au lieu d'utiliser des K-mers, DIAMOND utilise un double indice, c'est-à-dire que la séquence recherchée ainsi que la base de données sont indexées pour faciliter la recherche de correspondances. DIAMOND est aussi plus résilient aux mutations que BLAST parce qu'il ne recherche pas de correspondances parfaites entre les séquences.

1.4 Apprentissage automatique

L'apprentissage automatique consiste à permettre à un ordinateur d'apprendre à résoudre un problème sans spécifier explicitement de méthodes de résolution (Samuel, 1959), par exemple, en laissant un algorithme de classification d'image améliorer sa précision sans interventions manuelles. Il s'agit d'un domaine vaste de recherche ; cette section en présente seulement les principes de bases, les architectures de réseaux neuronaux les plus utilisées et leur application à la métagénomique.

1.4.1 Principes de base de l'apprentissage automatique

L'apprentissage automatique se divise en cinq principaux types.

1.4.1.1 Apprentissage supervisé

L'apprentissage supervisé consiste à entraîner des modèles en leur fournissant des données étiquetées, c'est-à-dire des données pour lesquelles une valeur cible est connue. Suivant la notation employée par (Vapnik, 1995), on définit :

$$\begin{cases} X = \{x_1, x_2, x_3, ...\}, x_i \in R^n, \text{ Un ensemble de vecteurs à classifier} \\ Y = \{y_1, y_2, y_3, ...\}, \text{ Valeurs cibles pour chaque vecteur à classifier} \\ \alpha, \text{ Un ensemble de paramètres à déterminer} \\ f(x, \alpha), \text{ Une fonction qui cherche à prédire } y_i \\ \hat{Y} = \{\hat{y}_1, \hat{y}_2, \hat{y}_3, ...\}, \text{ Valeurs prédites par } f(x, \alpha) \text{ pour chaque } x_i. \end{cases} \tag{1.1}$$

On suppose que les valeurs cibles sont tirées d'une distribution inconnue $F(y_i|x_i)$. On mesure l'écart entre une valeur cible attendue, y, et une valeur cible prédite, $f(x, \alpha)$, au moyen d'une fonction de perte $L(y, f(x, \alpha))$. Afin de choisir les meilleurs paramètres pour approximer au mieux la distribution $F(y_i|x_i)$, on doit minimiser l'équation 1.2 :

$$\begin{aligned} R(\alpha) &= \int L(y, f(x, \alpha)) dF(x, y) \\ &= \frac{1}{k} \sum_{i=1}^{k} L(y_i, f(x_i, \alpha)) \quad \text{(pour des valeurs discrètes).} \end{aligned} \tag{1.2}$$

Par conséquent, les paramètres optimaux α^* sont obtenus par l'équation 1.3 :

$$\alpha^* = \arg\min_{\alpha} \frac{1}{k} \sum_{i=1}^{k} L(y_i, f(x_i, \alpha)). \tag{1.3}$$

L'apprentissage supervisé vise principalement les problèmes de classification et de régression, pour lesquels les valeurs de Y sont respectivement discrètes ou continues. Cette approche est particulièrement utile en classification de données métagénomiques parce que les bases de données annotées contiennent des séquences d'ADN appartenant à des taxons connus.

1.4.1.2 Apprentissage non supervisé

L'apprentissage supervisé consiste à entraîner des modèles en leur fournissant des données non étiquetées, c'est-à-dire des données pour lesquelles aucune valeur cible n'est connue. Au lieu de minimiser l'écart entre une prédiction et une cible, on cherche les paramètres α qui modélisent les propriétés d'une distribution $F(x)$. Cette approche comprend plusieurs applications (Hasty *et al.*, 2009) ; nous en présentons deux principales.

Le regroupement de données par classes (*data clustering*) consiste à assigner à chaque donnée $X = \{x_1, x_2, ..., x_n\}$ une classe $S = \{s_1, s_2, ..., s_k\}$ en minimisant la variance intra-classe, donnée par l'équation 1.4 :

$$argmin_S \sum_{i=1}^{k} \sum_{x \in s_i} (x - \mu_i)^2, \tag{1.4}$$

où μ_i désigne le centroïde des données dans la classe s_i.

La compression de données vise à minimiser l'erreur de reconstruction entre l'entrée x et une version $\hat{x}$ reconstruite après compression, tel qu'indiqué par l'équation 1.5 :

$$\alpha^* = argmin_\alpha \sum_{i=1}^{n} \left[x_i - g_\alpha(f_\alpha(x_i)) \right]^2 . \tag{1.5}$$

L'apprentissage non supervisé permet d'extraire des propriétés de données non étiquetées, ce qui est le cas pour les bases de données très volumineuses où l'étiquetage serait trop coûteux.

1.4.1.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé combine les approches supervisée et non supervisée pour entraîner des modèles à partir de données étiquetées et non étiquetées, ce qui permet d'atteindre des performances supérieures aux approches prises séparément. Cette approche est intéressante pour la métagénomique, où des bases de données complémentaires non étiquetées peuvent être utilisées. En général, cette approche

minimise une fonction qui comporte une perte supervisée et une perte de régulation non supervisée (Chapelle *et al.*, 2006).

1.4.1.4 Apprentissage auto-supervisé

L'apprentissage auto-supervisé est une phase préparatoire de l'apprentissage supervisé ou non supervisé qui consiste à entraîner un modèle à reconnaître les signaux les plus utiles dans les données, ce qui permet d'augmenter les performances lors de la phase principale d'entraînement (Doersch et Zisserman, 2017).

1.4.1.5 Apprentissage par renforcement

L'apprentissage par renforcement (Sutton et Barto, 2014) diffère des autres approches parce qu'elle consiste à laisser un modèle interagir dynamiquement avec un autre système pour apprendre à résoudre une tâche. Cette approche est peu fréquente en métagénomique parce que les analyses de séquences sont basées sur des données non dynamiques.

1.4.2 Réseaux neuronaux

Les réseaux neuronaux, aussi appelés réseaux de neurones artificiels, sont une famille de modèles d'apprentissage automatique basés sur des unités de traitement nommées neurones artificiels inspirées des neurones biologiques (Goodfellow *et al.*, 2016).

1.4.2.1 Entraînement

L'équation 1.6 montre la modélisation mathématique d'un neurone artificiel :

$$z = f\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{1.6}$$

où z est la valeur de sortie du neurone, x_i est une valeur d'entrée, w_i est un paramètre associé à une entrée nommée *poids*, b est un paramètre nommé *biais* et $f(\cdot)$ est une fonction non linéaire nommée *fonction d'activation*. Un réseau neuronal est composé d'un ensemble de telles unités. La forme la plus simple est le perceptron multicouche, illustré par la figure 1.1.

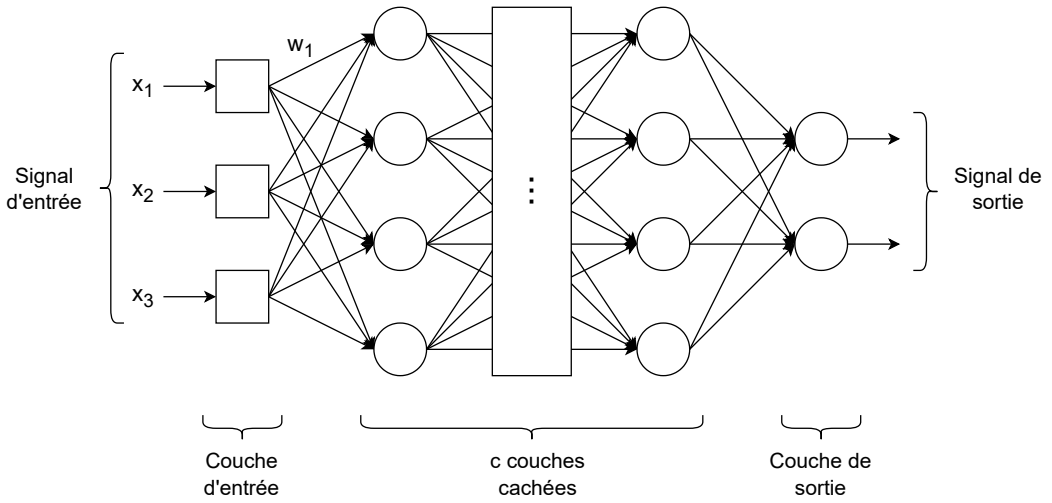

Figure 1.1 Diagramme d'un perceptron multicouche

Chaque cercle dans la figure 1.1 représente un neurone qui calcule l'équation 1.6 en prenant x_i comme une valeur transmise par la couche précédente (représentée par une flèche), w_i comme un paramètre ajustable associé à chaque entrée et b comme un paramètre ajustable distinct des entrées. On dénombre ainsi, pour chaque couche cachée comportant n neurones, n^2 poids et n biais. Un réseau neuronal comportant plusieurs couches cachées est dit *profond* (LeCun *et al.*, 2015). La sortie $\mathbf{h}$ d'une couche de n neurones recevant un vecteur d'entrées $\mathbf{x}$ de m valeurs est :

$$\mathbf{h}_{n\times 1} = f\left(\mathbf{W}_{n\times m}\mathbf{x}_{m\times 1} + \mathbf{b}_{n\times 1}\right). \tag{1.7}$$

L'équation 1.7 calcule la propagation avant du réseau. On compare la sortie finale du réseau avec un résultat cible $\mathbf{y}_{k\times 1}$ au moyen d'une fonction de coût 1.8 qui quantifie l'erreur du réseau :

$$C(y, \hat{y}) = \sum_{i=1}^{k} L(y_i, \hat{y}), \tag{1.8}$$

où $L(\cdot)$ est une fonction de perte, par exemple, $L(y_i, \hat{y}) = (y - \hat{y})^2$. La rétropropagation consiste à calculer l'erreur associée à chaque neurone pour ajuster les paramètres de façon à minimiser le coût. L'erreur $e_i^{(n)}$

du neurone j de la couche n est donnée par l'équation 1.9 :

$$e_i^{(n)} = \frac{\partial C}{\partial h_i^{(n)}}. \tag{1.9}$$

Par la règle de la dérivée en chaîne, on calcule l'erreur associée à un neurone d'une couche précédente avec l'équation 1.10 :

$$e_j^{(n-1)} = f\prime^{(n-1)}(h_j^{(n-1)}) \sum_i w_{ij}^{(n)} e_i^{(n)}. \tag{1.10}$$

On propage l'erreur jusqu'à la couche initiale du réseau. Ce processus se nomme rétropropagation et permet d'obtenir une erreur pour chaque neurone, laquelle sert à ajuster les paramètres par l'équation 1.11 :

$$\begin{aligned} w_{ij}^{(n)} &= w_{ij}^{(n)} - \lambda \cdot e_i^{(n)} x_i^{(n-1)} \\ b_i^{(n)} &= b_i^{(n)} - \lambda \cdot e_i^{(n)}, \end{aligned} \tag{1.11}$$

où $\lambda \in [0, 1]$ est un taux d'apprentissage qui détermine la vitesse à laquelle les paramètres sont mis à jour. λ peut être une valeur fixe ou ajustée avec une inertie entre les mises à jour pour améliorer la stabilité de l'entraînement. On peut mettre les paramètres à jour après chaque propagation, mais il est plus efficace en pratique de calculer et d'additionner les erreurs pour plusieurs échantillons (un « lot » de données) puis de mettre les paramètres à jour en utilisant la somme des erreurs.

1.4.2.2 Architectures

Cette section présente des architectures de réseaux neuronaux développées pour améliorer les performances des perceptrons multicouches.

1.4.2.3 Réseau neuronal convolutionel

Les réseaux neuronaux convolutionels (Goodfellow *et al.*, 2016) (Krizhevsky *et al.*, 2017) utilisent des couches de neurones qui réalisent des opérations de convolution sur des données locales pour en extraire les motifs

les plus pertinents pour une tâche à exécuter. Contrairement aux couches denses des perceptrons multicouches, une couche convolutionnelle compte seulement un biais et $K \times C_e \times C_s$ poids, où K est la largeur du noyau de convolution, C_e est le nombre de canaux d'entrées et C_s est le nombre de canaux de sortie. Une couche convolutionnelle peut définir plusieurs canaux de sortie, ce qui augmente la quantité de données à traiter par une couche dense subséquente.

1.4.2.4 Réseau neuronal résiduel

Les réseaux neuronaux résiduels connectent des couches de neurones à des couches qui les suivent immédiatement mais aussi à des couches davantage en aval dans le réseau ; cette approche facilite l'optimisation des réseaux et peux en augmenter l'efficacité (He *et al.*, 2016). Les connexions résiduelles aident à prévenir la dégradation qui survient lorsque les réseaux comprennent un nombre élevé de couches, ce qui permet d'entraîner des modèles plus profonds.

1.4.2.5 Autoencodeur

Les autoencodeurs (Bank *et al.*, 2023) sont un type de réseaux neuronaux conçus pour encoder des données sous une forme compacte dans un *espace latent* puis de reconstruire des données similaires à celles présentées en entrée. Les autoencodeurs peuvent ainsi apprendre à représenter des données sous une forme *significative*, c'est à dire que la forme comprimée des données comprend les informations les plus importantes pour reconstruire le signal. Cette architecture se prête à l'apprentissage supervisé et non supervisé.

Dans certains contextes, on peut améliorer les performances des autoencodeurs en modifiant l'espace latent. Les autoencodeurs débruiteurs sont entraînés à partir de signaux d'entrés modifiés par du bruit et sont encouragés à retirer le bruit durant la reconstruction du signal, ce qui les aide à identifier les motifs réellement significatifs (Gondara, 2016). Les autoencodeurs variationnels utilisent une distribution statistique au lieu de vecteurs comme espace latent, ce qui les rend particulièrement apte pour la génération de données (Pinheiro Cinelli *et al.*, 2021).

1.4.2.6 Réseau neuronal récurrent

Les réseaux neuronaux récurrents utilisent des connections récurrentes qui gardent en mémoire un *état caché* lors du traitement (Sak *et al.*, 2014). Les architectures décrites jusqu'ici sont dites à action directe

parce que l'information se propage dans une seule direction dans le réseau (la phase de rétropropagation mise à part). Dans un réseau récurrent, les sorties des neurones à un pas dans le temps sont fournies comme entrée au réseau à un pas subséquent, ce qui lui permet d'analyser les dépendances entre les éléments d'une séquence.

1.4.2.7 Transformateur

Les transformateurs forment une famille d'architectures développées à partir de 2017 basées sur l'attention multitête (Vaswani *et al.*, 2017). L'entrée du modèle est une matrice $X_{n \times d_m}$, où n est la longueur de la séquence et d_m est la taille du vecteur de plongement pour chaque élément de la séquence. L'équation 1.12 présente le mécanisme d'auto-attention :

$$\text{Attention}(Q, K, V) = \left(\frac{QK^T}{\sqrt{d_k}} V \right), \tag{1.12}$$

où $Q = XW^Q$, $K = XW^K$ et $V = XW^V$. Les termes W désignent des matrices de paramètres de dimension $(d_m \times d_k)$, où d_k est la dimension de projection. L'équation 1.13 présente le mécanisme d'attention multitête :

$$\text{Attention Multitête}(Q, K, V) = Concat\left(t_1, t_2, ..., t_h\right) W^O, \tag{1.13}$$

où t_i est un mécanisme d'auto-attention tel que réalisé par l'équation 1.12 avec $d_k = \frac{d_m}{h}$ et W^O est une matrice de paramètres. L'intérêt d'utiliser plusieurs têtes d'attention est qu'elles parviennent à se spécialiser durant l'entraînement afin d'analyser différents types de relations, ce qui augmente les performances du réseau (Clark *et al.*, 2019). On forme un bloc transformateur en ajoutant à l'attention multitête des couches dense ainsi que de normalisation et sommation. La figure 1.2 montre une architecture de transformateur complète.

Pour une tâche de classification, on concatène à la séquence d'entrée un élément nommé jeton de classification qui est extrait de la sortie du réseau pour effectuer des prédictions. Pour le pré-entraînement auto-supervisé, la séquence de sortie est comparée à une séquence connue. Par exemple, avec la modé-

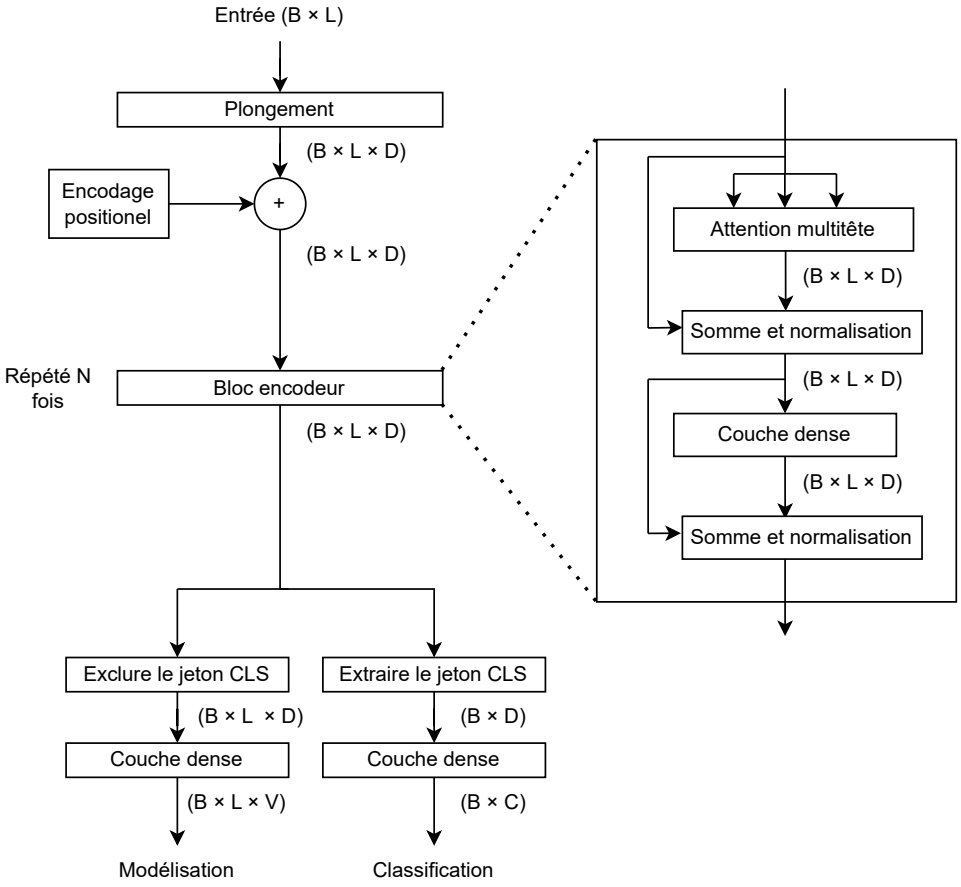

Figure 1.2 Diagramme d'un transformateur encodeur seul

lisation de langue par masque, on remplace aléatoirement des éléments de la séquence d'entrée par des jetons vides et le modèles doit apprendre à les reconstituer.

Les transformateurs sont capables d'analyser les relations entre des éléments éloignés dans une séquence, mais contrairement aux RNN, ils sont plus facilement parallélisables grâce au mécanisme d'attention multitête. L'entraînement demeure toutefois coûteux parce que le modèle compare chaque élément d'une séquence avec tous les autres éléments de la séquence ; le temps de calcul augmente donc quadratiquement, c'est-à-dire en temps $O(n^2)$, où n correspond à la longueur de la séquence d'entrée. Néanmoins, l'entraînement s'appuie principalement sur des multiplications matricielles, qui sont fortement optimisées par les unités de traitement graphique (Gu et Dao, 2024).

Une modification des transformateurs conventionnels est le transformateur linéaire (Katharopoulos *et al.*, 2020), qui utilise la propriété d'associativité de la multiplication matricielle pour modifier l'équation 1.12 et calculer l'auto-attention en temps $O(n)$ au lieu de $O(n^2)$. Cette opération compresse le contexte du transformateur, ce qui les rend plus rapides à entraîner mais diminue leurs performances comparativement aux transformateurs conventionnels.

1.4.2.8 Réseau neuronal à espace d'états

Les réseaux neuronaux à espace d'états (SSM) sont une catégorie de modèles développés à partir de 2021 qui produisent une sortie $y(t)$ à partir d'un espace latent caché $h(t)$ obtenu d'une entrée $x(t)$ (Gu et Dao, 2024) comme le montre l'équation 1.14 :

$$\begin{aligned} h'(t) &= Ah(t) + Bx(t) \\ y(t) &= Ch(t), \end{aligned} \tag{1.14}$$

où A est la matrice d'évolution qui contrôle la sélection des données à préserver dans l'espace latent et B ainsi que C sont des projections d'entrée et de sortie. On discrétise l'équation 1.14 pour obtenir l'équation 1.15 :

$$\begin{aligned} h_t &= \overline{A}h_{t-1} + \overline{B}x_t \\ y_t &= \overline{C}h_t, \end{aligned} \tag{1.15}$$

où la barre horizontale dénote des paramètres discrets obtenus par bloqueur d'ordre zéro. Les performances des SSM sont optimisées par deux mécanismes principaux :

- Pour éviter la récurrence séquentielle (c'est-à-dire le calcul des valeurs h_t qui dépendent de h_{t-1}, comme avec les RNN), les SSM utilisent le scan parallèle (Martin et Cundy, 2018), une technique qui convertit les opérations associatives séquentielles en opérations parallèles. Cela diminue le temps de calcul total mais augmente le nombre total d'opérations. Comme les opérations dans un scan parallèle

doivent être associatives, cette technique ne peut utiliser aucune opération non linéaire, ce qui exclue la plupart des fonctions d'activation.

- Pour réduire les opérations coûteuses de lectures et écritures dans la mémoire, on peut optimiser le transfert de données. Plus spécifiquement, les paramètres $(\overline{A}, \overline{B})$ de l'équation 1.15 (dont la taille est proportionnelle à la longueur de la séquence d'entrée) sont recalculées à partir de leurs paramètres (dont la taille est fixe) au lieu d'être emmagasinés en mémoire, ce qui minimise les transferts de données et accélère le temps de calcul.

Ces optimisations diminuent significativement le coût d'entraînement des SSM par rapport transformateurs parce que le temps de calcul en fonction de la taille de l'entrée augmente linéairement (en temps $O(n)$) au lieu d'augmenter quadratiquement (Dao et Gu, 2024), mais elles excluent toute utilisation de fonctions non linéaires, qui sont pourtant essentielles pour garantir les performances de réseaux neuronaux profonds (Hornik, 1991). La non linéarité doit donc être apportée *autour* d'un bloc de calcul de la fonction SSM. Un modèle basé sur un bloc SSM est le modèle Mamba, illustré par la figure 1.3.

Dans la figure 1.3, la non linéarité est apportée par les fonctions d'activation dans les blocs de projection non linéaires, les fonctions d'activation SiLU et le bloc de convolution. Ces opérateurs mélangent les données et permettent au bloc SSM de mieux les analyser. Les blocs marqués « PL » représentent des projections linéaires. Les auteurs de Mamba (Gu et Dao, 2024) rapportent des performances supérieurs à des transformateurs de taille similaire pour des tâches de modélisation des langues naturelles, analyse de séquences d'ADN et modélisation de données audios. Ces performances s'expliquent par la sélectivité du bloc SSM. En compressant l'espace latent, Mamba sélectionne les données les plus importantes pour la tâche à réaliser, contrairement aux transformateurs linéaires qui le compressent sans discernement. Pour la première fois, un modèle de complexité linéaire (Mamba) peut ainsi égaler un modèle de complexité quadratique (le transformateur conventionnel).

Le modèle Mamba peut être entraîné plus rapidement que les transformateurs, mais puisqu'il est basé sur un espace latent qui compresse le contexte d'une séquence, il ne dispose pas d'un contexte réellement global, contrairement aux transformateurs. D'autres auteurs ont combiné le modèle Mamba à des transformateurs dans le but de combiner l'efficacité de Mamba avec la compréhension des transformateurs (Glorioso *et al.*, 2024).

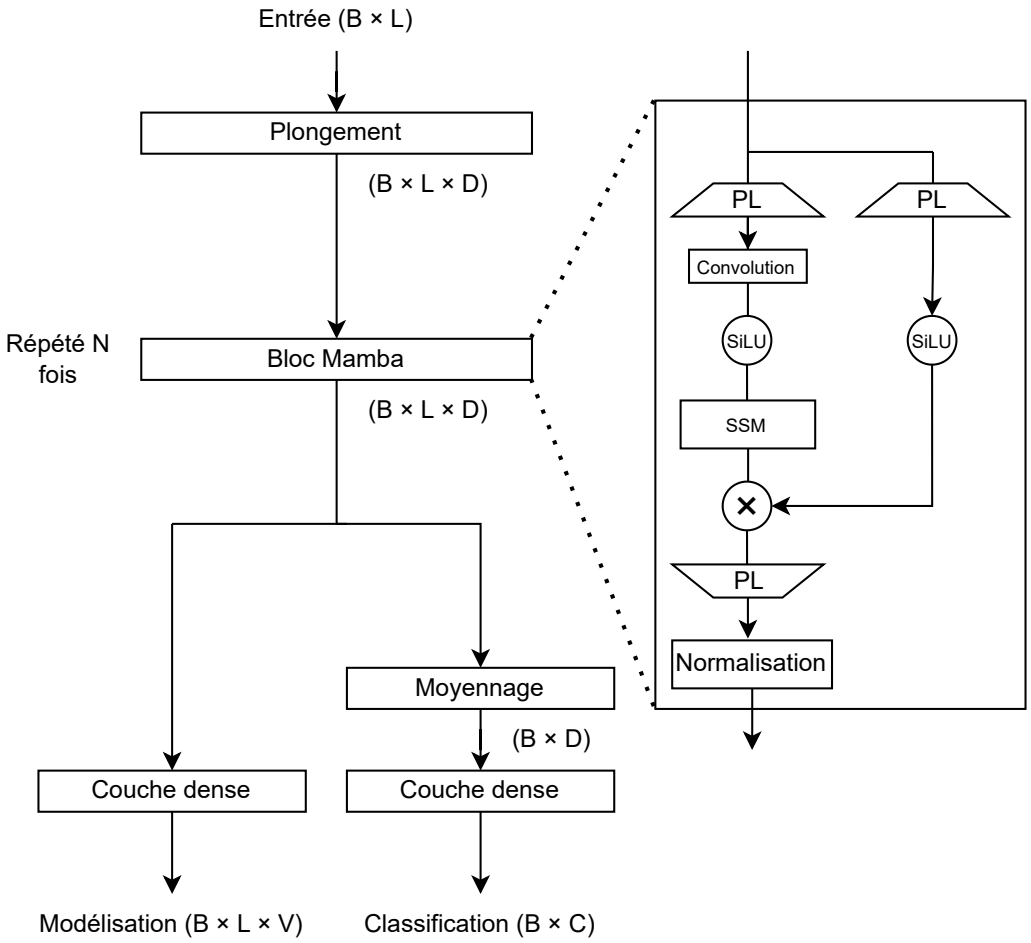

Figure 1.3 Diagramme d'un modèle Mamba

1.4.3 Application à la classification taxonomique

Les réseaux neuronaux ont été appliqués à plusieurs tâches en lien avec la métagénomique (Roy *et al.*, 2024). On peut diviser ces modèles selon leur application à des analyses taxonomique ou fonctionnelle. Les réseaux neuronaux peuvent analyser des lectures ou des contigs directement ou encore utiliser des K-mers extraits de lectures.

1.4.3.1 Analyse taxonomique

Les analyses taxonomiques visent à assigner un taxon à une séquence. Dans le cadre d'expériences métagénomiques, on doit déterminer le taxon appartenant à des séquences sans disposer a priori d'information sur leur taxonomie. Plusieurs modèles ont été développés dans ce but.

Seq2species (Busia *et al.*, 2019) est un réseau neuronal qui utilise des séquence 16S pour classifier des espèces en $13\ 838$ taxons distincts. Le réseau est basé sur une architecture composée de couches convolutionnelles suivies de couches denses. Les séquences de nucléotides sont représentées par un encodage 1 parmi n (*one-hot encoding*), qui consiste à assigner à chaque type de nucléotide un vecteur orthogonal. Le processus d'entraînement modifie aléatoirement certaines données pour simuler l'ajout de bruit dans les séquences. La rétropropagation utilise l'optimiseur ADAM et le calcul de la perte utilise la fonction d'entropie croisée.

DeepMicrobes (Liang *et al.*, 2020) est un modèle composé de quatre composantes principales : un encodage en K-mers, un RNN (LSTM bidirectionnel), un mécanisme d'auto-attention et des couches denses pour produire la sortie. DeepMicrobes n'est pas limité aux séquences 16S et parvient à utiliser toutes les données dans les séquences pour identifier les taxons. DeepMicrobes utilise un encodage basé sur les K-mers : la séquence d'entrée est segmentée en sous-séquences de 12 nucléotides qui sont jetonisées. Comme le vocabulaire résultant serait trop grand (4^16 combinaisons possibles), les jetons sont plongés dans un vecteur de dimension 100. DeepMicrobes dépasse plusieurs techniques de classification taxonomique mais requière de grandes ressources de calcul à cause du RNN et du mécanisme d'auto-attention.

DeepVirFinder (Ren *et al.*, 2020) est un réseau neuronal d'identification de virus dans les métagénomes basé sur les couches convolutionnelles et denses. Le modèle est entraîné avec des séquences tirées de RefSeq. Pour augmenter les données, les auteurs entraînent le modèle avec les séquences originales et leur complément, ce qui améliore les performances du modèle.

CNN-RAI (Karagöz et Nalbantoglu, 2021) est un modèle de classification taxonomique qui utilise des couches convolutionnelles et l'indice d'abondance relative, une mesure qui estime l'importance de K-mers en fonction de leur fréquence dans une séquence en comparaison avec leur fréquence dans la majorité des séquences. Cette mesure est proche de l'indice TF-IDF (*Term Frequency-Inverse Document Frequency*, ou fréquence de terme - fréquence inverse de document), qui suppose qu'une sous-séquence peu fréquente dans

la plupart des séquences mais abondante dans une séquence particulière a davantage d'importance dans ce contexte. Les données pour l'indice d'abondance relative sont générées avec l'outil RAIphy (Nalbantoglu *et al.*, 2011), un programme semi-supervisé de classification de fragments métagénomiques.

RNN-VirSeeker (Liu *et al.*, 2022) est un modèle d'identification des séquences de virus dans les métagénomes. Le modèle est basé sur un RNN et est entraîné sur des séquences de 500 nucléotides tirées de RefSeq. Les auteurs rapportent des performances supérieures au modèle DeepVirFinder pour des tâches d'identification similaires.

BERTax (Mock *et al.*, 2022) est un modèle basé sur BERT, un modèle de transformateur encodeur seul développée pour le traitement des langues naturelles (Devlin *et al.*, 2019). BERTax en reprend l'architecture mais l'adapte à la classification taxonomique. Les séquences d'ADN sont converties en jetons de 3-mers, pour un vocabulaire de taille $4^3 = 64$ et la sortie est un vecteur dont chaque élément correspond à la probabilité d'appartenir à un taxon spécifique. L'étape de pré-entraînement comporte deux volets. (1) La modélisation masquée de langage (*Masked Language Modelling*, *MLM*) consiste à masquer des jetons de la séquence d'entrée et à calculer la capacité du modèle à déterminer la séquence originale. (2) La prédiction de phrase suivante (*Next Sentence Prediction*, *NSP*) consiste à remplacer la seconde moitié d'une séquence par une séquence appartenant à un autre échantillon et de mesurer la capacité du modèle à déterminer si la séquence est tirée d'un même échantillon. Le pré-entraînement n'incorpore aucune donnée taxonomique ; il s'agit d'une phase non supervisée qui permet au modèle d'apprendre une représentation efficace des données d'entraînement. Lors de la phase d'ajustement, le modèle est utilisé pour prédire les taxons auxquels appartiennent les séquences. Ce modèle démontre des performances supérieures à DeepMicrobe et Kraken2, mais demeure moins performance que MMSeqs 2.

1.4.3.2 Analyse fonctionnelle

Les analyses fonctionnelles visent à identifier les fonctions liées à des séquences particulières. Un point d'intérêt particulier pour la communauté scientifique est d'identifier les gènes de résistance aux antimicrobiens (GRA, ou Antimicrobial Resistance Gene, ARG) en raison de leur importance en milieu clinique.

DeepARG (Arango-Argoty *et al.*, 2018) est un modèle d'identification de gènes de résistance aux antimicrobiens, particulièrement pour les antibiotiques. Les modèle est basé sur le perceptron multicouche et utilise les bases de données CARD (Alcock *et al.*, 2023), ARDB (Liu et Pop, 2009) et UniProt (Consortium,

2025) comme références. Les couches cachées du réseau comportent respectivement 2000, 1000, 500 et 100 neurones. La sortie du réseau assigne aux séquences d'entrée une probabilité d'appartenir à un gène de résistance aux antimicrobiens.

ONN4MST (Zha *et al.*, 2022) est un modèle basé sur des réseaux d'ontologie, ce qui lui permet de produire des résultats interprétables. La plupart des réseaux neuronaux fonctionnent en « boîte noire », ce qui veut dire que le processus de génération des résultats est opaque et complique leur applicabilité à des décisions critiques (Rudin, 2019). ONN4MST utilise un réseau expressément conçu pour faciliter l'interprétation des résultats en fournissant des similarités entre des séquences et des microbiomes de composition connue.

DeephageTP (Chu *et al.*, 2022) est un modèle d'identification de protéines spécifiques au bactériophages dans les ensembles de données métagénomiques. DeephageTP utilise des couche convolutionnelles et un ensemble de données tirés de UniProtKB et UniRef, qui font partie de UniProt (Consortium, 2025).

PlasGUN (Fang *et al.*, 2020) est un modèle d'identification de plasmides dans les ensembles de données métagénomiques composées de lectures courtes (de 100 à 900 nucléotides) basé sur les CNN. Les auteurs rapportent des performances supérieures aux techniques classiques d'identification de plasmides.

CHAPITRE 2
MÉTHODOLOGIE

Ce chapitre présente le processus d'élaboration et de validation de modèles de classification de données métagénomiques.

2.1 Bases de données

Les modèles sont entraînés et validés à partir de quatre bases de données.

1. RefSeq (Goldfarb *et al.*, 2024)
2. L'ensemble de données généré par BERTax à partir de RefSeq (Mock *et al.*, 2022)
3. GTDB (Parks *et al.*, 2025)
4. NCBI Taxonomy (Schoch *et al.*, 2020)

2.2 Encodage

Les données tirées de RefSeq sont des fichiers FASTA qui contiennent des chaînes de caractères : A, C, G, T, et N pour les nucléotides incertains. Puisque les réseaux neuronaux prennent en entrée des matrices numériques, il est nécessaire de transformer ces données en un autre format.

2.2.1 Abondance de K-mers

L'encodage d'abondance de K-mer consiste à mesurer la fréquence de K-mers dans une séquence. On obtient un dictionnaire qui associe à chaque K-mer une fréquence relative. Cette représentation entraîne une perte de données en lien avec l'ordonnancement des nucléotides et ne retient que les fréquences relative de sous-séquences. Il s'agit néanmoins d'une représentation utiles pour des modèles simples.

2.2.2 Jetonisation en K-mers

La jetonisation en K-mers consiste à diviser une séquence en sous-séquences de longueurs identiques et à leur assigner un identifiant numérique unique à chaque sous-séquence unique. La taille du vocabulaire

(c'est-à-dire, le nombre d'identifiants différents) équivaut à 4^J, où J est la longueur des sous-séquences. Un désavantage de la jetonisation est qu'elle introduit des artefacts qui ne sont pas présents dans la séquence originale, par exemple, en ajoutant des segments entre les nucléotides.

2.2.3 Jetonisation par encodage de paires d'octets

La jetonisation par encodage de paires d'octets utilise un algorithme pour créer des jetons de tailles variables. Les sous-séquences plus fréquentes peuvent ainsi être représentées avec moins de jetons. Cette approche réduit les artefacts en comparaison avec la jetonisation en K-mers et est répandue dans les grands modèles de langue (HuggingFace, 2026).

2.2.4 1 parmi N

L'encodage 1 parmi N assigne un vecteur orthogonal à chaque type de nucléotides, comme le montre l'équation 2.1 :

$$\begin{aligned} A &= [1, 0, 0, 0] \\ C &= [0, 1, 0, 0] \\ G &= [0, 0, 1, 0] \\ T &= [0, 0, 0, 1]. \end{aligned} \tag{2.1}$$

L'encodage 1 parmi N n'introduit pas d'artefact, contrairement à la jetonisation, mais il requiert davantage de ressources de calcul parce que les données sont non compressées.

2.3 Modèles

Les modèles présentés dans cette section sont validés.

2.3.1 Modèle aléatoire

Le modèle aléatoire consiste à assigner un taxon aléatoire pour chaque séquence. Ce modèle agit comme référence pour les autres modèles.

2.3.2 Forêt aléatoire

La forêt aléatoire (Breiman, 2001) est un modèle d'apprentissage automatique classique (c'est-à-dire non basé sur les réseaux neuronaux). Ce modèle utilise l'encodage d'abondance de K-mers pour classifier les séquences et agit comme une seconde référence pour comparer les performances des modèles plus performants.

2.3.3 Modèles de classification taxonomique classiques

On utilise les modèles Kraken 2 (Lu *et al.*, 2022) et MMSeqs 2 (Kallenborn *et al.*, 2025) pour les comparer avec les méthodes d'apprentissage automatique. Ces modèles construisent une base de donnée de référence à partir des lectures de l'ensemble d'entraînement et leurs performances sont validées avec l'ensemble de tests.

2.3.4 Perceptron multicouche

Le perceptron multicouche classifie les séquences par une architecture composée uniquement de couches denses. Il utilise les hyperparamètres définis par le tableau 2.1.

Nom de l'hyperparamètre	Valeur
Nombre de couches	4
Nombre de neurones par couche	512
Fonction d'activation	ReLU
Taux d'abandon	$0,2$

Table 2.1 Hyperparamètres du perceptron multicouche

2.3.5 Réseau neuronal convolutionnel

Le réseau neuronal convolutionnel classifie les séquences par une architecture composée de couches convolutionnelles suivies de de couches denses. Il utilise les hyperparamètres définis par le tableau 2.2.

Nom de l'hyperparamètre	Valeur
Nombre de couches convolutionnelles	3
Nombre de couches denses	2
Nombre de neurones par couche dense	512
Fonction d'activation	ReLU
Taille du noyau de convolution	9
Enjambée	1

Table 2.2 Hyperparamètres du réseau neuronal convolutionnel

2.3.6 Autoencodeur débruiteur

L'autoencodeur débruiteur classifie les séquences en les compressant dans un espace réduit et en essayant de reconstruire les données d'entrée. Il utilise les hyperparamètres définis par le tableau 2.3. Le bruit ajouté au réseau est tiré d'une distribution normale avec une moyenne de 0 et une variance de $0,1$.

Nom de l'hyperparamètre	Valeur
Nombre de couches convolutionnelles de compression	3
Nombre de couches denses de classification	2
Nombre de neurones par couche dense	512
Nombre de couches convolutionnelles de décompression	1
Nombre de couches denses de reconstruction	2
Fonction d'activation	ReLU
Taille du noyau de convolution	9
Enjambée	1

Table 2.3 Hyperparamètres de l'autoencodeur débruiteur

2.3.7 Transformateur encodeur seul

Le transformateur encodeur seul testé a une architecture définie par les hyperparamètres présentés dans le tableau 2.4. L'architecture est similaire au modèle BERT, un autre transformateur encodeur seul, mais on entraîne ce modèle avec et sans pré-entraînement pour analyser ses performances.

Nom de l'hyperparamètre	Valeur
Taille des jetons	4
Taille du vocabulaire	256
Nombre de couches	4
Nombre de têtes	4
Dimension de plongement	128

Table 2.4 Hyperparamètres du transformateur encodeur seul

2.3.8 Mamba

Le modèle Mamba testé a une architecture définie par les hyperparamètres présentés dans le tableau 2.5.

Nom de l'hyperparamètre	Valeur
Taille des jetons	4
Taille du vocabulaire	256
Nombre de couches	4
Dimension de l'état	16
Facteur d'expansion	2
Taille de convolution	4

Table 2.5 Hyperparamètres du modèle Mamba

La classification finale est obtenue de deux manières en effectuant un moyennage des vecteurs selon la dimension de traitement de Mamba afin de convertir la représentation de l'espace latent en vecteur de classification.

2.3.9 Mamba à flux parallèles

Une particularité de Mamba est son utilisation d'un espace latent pour compresser les données au lieu d'utiliser un contexte entier. Cela améliore les performances du modèle en réduisant la quantité de données à traiter mais empêche aussi le modèle d'avoir accès à toutes les données pertinentes pour la tâche qu'il doit résoudre. Un autre désavantage de cette particularité est que l'échelle temporelle de Mamba a tendance à s'effondrer durant l'entraînement et à considérer principalement un seul niveau de distance entre les données. Par exemple, le modèle pourrait décider d'accorder davantage d'importance à des nucléotides

rapprochées pour favoriser de petits motifs et ainsi ignorer les liens entre les nucléotides éloignées qui pourraient pourtant contribuer à la classification taxonomique.

Pour pallier ce désavantage, on propose une modification de l'architecture Mamba qui force l'utilisation de deux échelles temporelles, une rapide et une lente. Cette contrainte est fixée en restreignant le paramètre dt à des intervalles différents pour chaque flux parallèle. Un élément porte (*gating unit*) sélectionne ensuite les sorties les plus importantes des deux flux parallèles et les fournis à des blocs Mamba subséquents. Après plusieurs tests, on réalise que cette architecture est plus efficace lorsque les flux parallèles sont placés en début du réseau plutôt qu'à la fin ou au milieu. La figure 2.1 présente cette architecture.

2.3.10 Combinaison de CNN avec d'autres modèles

La jetonisation est une technique répandue pour convertir des données séquentielles, comme du texte ou des séquences de nucléotides, en format acceptable pour des réseaux neuronaux. Elle introduit toutefois des artefacts en traitant les séquences, ce qui peut nuire aux performances des réseaux. On présente une méthode alternative de prétraitement des données qui évite l'introduction d'artefacts, le prétraitement par couches convolutionnelles.

La figure 2.2 présente une architecture Mamba modifiée pour utiliser le prétraitement par CNN au lieu de la jetonisation pour préparer les séquences de nucléotides. L'entrée est une séquence de nucléotides en encodage 1 parmi N. Les convolutions permettent de mélanger les données locales pour extraire des motifs utiles. De plus, en utilisant une enjambée (*stride*) supérieure à 1, on peut diminuer la taille des séquences et contrôler la taille de l'entrée pour réduire le temps de calcul.

2.3.11 Combinaison de CNN, Mamba et flux parallèles

On combine le modèle Mamba avec les innovations présentées précédemment, la couche de pré-traitement des données par CNN et le traitement par flux parallèles, pour obtenir un autre modèle.

2.3.12 Mamba avec couche d'attention

Au lieu d'utiliser le moyennage à la sortie du modèle, cette variante utilise une couche d'attention pour tirer profit des relation entre les termes à la sortie du modèle, ce que le moyennage utilisé habituellement avec

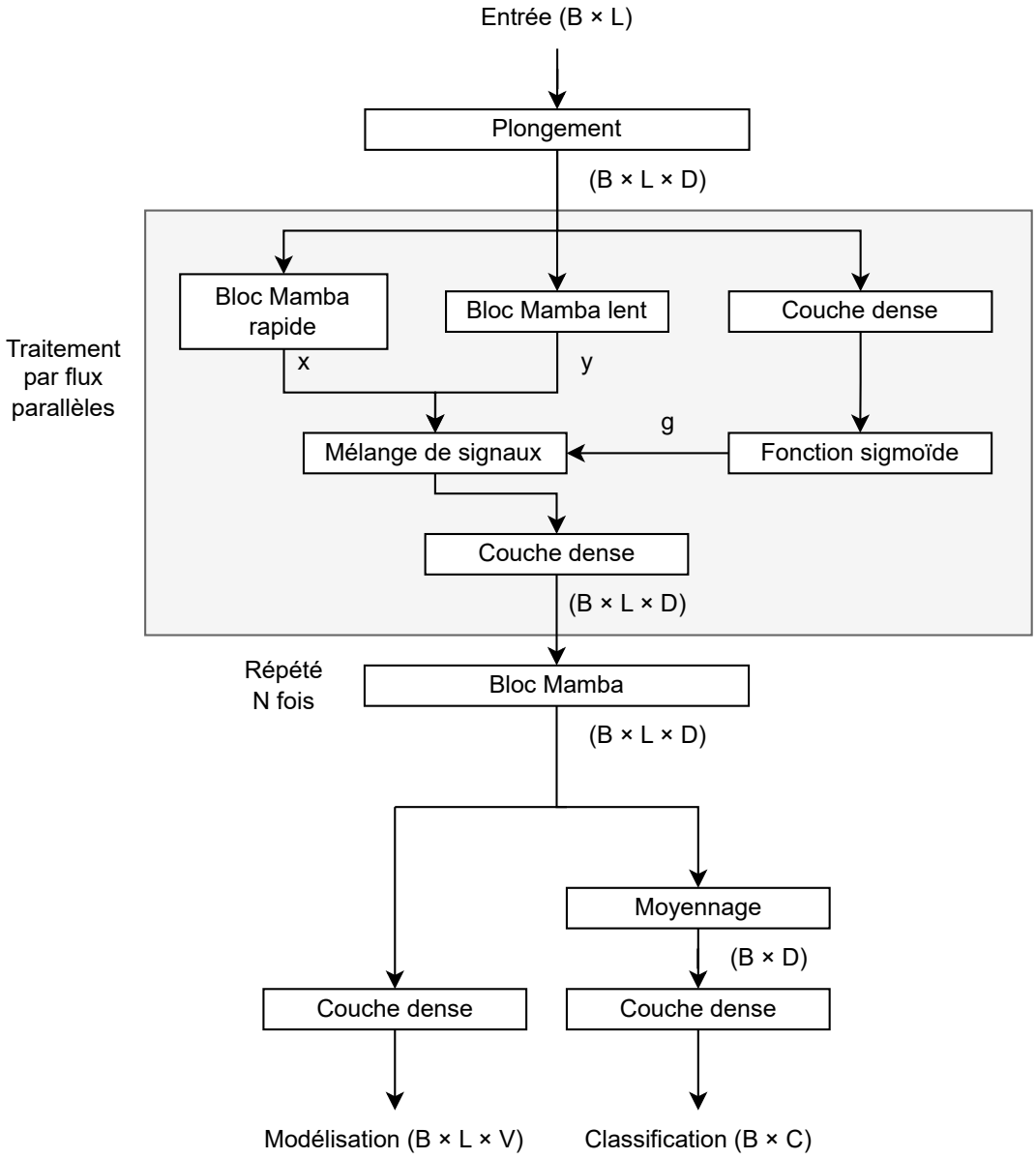

Figure 2.1 Diagramme d'un modèle Mamba avec flux parallèles

le modèle Mamba ne réalise pas. Cette technique se rapproche du modèle Zamba (Glorioso *et al.*, 2024), un hybride entre les modèles SSM et le transformateur.

2.3.13 Comparaison des tailles de modèles

Afin de mesurer la possibilité d'améliorer les performances de Mamba en fonction du nombre de paramètres, on entraîne et teste des modèles qui partagent la même architecture sous-jacente mais qui utilisent

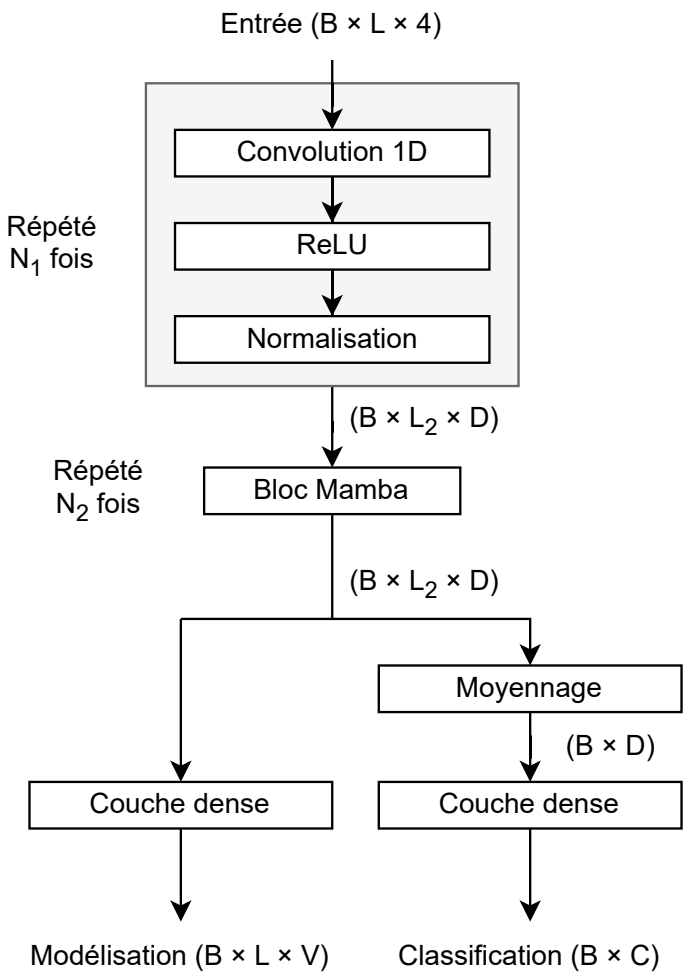

Figure 2.2 Diagramme d'un modèle Mamba avec prétraitement par CNN

un nombre de couches et une dimension interne différentes, comme le montre le tableau 2.6.

Nom de la variante	Nombre de couches	Dimension	Nombre de paramètres
Petit	4	128	553 373
Moyen	6	250	2 586 221
Grand	12	250	5 083 906

Table 2.6 Hyperparamètres de modèles Mamba de différentes tailles

2.4 Génération des données

On utilise les données présentées à la section 2.1 pour générer deux ensembles de données avec lesquels entraîner et tester les modèles.

Le **jeux de données 1** est tiré de BERTax (Mock *et al.*, 2022). Utiliser ce modèle permet de comparer les performances des nouveaux modèles avec ceux développés par d'autres équipes pour contre-valider nos expériences. Les auteurs de BERTax ont élaboré quatre ensembles de données pour faire différents tests ; nous retenons leurs jeux de données dit « final » et « pretraining » pour l'évaluation et le pré-entraînement. Il s'agit d'un ensemble synthétique phylogénétiquement diversifié comprenant 2.708×10^6 lectures d'eucaryotes, 1.903×10^6 lectures de bactéries, 0.535×10^6 lectures d'archées et 0.254×10^6 lectures de virus. Les lectures ont toutes une longueur de 1500 nucléotides. Toutes les lectures sont tirées de RefSeq et la taxonomie est basée sur NCBI Taxonomy.

Le **jeux de données 2** est un ensemble de données original créé à partir de génomes de références disponibles sur RefSeq. On l'élabore pour tester les performances des modèles sur des lectures plus éloignées phylogénétiquement et donc difficiles à prédire. On obtient 695 genres de bactéries et archées selon l'arbre phylogénétique défini par GTDB. Pour chaque genre, on sélectionne 10 génomes de référence sur RefSeq, pour un total de 6950 génomes de référence. Chaque génome de référence correspond à une espèce distincte. Pour chaque génome de référence, on génère 250 lectures synthétiques de 3000 nucléotides. On obtient 1.738×10^6 lectures.

2.5 Séparation des données de test et d'entraînement

Pour le **jeux de données 1**, on divise les données en suivant la même procédure que les auteurs originaux : on assigne un identifiant unique aux 155 genres les plus abondants. Les autres genre sont regroupé dans une classe distincte nommée « autres ». On obtient 156 classes. L'ensemble de tests en créé en échantillonnant aléatoirement 2000 lecture par classe. Dans les lectures restantes, 95% sont assignées à l'ensemble d'entraînement et les 5% restant, à l'ensemble de validation. Puisque la classe « autres » comprend beaucoup plus de lectures que le reste des classes, on la limite à $100\ 000$ lectures pour équilibrer l'ensemble d'entraînement. On applique la même procédure pour séparer les données à un niveau taxonomique plus élevé : on assigne un identifiant unique aux 43 phylums les plus abondants et on regroupe les lectures non assignées à une classe « autres ». pour obtenir 44 classes. On utilise aussi l'ensemble de données de pré-entraînement pour entraîner les modèles semi-supervisés.

Pour le **jeux de données 2**, pour chaque genre (qui comprennent tous 10 génomes de référence), on réserve 3 génomes pour l'ensemble de tests. Les 7 génomes de référence restants sont utilisés pour l'entraînement et la validation, à raison respectivement de 95% et 5% des lectures. On obtient $521\ 250$ lectures de tests,

$1\ 155\ 438$ lectures d'entraînement et $60\ 812$ lectures de validation. Contrairement au jeu de données 1, cette division n'utilise pas de classe « autres ».

2.6 Entraînement

Les modèles sont testés selon une approche supervisée et auto-supervisée.

2.6.1 Entraînement supervisé

Avec une approche **supervisée**, les modèles doivent classifier une séquence d'entrée x dans un taxon cible $y \in [0, C]$, où C désigne le nombre de classes avec $C > 2$. On valide deux fonctions de perte.

La perte d'entropie croisée (EC) (LeCun *et al.*, 2015) est donnée par l'équation 2.2 :

$$EC = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\left(y_{i,j}log(p_{i,j})\right), \tag{2.2}$$

où N est le nombre d'échantillons, C est le nombre de classes, $p_{i,j}$ est la probabilité prédite par le modèle qu'un échantillon i appartienne à la classe j et $y_{i,j}$ est donné par l'équation 2.3 :

$$y_{i,j} = \begin{cases} 1 \ \ si\ i = j \\ 0 \ \ si\ i \neq j. \end{cases} \tag{2.3}$$

La perte focale (PF) est une modification de l'entropie croisée spécifiquement conçue pour les jeux de données déséquilibrées, ce qui signifie que certaines classes comprennent significativement plus d'échantillons que d'autres, par exemple, par un facteur de 1 pour 1000 (Lin *et al.*, 2017). La perte focale pour un problème de classification binaire est donnée par l'équation 2.4 :

$$PF = -\alpha(1 - p_t)^{\gamma}log(p_t), \tag{2.4}$$

où α et γ sont des paramètres d'ajustement et p_t est donné par l'équation 2.5 :

$$p_t = \begin{cases} p & si\ y = 1 \\ 1 - p & si\ y = 0. \end{cases} \tag{2.5}$$

La perte hiérarchique (Wu C, 2019) est une modification de l'entropie croisée qui prend en considération l'organisation des taxons en rangs taxonomiques afin de pénaliser les prédictions éloignées. Par exemple, une prédiction qui attribue à un taxon une espèce différente mais le bon genre sera pénalisé moins fortement qu'une prédiction qui attribue au même taxon une espèce appartenant à un autre genre. En théorie, cette fonction de perte encourage les modèles à tenir compte des liens hiérarchiques entre les classes, mais en pratique, l'équipe de Wu relève qu'elle est surtout utile pour les classes comportant peu d'échantillons.

On utilise l'optimiseur Adam (Kingma et Ba, 2017) et on applique un plafond de 1 aux valeurs des paramètres pour minimiser le surajustement. Pour le jeu de donnée 2, on ajoute un amortissement du taux d'apprentissage pour ralentir la vitesse de modification de paramètres à partir de la dixième époque.

2.6.2 Entraînement auto-supervisé

On utilise la modélisation masquée de langue (*Masked Language Modeling*, *MLM*) pour pré-entraîner les modèles autoencodeur, transformateur et Mamba par une approche auto-supervisée. Les autres modèles ne sont pas conçus pour bénéficier de l'apprentissage auto-supervisé.

Le MLM consiste à remplacer aléatoirement des jetons de l'entrée en jeton spécial dénoté MSK. Le modèle doit déterminer les jetons présents à l'origine dans la séquence. On masque 15% des jetons et on calcule la perte avec la fonction d'entropie croisée. Pour suivre l'amélioration des performances des modèles durant l'entraînement, on calcule l'entropie de la sortie, donnée par l'équation 2.6 :

$$H(p, \hat{p}) = -\sum i = 1^N \sum_{j=1}^{V} p_{i,j} log(\hat{p}_{i,j}), \tag{2.6}$$

où N est le nombre de jetons masqués, V est la taille du vocabulaire (vaut 4^k pour des K-mers), $p_{i,j}$ est la probabilité réelle qu'un jeton masqué i ait la valeur j (vaut 1 pour le vrai jeton et 0 pour les autres) et $\hat{p}_{i,j}$ est la probabilité prédite que le jeton masqué i ait la valeur j. Une mesure liée à l'entropie est la perplexité

$PP = e^H$. La perplexité est plus intuitive parce qu'elle estime le nombre de jetons entre lesquels le modèle hésite.

On teste deux approches de MLM. Avec la technique standard, les jetons sont masqués aléatoirement. Cela convient aux transformateurs parce qu'ils ont un contexte *global* grâce au mécanisme d'auto-attention et parviennent à utiliser les jetons dans toutes la séquence pour déduire les jetons situés à différentes positions. Toutefois, les modèles Mamba sont séquentiels et ne peuvent pas s'appuyer sur les données en aval d'une séquence pour la tâche de MLM. On utilise donc aussi le masquage causal, qui consiste à masquer les derniers jetons de la séquence, ce qui est adapté aux modèles séquentiels.

2.6.3 Prévention du surajustement

Pour l'entraînement supervisé, on utilise l'ensemble de validation pour calculer la perte globale à la fin de chaque époque. L'entraînement est arrêté hâtivement si la perte se dégrade. On utilise une patience de 2 époques pour éviter d'arrêter l'entraînement trop rapidement. On entraîne les modèles pour un maximum de 12 époques avec le jeu de données 1 et un maximum de 16 époques pour le jeu de données 2.

Pour l'entraînement auto-supervisé, on calcule la perplexité à chaque 1000 pas (c'est-à-dire, à chaque rétropropagation) et on arrête hâtivement avec un patience de 2.

2.7 Évaluation

Pour calculer les performances des modèles, on reprend les métriques utilisées par (Portik *et al.*, 2022). Soient une lecture L, un taxon T et une prédiction qui assigne à L un taxon P. On dit d'une prediction qu'elle est :

- un vrai positif, VP, si $L \in T$ et $L \in P$,
- un faux positif, FP, si $L \notin T$ et $L \in P$,
- un vrai négatif, VN, si $L \notin T$ et $L \notin P$ et
- un faux négatif, FN, si $L \in T$ et $L \notin P$.

On définit ensuite les métriques suivantes :

$$\text{Précision} = \frac{\text{Nombre de VP}}{\text{Nombre de VP} + \text{Nombre de FP}}, \tag{2.7}$$

$$\text{Rappel} = \frac{\text{Nombre de VP}}{\text{Nombre de VP} + \text{Nombre de FN}}, \tag{2.8}$$

$$\text{Mesure F} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}. \tag{2.9}$$

Ces métriques s'appliquent à des problèmes de classification binaire. Or, la classification taxonomique est un problème impliquant plusieurs classes. On précise donc des métriques dérivées pour combiner les performances pour chaque classe en les tenant équitablement en compte :

$$\text{Précision}_{macro}\% = \frac{1}{C} \sum_{i=1}^{C} \frac{VP_i}{VP_i + FP_i} \times 100\%, \tag{2.10}$$

$$\text{Rappel}_{macro}\% = \frac{1}{C} \sum_{i=1}^{C} \frac{VP_i}{VP_i + FN_i} \times 100\%, \tag{2.11}$$

$$\text{Mesure F}_{macro}\% = \frac{1}{C} \sum_{i=1}^{C} \frac{2 \times VP_i}{2 \times VP_i + FP_i + FN_i} \times 100\%, \tag{2.12}$$

où C est le nombre de classes et X_i désigne le nombre de VP, FP, FP et FN pour une classe i.

2.8 Logiciel

Un projet de code source ouvert est développé pour entraîner et tester les modèles. Il est publiquement disponible à l'adresse `https://github.com/Vincent-Therrien/stelaro` et comporte deux composantes :

- Une partie dorsal écrit en Rust installe les données des bases du NCBI et de GTDB et les transforme en vue de les préparer à l'entraînement des modèles. Cette composante est conçue pour être résilient aux erreurs.

- Une partie frontale écrite en Python entraîne des modèles d'apprentissage automatique et visualise les résultats.

2.9 Résumé des modèles

La tableau 2.7 présente un résumé des modèles testés.

Nom du modèle	Encodage	Fonctionnement
Modèle aléatoire	N/A	Modèle de base
Forêt aléatoire	K-mers	Ensemble d'arbres décisionnels. Modèle classique utilisé comme référence.
MLP	1 parmi N	Perceptron multicouche, réseau neuronal simple
CNN	1 parmi N	Modèle basé sur la convolution
AED	1 parmi N	Autoencodeur débruiteur, modèle basé sur la compression efficace des séquences
Transformateur	Jetonisation	Transformateur encodeur seul, complexité $O(n^2)$ avec contexte global
Transformateur-CNN	1 parmi N	Identique au transformateur mais séquences pré-traitées par un CNN
Mamba	Jetonisation	Modèle SSM optimisé, complexité $O(n)$ avec contexte compressé dans un espace latent
Mamba-CNN	1 parmi N	Identique au modèle Mamba mais séquences pré-traitées par un CNN
Mamba-Dual	Jetonisation	Modèle Mamba avec flux parallèle pour traiter les données à des échelles distinctes
Mamba-CNN-Dual	1 parmi N	Modèle Mamba avec prétraitement par CNN et flux parallèles

Table 2.7 Résumé des modèles testés

CHAPITRE 3

RÉSULTATS

3.1 Effets du pré-entraînement

Les figures 3.1 et 3.2 présentent respectivement l'entropie observée pour le masque causal et le masque MLM en testant les modèles avec le jeu de données 1.

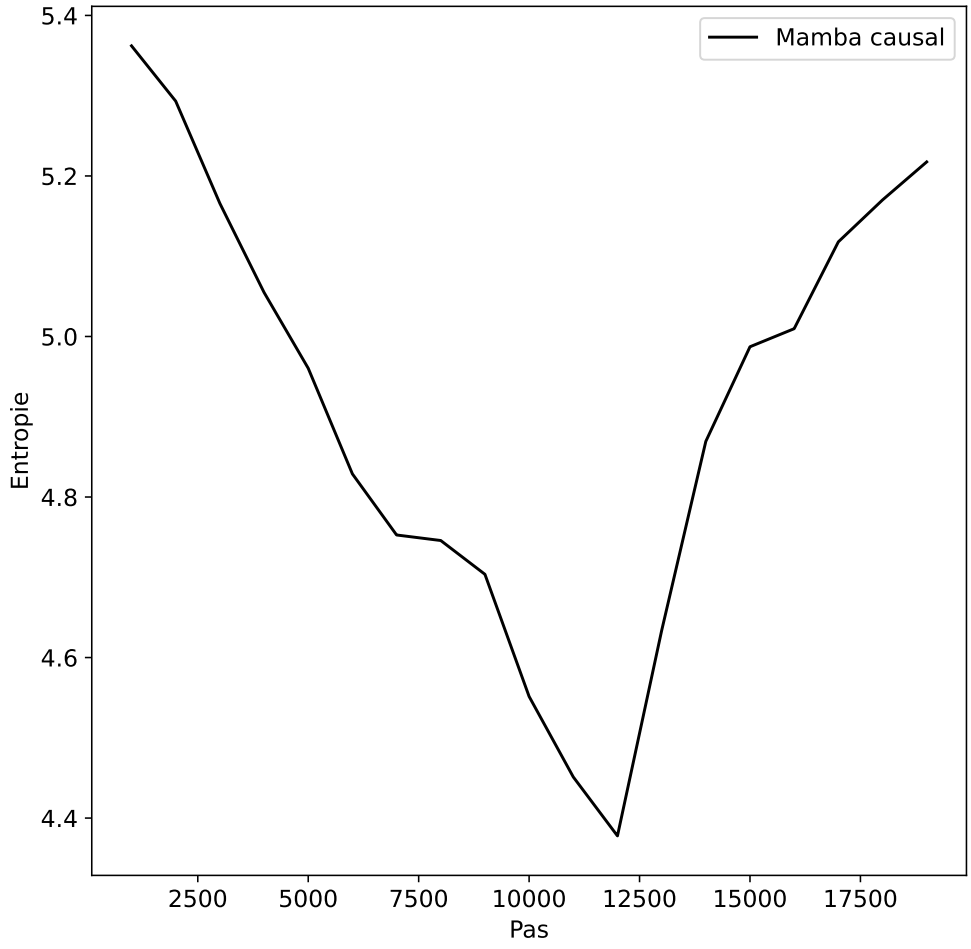

Figure 3.1 Entropie du pré-entraînement causal

La figure 3.3 montre la perte observée pour des modèles pré-entraînés et non pré-entraînés (vierge). On rapporte les résultats pour le modèle Mamba pré-entraîné par MLM.

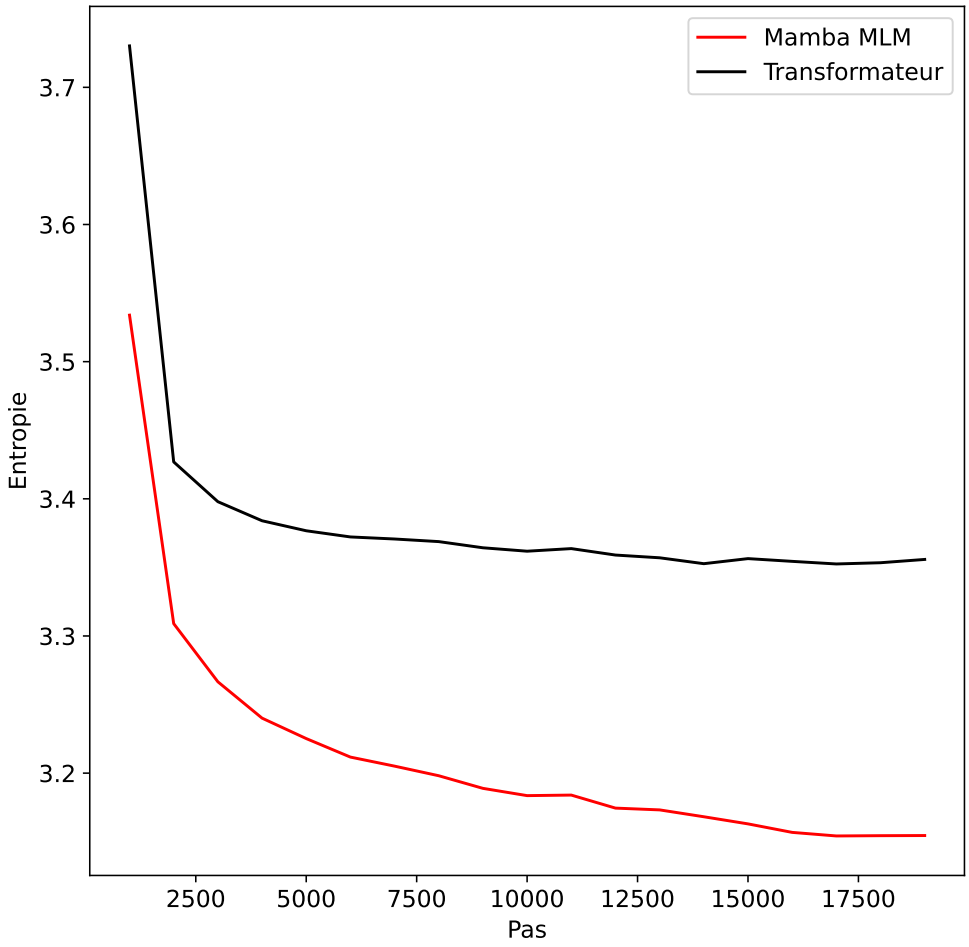

Figure 3.2 Entropie du pré-entraînement MLM

On note que, pour un transformateur de petite taille (moins de 5 millions de paramètres, comme celui testé à la figure 3.3), il est possible d'entraîner le modèle sans pré-entraînement préliminaire. Toutefois, pour des transformateurs de plus grande taille (plus de 10 millions de paramètres), le modèle ne converge pas vers une solution sans pré-entraînement avec le jeu de données 1. Il s'agit donc d'une phase obligatoire pour ces modèles.

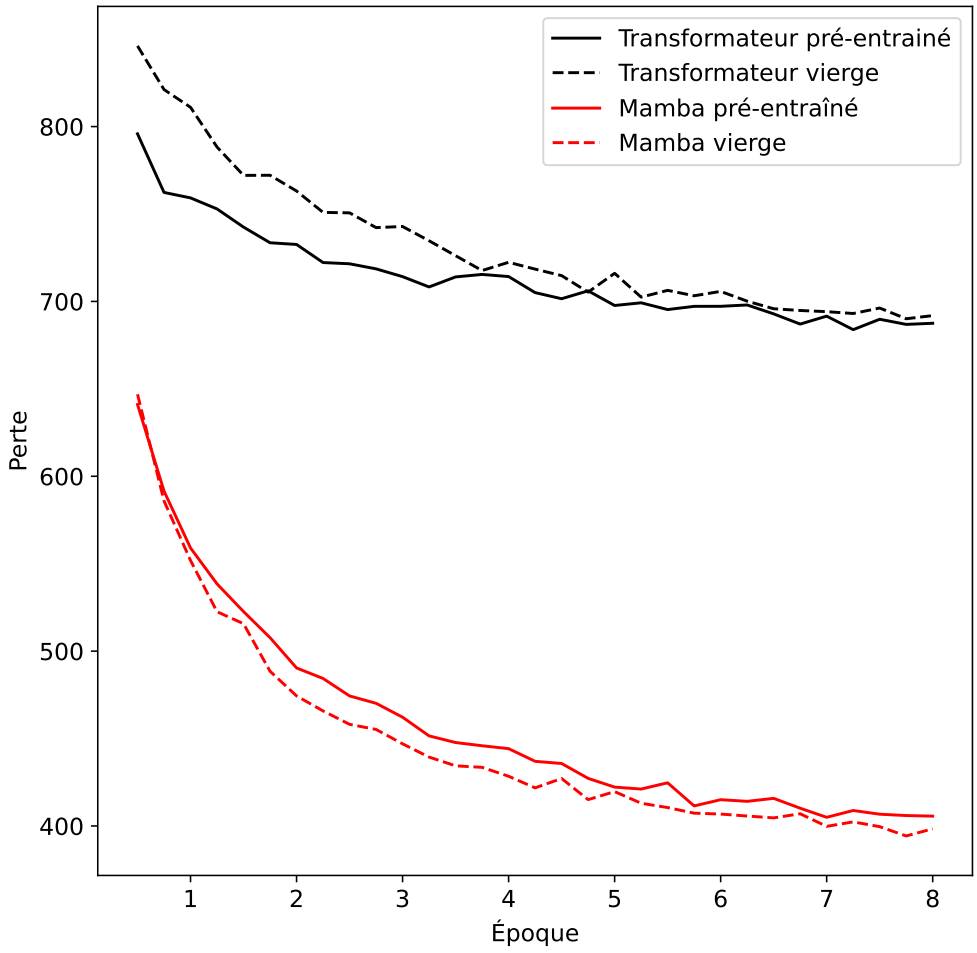

Figure 3.3 Perte observée avec et sans pré-entraînement MLM

3.2 Effets de l'encodage

La figure 3.4 montre la perte observée pour différentes techniques de formatage de l'entrée d'un modèle Mamba. Les tests ont été effectués avec le jeu de données 1.

La figure 3.5 compare la précision observée pour un modèle Mamba qui utilise l'encodage par jetonisation de 3-mer (vocabulaire de 64 jetons) et l'encodage BPE avec un vocabulaire de 8192 jetons. Les résultats sont obtenus avec le jeu de données 1. L'encodage BPE est obtenue à partir des séquences dans le sous-ensemble de pré-entraînement.

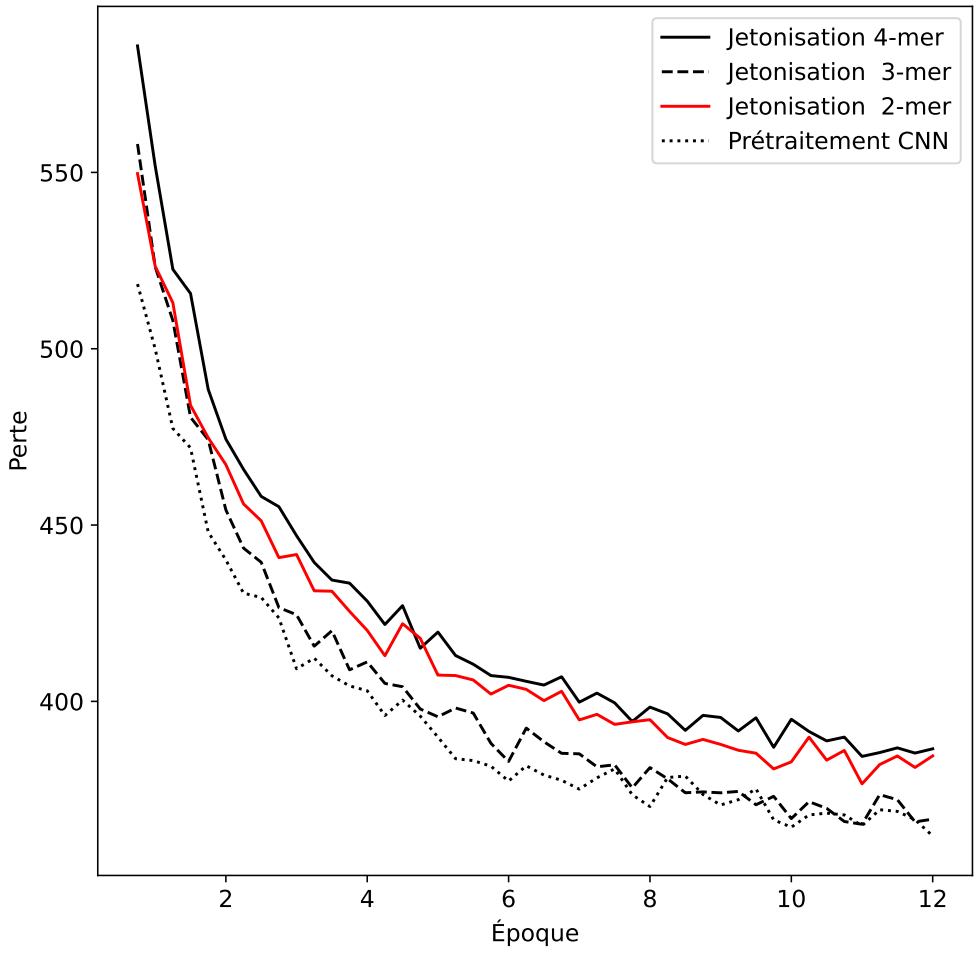

Figure 3.4 Effet de la jetonisation sur la perte

3.3 Effet de la fonction de perte

La figure 3.6 présente la précision observée avec l'ensemble de validation en utilisant le jeu de donnée 1 et un modèle Mamba avec la perte par entropie croisée et la perte focale.

La figure 3.7 présente la précision observée avec l'ensemble de validation avec le jeu de donnée 1 et un modèle Mamba en utilisant la perte par entropie croisée et la perte hiérarchique.

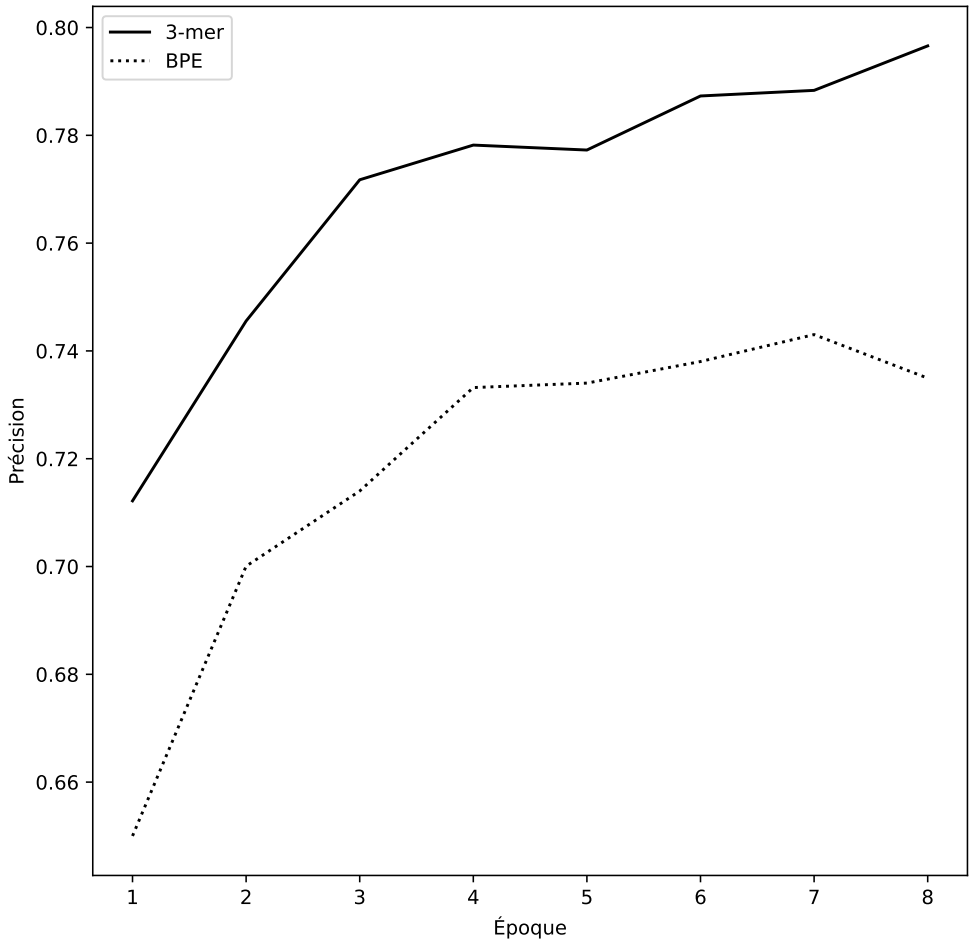

Figure 3.5 Comparaison de la jetonisation BPE avec la jetonisation par 3-mer

3.4 Efficacité des performances

Suivant la norme ISO 25010 :2023 (ISO/CEI, 2023), on entend par l'efficacité des performances deux principales caractéristiques dans le cadre de ce projet :

1. Le comportement dans le temps désigne la durée nécessaire pour entraîner les modèles.
2. L'utilisation de ressources désigne la quantité de mémoire à utiliser et de calculs à effectuer.

Le tableau 3.1 présente l'efficacité des performances pour les modèles entraînés sur le jeu de données 1 avec

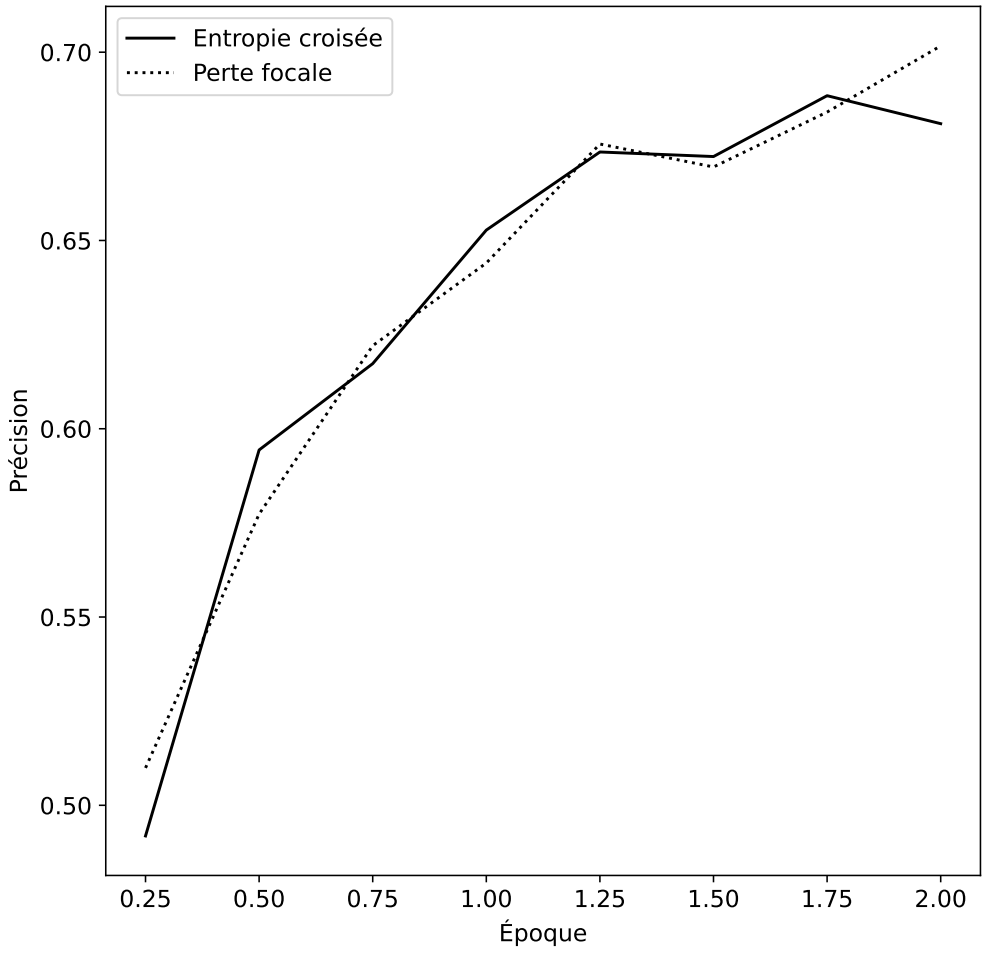

Figure 3.6 Effet de la fonction de perte focale sur la précision

des séquences d'entrée de 1500 nucléotides et 156 classes cible. On utilise des lots de 128 échantillons.

Le tableau 3.2 présente l'efficacité des performances pour différentes techniques de pré-traitement des données pour le transformateur et le modèle Mamba.

3.4.1 Mamba avec couche d'attention

La figure 3.8 montre la précision obtenue avec un modèle Mamba qui utilise une couche de moyennage ou d'attention pour convertir son espace latent en vecteur de classification.

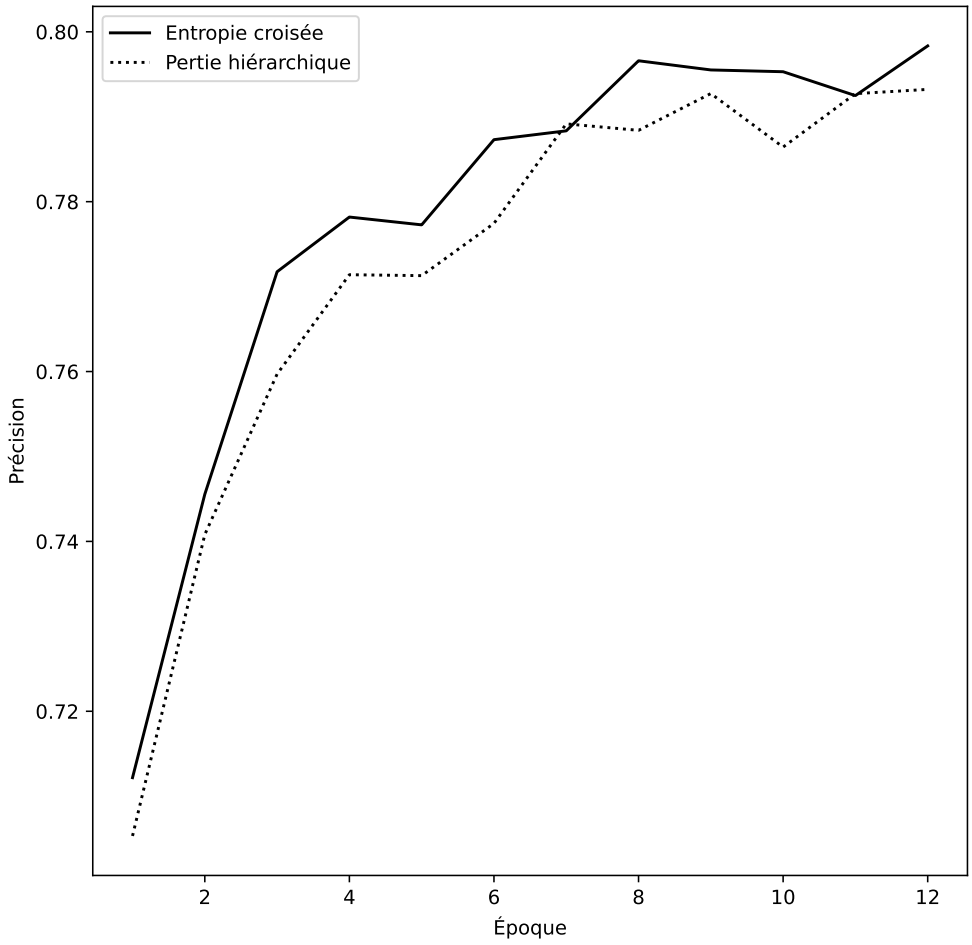

Figure 3.7 Effet de la fonction de perte hiérarchique sur la précision

3.5 Taille des modèles

Le tableau 3.3 compare les performances obtenues pour des modèles Mamba de tailles différentes. Les hyperparamètres de chaque modèle sont présentés au tableau 2.6. Les résultats sont rapportés pour le jeu de données 1 au niveau du genre après 12 époques d'entraînement. Tous les modèles rapportés dans le tableau utilisent l'encodage 3-mer et l'entropie croisée comme fonction de perte.

Table 3.1 Performance des modèles pour $N = 1500$

Outil	Nombre de paramètres	Durée d'un pas (s)
BERTax	$\approx 9\ 000\ 000$	≈ 2.00
MLP	$3\ 677\ 852$	0.010
CNN	$24\ 671\ 020$	0.024
AED	$26\ 182\ 286$	0.100
Transformer	$927\ 517$	0.188
Transformer-CNN	$1\ 047\ 453$	0.201
Mamba	$553\ 373$	0.176
Mamba-Dual	$586\ 268$	0.148
Mamba-CNN	$640\ 156$	0.189
Mamba-Dual-CNN	$724\ 636$	0.164

Table 3.2 Efficacité des performances pour différentes méthodes de pré-traitement des données

Modèle	Nombre de paramètres	Durée d'un pas (s)	Mesure-F %
Transformateur, 4-mer	$927\ 517$	$0,180$	$52,96$
Transformateur-CNN	$1\ 047\ 453$	$0,192$	$69,40$
Mamba, 4-mer	$553\ 373$	$0,138$	$72,45$
Mamba, 3-mer	$504\ 029$	$0,191$	$73,06$
Mamba, 2-mer	$491\ 693$	$0,301$	$72,49$
Mamba-CNN	$640\ 156$	$0,182$	$74,07$

Table 3.3 Efficacité des performances pour des modèles Mamba de tailles différentes

Modèle	Nombre de paramètres	Précision %
Petit	$553\ 373$	$75,59$
Moyen	$2\ 586\ 221$	$80,51$
Grand	$5\ 083\ 906$	$80,38$

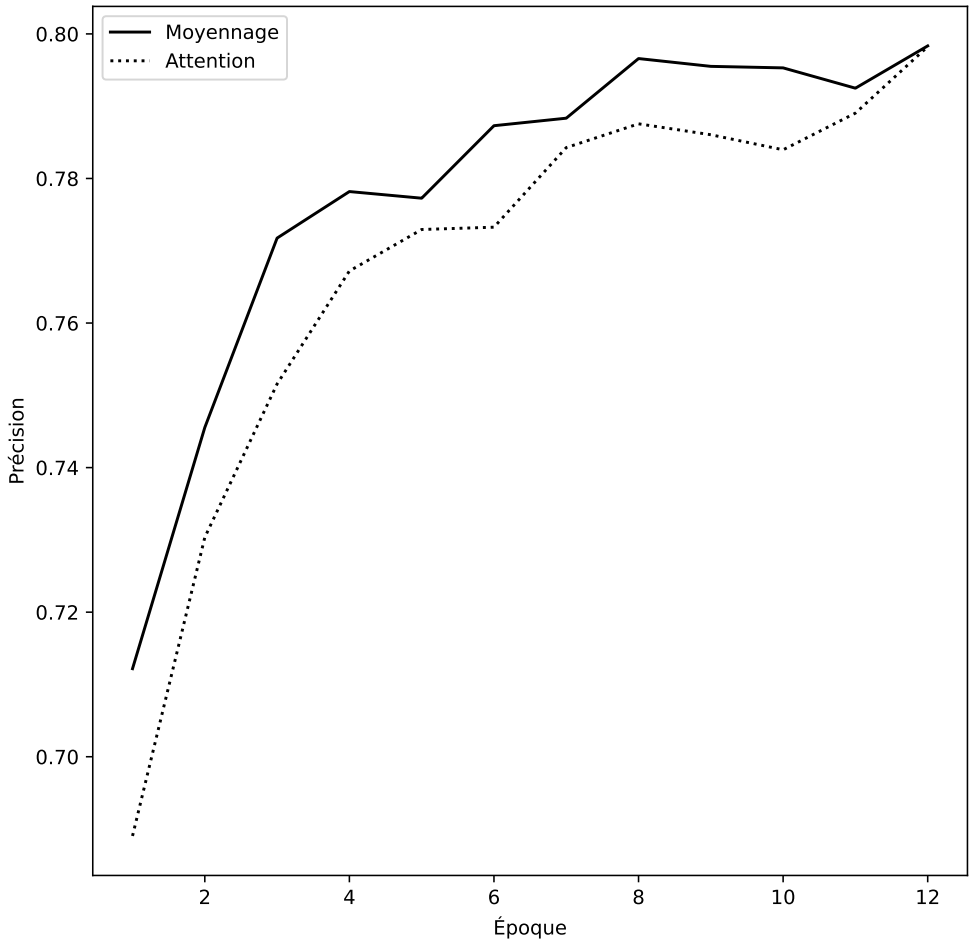

Figure 3.8 Effet du type de couche finale sur la précision

3.6 Comparaison des modèles

3.6.1 Jeu de données 1

Le tableau 3.4 compare les performances de différents modèles pour la classification du jeu de données 1 présenté à la section 2.4. Les modèles classifient des lectures de 1500 nucléotides dans 156 genres et 44 phylums. Les meilleurs résultats de chaque colonne sont marqués en **gras** et les seconds meilleurs, en *italique*.

Table 3.4 Comparaison des performances pour le jeu de données 1

	Macro précision %		
Outil	Domaine	Phylum	Genre
MMSeq2	$96,94$	$92,90$	$74,76$
MMSeq2 Tax.	$96,94$	$\mathit{93},\mathit{47}$	$75,09$
Kraken 2	$93,65$	$87,13$	$70,56$
DeepMicrobes	$\mathit{98},\mathit{13}$	$92,11$	$66,43$
BERTax	$\mathbf{98},\mathbf{62}$	$\mathbf{95.10}$	$66,92$
Forêt aléatoire	$60,89$	$44,98$	$21,07$
MLP	$55,52$	$27,92$	$15,56$
CNN	$69,62$	$72,10$	48.43
AED	$65,36$	$75,39$	50.10
Transformer	$82,07$	$77,82$	$57,48$
Transformer-CNN	$91,34$	$89,54$	$72,38$
Mamba	$96,39$	$91,34$	$75,59$
Mamba-Dual	$96,35$	$91,51$	$74,86$
Mamba-CNN	$97,27$	$92,19$	$\mathbf{76},\mathbf{98}$
Mamba-Dual-CNN	$97,53$	$92,84$	$\mathit{76},\mathit{53}$
Random classifier	$25,00$	$2,27$	$0,64$

3.6.2 Jeu de données 2

Le tableau 3.5 présente la précision obtenue avec le jeu de données 2 présenté à la section 2.4. Les modèles classifient des lectures de 3000 nucléotides dans 695 genres. Les meilleurs résultats de chaque colonne sont marqués en **gras** et les seconds meilleurs, en *italique*.

La figure 3.9 montre les performances des modèles à différents niveaux taxonomiques.

Les résultats détaillés pour ce jeu de données sont disponible dans l'annexe A.

Table 3.5 Précision macro pour le jeu de données 2.

	Précision macro %					
Tool	Domaine	Phylum	Classe	Ordre	Famille	Genre
MMSeqs 2	$98,05$	$92,62$	$91,39$	$83,96$	$77,91$	$56,02$
Kraken 2	$88,97$	$71,00$	$69,91$	$64,41$	$60,67$	$48,49$
Transformer	$97,61$	$81,21$	$79,54$	$69,13$	$63,92$	$51,55$
Transformer-CNN	$98,22$	$88,47$	$87,63$	$80,40$	$75,77$	$63,77$
Mamba	$98,92$	$93,88$	$92,67$	$87,61$	$83,80$	$73,28$
Mamba-Dual	$99,00$	$94,67$	$\mathbf{93,55}$	$\mathbf{88,97}$	$84,71$	$73,36$
Mamba-CNN	$\mathit{99,08}$	$\mathbf{94,87}$	$\mathit{93,51}$	$\mathit{88,91}$	$\mathit{84,76}$	$\mathit{73,48}$
Mamba-Dual-CNN	$\mathbf{99,20}$	$\mathit{94,72}$	$93,34$	$88,64$	$\mathbf{84,97}$	$\mathbf{73,68}$
Modèle aléatoire	$50,00$	$3,85$	$2,50$	$0,93$	$0,46$	$0,14$

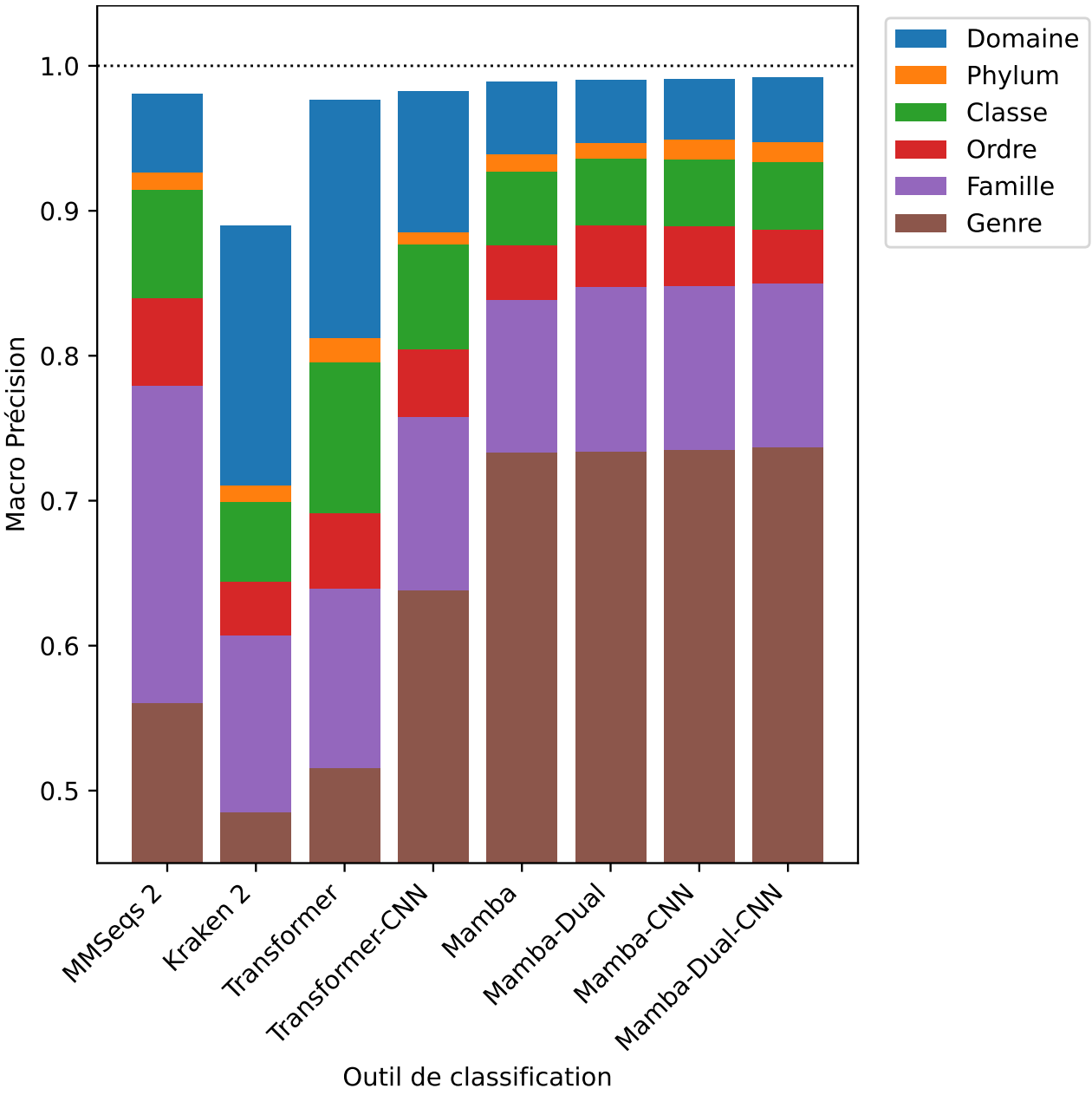

Figure 3.9 Précision macro à 6 niveaux taxonomiques pour le jeu de données 2

CHAPITRE 4
DISCUSSION

4.1 Effet du pré-entraînement

Les figures 3.1 et 3.2 montrent respectivement l'entropie observée pour un pré-entraînement causal ou MLM. On note que le pré-entraînement est instable avec l'entraînement causal parce que l'entropie augmente après une période de diminution. On observe toutefois une stabilisation de l'entropie avec l'entraînement MLM. Une explication pour la différence de performances entre les deux types d'entraînement est la difficulté de la tâche. L'entraînement causal est plus complexe parce que le modèle doit déterminer une séquence ininterrompue de nucléotides à la fin des échantillons. L'entraînement MLM est plus simple parce que le modèle peut s'appuyer sur les données avant et après les jetons masquées pour identifier leur valeur originale. Des modèles plus grands et une plus grande quantité de données d'entraînement pourraient permettre aux modèles d'atteindre une entropie et une perte plus basse durant le pré-entraînement, mais à l'échelle des modèles testés, l'entraînement MLM est plus efficace.

La figure 3.3 montre la perte observée pour deux modèles, le transformateur et Mamba, pour deux régimes d'entraînements, pré-entraînement suivi d'ajustement et entraînement direct, pour un total de quatre courbes. On note les observations suivantes :

- Le transformateur bénéficie du pré-entraînement, ce qui concorde avec les recherches en apprentissage automatique selon lesquelles le pré-entraînement est nécessaire pour atteindre de bonnes performances avec cette architecture (Clark *et al.*, 2019). Toutefois, on note que pour un nombre élevé d'époques, les performances des deux régimes d'entraînement finissent par se ressembler. Néanmoins, pour des transformateurs comportant davantage de paramètres, la phase de pré-entraînement ne sert pas seulement à diminuer la durée de l'entraînement ; c'est plutôt un processus nécessaire pour permettre au modèle de converger vers une solution. On conclut que le pré-entraînement est requis pour tous les transformateurs de classification taxonomique.
- Le modèle Mamba ne bénéficie pas du pré-entraînement parce que les courbes correspondant aux modèles pré-entraîné et non pré-entraîné ne diffèrent pas significativement. On conclut que Mamba n'a pas besoin de pré-entraînement pour excéder les performances d'un transformateur de taille similaire pour cette tâche. Ce résultat peut s'expliquer par le mécanisme de compression à la base

de Mamba. En entraînant le modèle à modéliser les séquences au lieu de les classifier, le modèle apprendrait une manière de sélectionner les caractéristiques des données d'entrée adaptée à la modélisation mais pas à la classification. Le pré-entraînement améliore les performances de modèles hybrides qui combinent les caractéristiques des transformateurs et des SSM (Ren *et al.*, 2025), mais un modèle Mamba seul n'obtient pas de gain de performance.

La non-nécessité du pré-entraînement pour les modèles Mamba en constitue un avantage parce qu'elle réduit le temps d'entraînement total sans causer de pertes de performance.

4.2 Effets de l'encodage

La figure 3.4 montre la perte observée au cours de l'entraînement pour différentes techniques de pré-traitement de données : la jetonisation en K-mers (transformation de sous-séquences en identifiants) et le traitement par couches CNN.

Pour la jetonisation, on note qu'une conversion de 3-mers est plus efficace que des 2-mers ou 4-mers. On peut expliquer ce phénomène par la nature des modèles SSM, qui compriment le contexte dans un espace latent. En utilisant une jetonisation basée sur les 2-mers, la séquence est la plus longue et le modèle doit emmagasiner plus d'informations. En utilisant une jetonisation basée sur lse 4-mers, la séquence est plus courte mais davantage d'artefacts sont introduits. La jetonisation de 3-mers atteint un équilibre entre ces deux facteurs et parvient aux meilleures performances.

La couche CNN entraîne les meilleurs résultats et la plus faible durée de convergence entre toutes les méthodes de pré-traitement. On conclut qu'il s'agit donc d'une manière efficace d'améliorer les performances d'un modèle Mamba pour la classification taxonomique.

La figure 3.5 compare la précision obtenue avec la jetonisation par 3-mers et l'encodage BPE, qui produit des jetons de longueurs variables. Les recherches démontrent que l'encodage BPE peut améliorer (Zhou *et al.*, 2024) ou diminuer (Wu *et al.*, 2026) les performances des modèles selon la tâche à effectuer. Pour la classification taxonomique, la nature hiérarchique de l'encodage BPE peut expliquer la perte de précision (Wu *et al.*, 2026).

4.3 Effet de la fonction de perte

La figure 3.6 montre la précision observée en entraînant un modèle Mamba avec deux fonctions de perte différentes, l'entropie croisée (standard dans les problèmes de classification multi-classes) et la perte focale (recommandée pour les jeux de données fortement déséquilibrés). On rapporte la précision sur l'ensemble de validation parce que la perte est calculée à partir d'équations différentes et ne constitue donc pas un point de comparaison fiable. On n'observe pas de différence significative entre les performances des modèles entraîné avec l'une ou l'autre des fonctions de perte. Dans le jeu de données 1, le nombre d'échantillon entre les classes abondantes et peu abondantes diffèrent par un facteur d'environ 1 pour 20. On en conclut que ce jeu de donnée n'est pas suffisamment déséquilibré pour justifier l'utilisation d'une autre fonction de perte que l'entropie croisée.

La figure 3.7 montre la précision observée en entraînant un modèle Mamba avec une fonction de perte d'entropie croisée ou de perte hiérarchique. Bien que la perte hiérarchique s'appuie sur les données taxonomiques et motive le modèle à tenir compte des distances entre les espèces lors de la classification, on n'observe pas de gain de performance en comparaison avec l'entropie croisée, qui considère toutes les classes comme si elles appartenaient toutes au même niveau taxonomique. Comme relevé par l'équipe de Wu (Wu C, 2019), la perte hiérarchique est surtout utile pour les classes représentées par un petit nombre d'échantillon, ce qui n'est pas le cas dans le jeu de données 1. On en conclut que l'utilisation d'une fonction de perte hiérarchique qui exploite les liens taxonomiques entre les classes n'entraîne pas de gain de performance notable.

4.3.1 Mamba avec couche d'attention

La figure 3.8 montre que l'utilisation d'une couche d'attention entraîne des performances moins élevées qu'une couche de moyennage. Contrairement aux résultats rapportés par (Glorioso *et al.*, 2024), la combinaison hybride de modèles SSM et d'attention n'améliore pas la classification avec les modèles à l'essai.

4.4 Taille des modèles

Le tableau 3.3 compare les performances de modèles Mamba qui comporte un nombre variable de paramètres avec le jeu de données 1. On observe une plus grande précision pour le modèle comportant $5\ M$ de paramètres tandis que les modèles plus petit et plus grand ont des performances moindre. Un nombre trop

réduit de paramètre empêche le modèle de représenter efficacement les données et un trop grand nombre de paramètres est davantage susceptible de provoquer un surajustement.

4.5 Efficacité des performances

Le tableau 3.1 montre le nombre de paramètres et la durée d'un pas pour les modèles testés. Le nombre de paramètres est corrélés au nombre de FLOPS pour une architecture donnée (Kaplan *et al.*, 2020), mais on note, comme attendu, des différences marquées entre les types d'architecture. Les modèles récents (transformateurs et Mamba) comprennent relativement peu de paramètres mais requièrent plus de FLOPS et un temps de calcul plus élevé pour calculer l'auto-attention et l'espace latent. Les modèles plus anciens (MLP, CNN et DAE) comprennent plus de paramètres mais demandent moins de ressources de calcul. Leurs performances sont néanmoins inférieures.

Le tableau 3.2 montre les performances associées aux différentes techniques de pré-traitement des données pour le modèle Mamba. Comme à la section 4.3, on note que le pré-traitement par CNN entraîne les meilleures performances, suivi de la jetonisation par 3-mers, 2-mers et 4-mers, dans cet ordre. Le modèle qui utilise le pré-traitement par CNN s'entraîne plus rapidement que celui qui utilise la jetonisation par 3-mers, ce qui constitue un autre avantage par rapport à cette alternative. Au niveau du transformateur, on observe que le pré-traitement par CNN entraîne une amélioration des performances encore plus marquée qu'avec le modèle Mamba. Le transformateur avait été développé à l'origine pour le traitement des langues naturelles (Vaswani *et al.*, 2017) et utilise donc la jetonisation en entrée et en sortie, mais dans le contexte de la classification taxonomique, le pré-traitement par CNN en constitue une alternative viable.

4.6 Comparaison des modèles

Le tableau 3.4 compare les performances de tous les modèles testés sur le jeu de données 1. On effectue les observations suivantes :

- Les modèles d'apprentissage automatique basés sur des architectures simples (forêt aléatoire, MLP, CNN, AED) livrent les pires performances, ce qui est attendu parce qu'ils ne sont pas conçus spécifiquement pour analyser des données séquentielles. On observe que les couches convolutionnelles augmentent grandement les performances parce qu'elles extraient des motifs locaux dans les séquences.

- Les modèles DeepMicrobes et BERTax, qui comprennent plusieurs millions de paramètres, offrent de bonnes performances pour les niveaux taxonomiques élevés, mais leurs performances sur un niveaux taxonomique faible sont plus faibles. Ces modèles sont basés sur l'auto-attention et leur entraînement favorise l'utilisation de motifs globaux dans les séquences, ce qui explique que l'identification de taxons précis soit plus ardue.
- Les modèles classiques (MMSeq2 et Kraken 2) offrent de bonnes performances pour un niveau taxonomique faible mais de moins bonnes performances pour les niveaux taxonomiques élevés. C'est un résultat attendu parce que ces méthodes sont basées sur une comparaison avec des bases de données de référence qui cherchent les meilleures correspondances entre une séquence de test et les séquences connues, donc leur compréhension de motifs globaux est moins poussée.
- Le modèle Mamba et ses variantes livrent les meilleures performances pour les niveaux taxonomiques faibles et de bonnes performances pour les niveaux élevés. En moyenne, ils livrent les meilleures performances de tous les modèles testés pour le jeu de données.

Le tableau 3.5 compare les performances de tous les modèles testés sur le jeu de données 2. La figure 3.9 présente les mêmes données sous la forme d'un graphe à barres verticales. On effectue les observations suivantes :

- Le modèle Mamba et ses variantes livrent les meilleures performances pour tous les niveaux taxonomique.
- Les améliorations de Mamba offrent de meilleures gains de performances qu'avec le jeu de données 1. On explique cela par la longueur des séquences, qui comprennent deux fois plus de nucléotides que le jeu de données 1. La variante Mamba-Dual, en particulier, montre des performances plus élevées dans ce cas de figure parce que le traitement des séquences en flux parallèles sur les longues séquences permet d'utiliser différentes échelles temporelles et diversifie les motifs analysés par le modèle.

4.7 Implication pour la métagénomique

Bien que les modèles d'apprentissage automatique métagénomiques demeurent principalement limités au domaine de la recherche (Roy *et al.*, 2024), les améliorations de ces modèles ouvrent la porte à des applications cliniques. Les modèles d'AA pourraient livrer de meilleures performances que les modèles classiques

et identifier avec plus de précision des pathogènes ou mieux caractériser des métagénomes. Plusieurs facteurs détermineront le degré d'applicabilité de ces modèles :

- Le prix d'entraînement et d'inférence des modèles déterminera leur accessibilité aux professionnels de la santé. Les modèles SSM, et en particulier le modèle Mamba, constituent ainsi une avenue intéressant dans ce domaine parce que leur temps d'entraînement linéaire permet d'obtenir des modèles moins dispendieux que des architectures basées sur des transformateurs. Certains petits modèles spécialisés pourraient aussi être directement entraînés ou ajustés sur du matériel informatique disponible au public, ce qui permettrait de les adapter à des besoins spécifiques, comme à l'identification de pathogènes d'un souche particulière.
- La fiabilité des résultats est nécessaire pour les applications cliniques. Les modèles basés sur les transformateurs et les SSM disposent d'une faible interprétabilité et davantage de recherche est requise pour comprendre leur fonctionnement et valider leur prédiction. Les résultats produits par les modèles d'AA peuvent néanmoins être validés avec des bases de données fiables, comme RefSeq, et validées avec des techniques de classification classiques.

4.8 Améliorations possibles

Les modèles présentés offrent de meilleures performances que les modèles récents pour la classification taxonomique, mais il est possible de les améliorer davantage avec les plusieurs modifications.

Augmenter la taille des modèles en haussant les dimensions de l'espace latent et le nombre de couches leur permet de modéliser des données plus complexes et d'effectuer des prédictions plus précises. Toutefois, comme le montre le tableau 3.3, les performances plafonnent après un certain nombre de paramètres (environ 5 millions pour le jeu de données 1), on n'observe plus de gian de performance notable. La taille des modèles doit donc être ajustée en fonction du type de problème à résoudre et de la quantité de données pour maximiser la précision du modèle et minimiser sa durée d'entraînement.

Spécialiser les modèles à différentes sous-tâches, avec une approche de mélange d'experts **??** par exemple, pourrait permettre aux modèles de classifier plus efficacement les séquences et de requérir moins de temps d'entraînement. Cette approche a été utilisée avec succès pour entraîner de grands modèles de langues pour un coût relativement faible (DeepSeek-AI *et al.*, 2025) et pourrait être appliquée à des modèles d'analyse

de données métagénomiques.

CONCLUSION

Ce mémoire présente des modèles d'apprentissage automatique développés pour la classification taxonomique de données métagénomiques. Les modèles sont basés sur différentes architectures (MLP, CNN, AED, transformateur et SSM) et sont entraînés à partir de données synthétiques générées avec la base de données RefSeq. Les performances des modèles sont comparés avec deux outils de classification classiques, MMSeqs 2 et Kraken 2, ainsi que deux modèles d'apprentissage automatique tiers, DeepMicrobes et BERTax.

Le modèle SSM Mamba de base a une précision macro de $88,07\%$ en considérant la moyenne de deux évaluation sur des jeux de données distincts . On teste deux améliorations architecturales. (1) Pré-traiter les données d'entrée avec des couches CNN au lieu d'utiliser la jetonisation entraîne une précision moyenne supérieure de $88,96\%$. (2) Effectuer le traitement de données avec des flux parallèles dans un modèle Mamba entraîne une précision moyenne de $88,31\%$. Un transformateur de taille comparable classifie les mêmes séquences avec une précision moyenne de $73,15\%$. D'autres modifications ont été testées, comme l'encodage des séquences par BPE, des fonctions de perte différentes de l'entropie croisée ou l'ajout d'un mécanisme d'attention, mais elles n'améliorent pas les performances des modèles.

Ces résultats démontrent que les modèles à architecture SSM constituent une alternative viable aux transformateurs pour la classification taxonomique. Ils parviennent à une plus grande précision et peuvent être entraînés en un temps linéaire $O(n)$ au lieu de quadratique, $O(n^2)$, pour les transformateurs.

ANNEXE A

RÉSULTATS DÉTAILLÉS

Cette section détaille les résultats présentés à la section 3.6.2.

Les tableaux A.1, A.2 et A.3 présentent respectivement la précision, rappel et mesure-F obtenue avec le jeu de données 2 présenté à la section 2.4. Les modèles classifient des lectures de 3000 nucléotides dans 695 genres. Les meilleurs résultats de chaque colonne sont marqués en **gras** et les seconds meilleurs, en

Table A.1 Précision macro pour le jeu de données 2.

	Précision macro %					
Tool	Domaine	Phylum	Classe	Ordre	Famille	Genre
MMSeqs 2	$98,05$	$92,62$	$91,39$	$83,96$	$77,91$	$56,02$
Kraken 2	$88,97$	$71,00$	$69,91$	$64,41$	$60,67$	$48,49$
Transformer	$97,61$	$81,21$	$79,54$	$69,13$	$63,92$	$51,55$
Transformer-CNN	$98,22$	$88,47$	$87,63$	$80,40$	$75,77$	$63,77$
Mamba	$98,92$	$93,88$	$92,67$	$87,61$	$83,80$	$73,28$
Mamba-Dual	$99,00$	$94,67$	$\mathbf{93,55}$	$\mathbf{88,97}$	$84,71$	$73,36$
Mamba-CNN	$\mathit{99,08}$	$\mathbf{94,87}$	$\mathit{93,51}$	$\mathit{88,91}$	$\mathit{84,76}$	$\mathit{73,48}$
Mamba-Dual-CNN	$\mathbf{99,20}$	$\mathit{94,72}$	$93,34$	$88,64$	$\mathbf{84,97}$	$\mathbf{73,68}$
Modèle aléatoire	$50,00$	$3,85$	$2,50$	$0,93$	$0,46$	$0,14$

Les figures A.1, A.2 et A.3 montrent les performances des modèles à différents niveaux taxonomiques.

Les figures A.4, A.5 et A.6 montrent les performances des modèles à différents niveaux taxonomiques. Les barres horizontales indiquent les médianes.

Table A.2 Rappel macro pour le jeu de données 2.

	Rappel macro %					
Tool	Domaine	Phylum	Classe	Ordre	Famille	Genre
MMSeqs 2	97,95	87,16	86,37	80,35	74,92	55,19
Kraken 2	89,97	64,76	64,10	59,63	57,65	48,99
Transformer	95,65	77,85	77,15	67,72	61,58	47,72
Transformer-CNN	97,51	88,47	87,48	80,63	75,15	62,09
Mamba	*98*, *97*	**95**, **03**	**94**, **05**	*88*, *99*	84,56	72,60
Mamba-Dual	98,86	*94*, *98*	93,79	88,65	84,57	*73*, *07*
Mamba-CNN	**99**, **01**	94,81	*93*, *94*	88,63	**84**, **65**	73,03
Mamba-Dual-CNN	98,88	94,82	93,93	**89**, **01**	*84*, *61*	**73**, **14**
Modèle aléatoire	50,00	3,85	2,50	0,93	0,46	0,14

Table A.3 Mesure-F macro pour le jeu de données 2.

	Mesure-F macro %					
Tool	Domaine	Phylum	Classe	Ordre	Famille	Genre
MMSeqs 2	97,99	89,65	88,61	81,92	76,14	55,34
Kraken 2	89,39	67,00	66,10	61,16	58,41	48,02
Transformer	96,61	78,70	77,02	66,97	61,20	47,47
Transformer-CNN	97,86	87,99	87,14	79,95	74,73	61,50
Mamba	*98*, *95*	94,38	93,26	88.11	83.82	72.18
Mamba-Dual	98,93	*94*, *76*	*93*, *57*	**88**, **65**	84,39	72,62
Mamba-CNN	**99**, **04**	**94**, **77**	**93**, **65**	*88*, *61*	*84*, *43*	*72*, *64*
Mamba-Dual-CNN	**99**, **04**	94,67	93,49	*88*, *61*	**84**, **48**	**72**, **75**
Modèle aléatoire	50,00	3,85	2,50	0,93	0,46	0,14

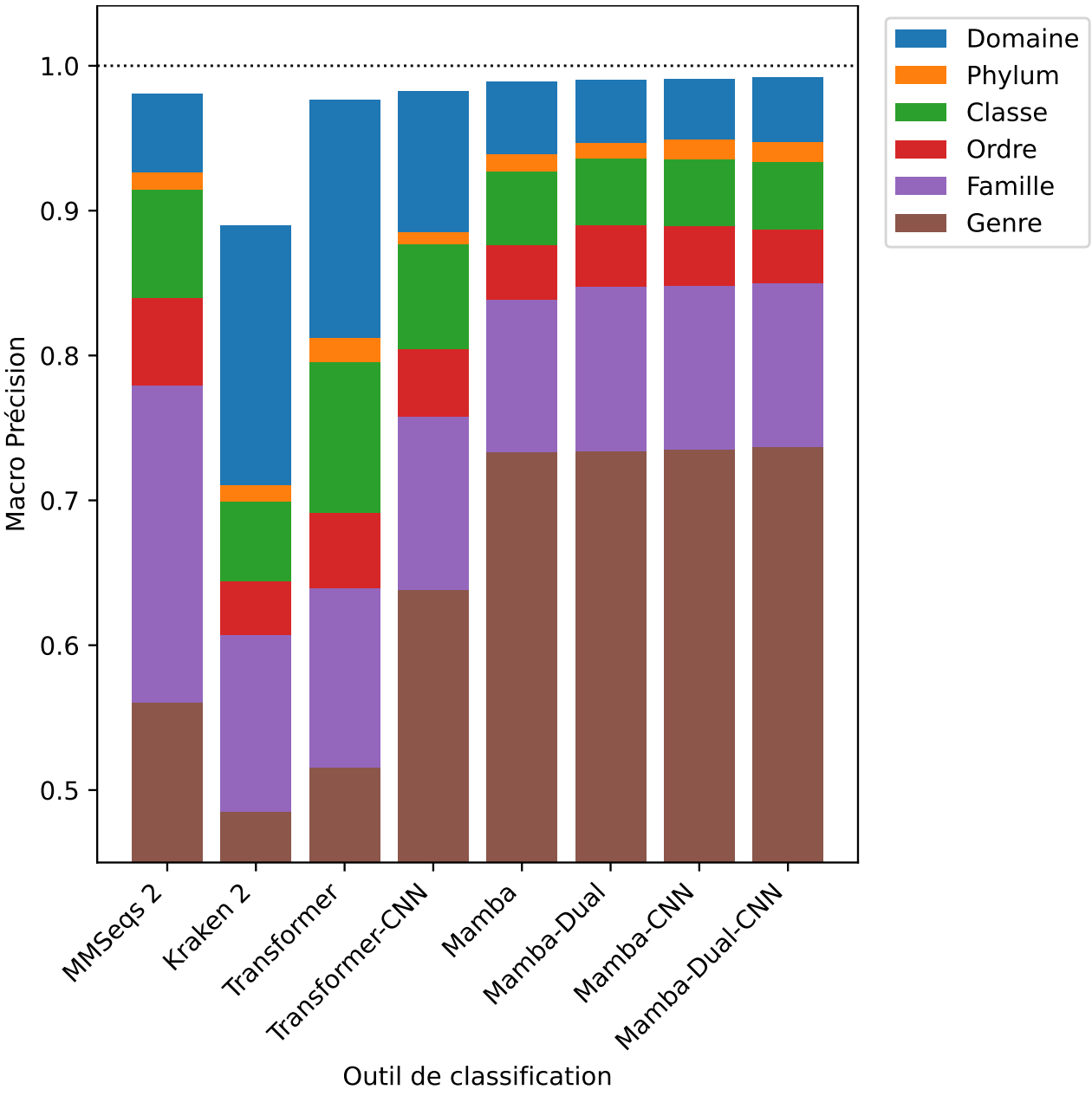

Figure A.1 Précision macro à 6 niveaux taxonomiques pour le jeu de données 2

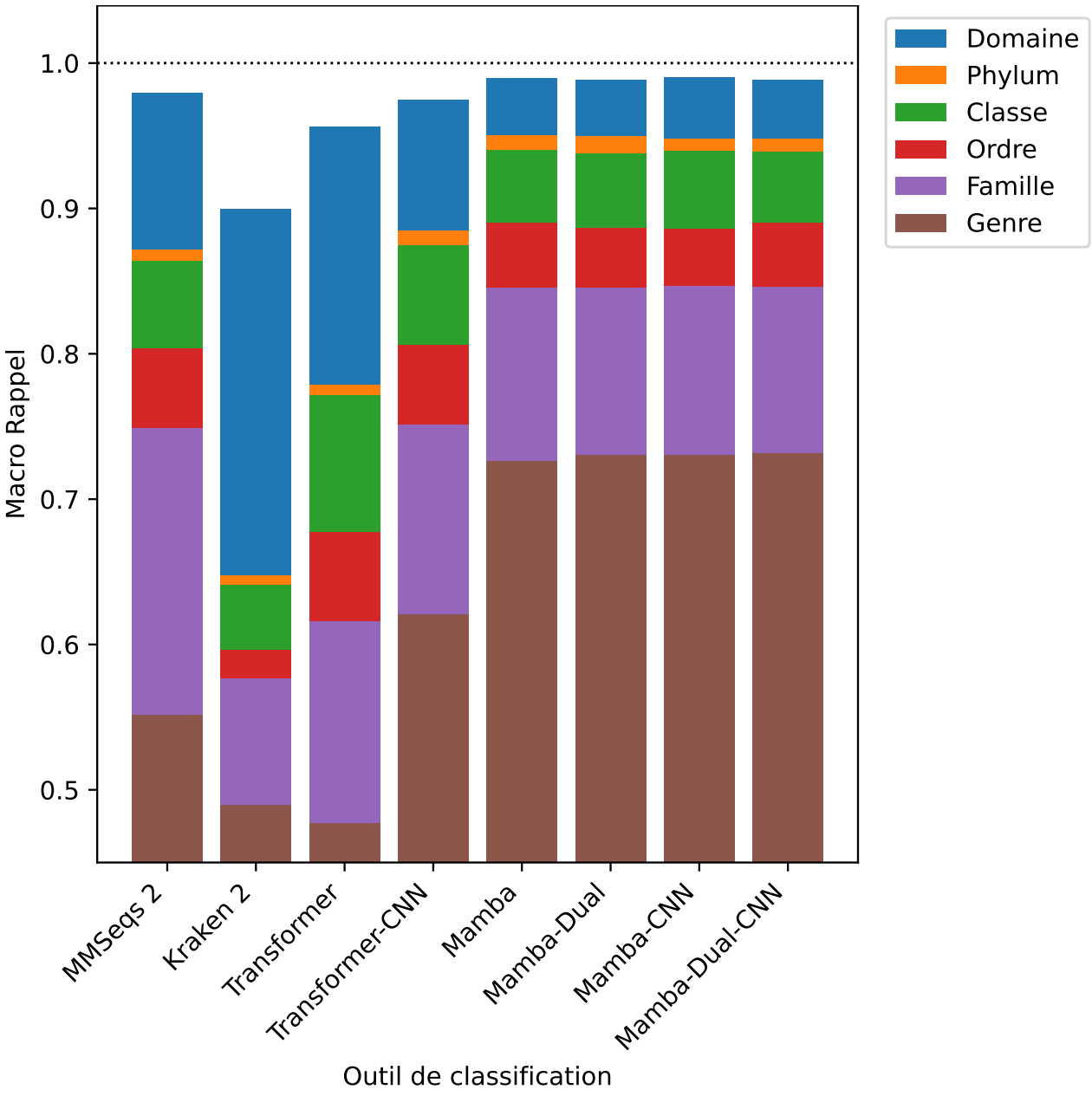

Figure A.2 Rappel macro à 6 niveaux taxonomiques pour le jeu de données 2

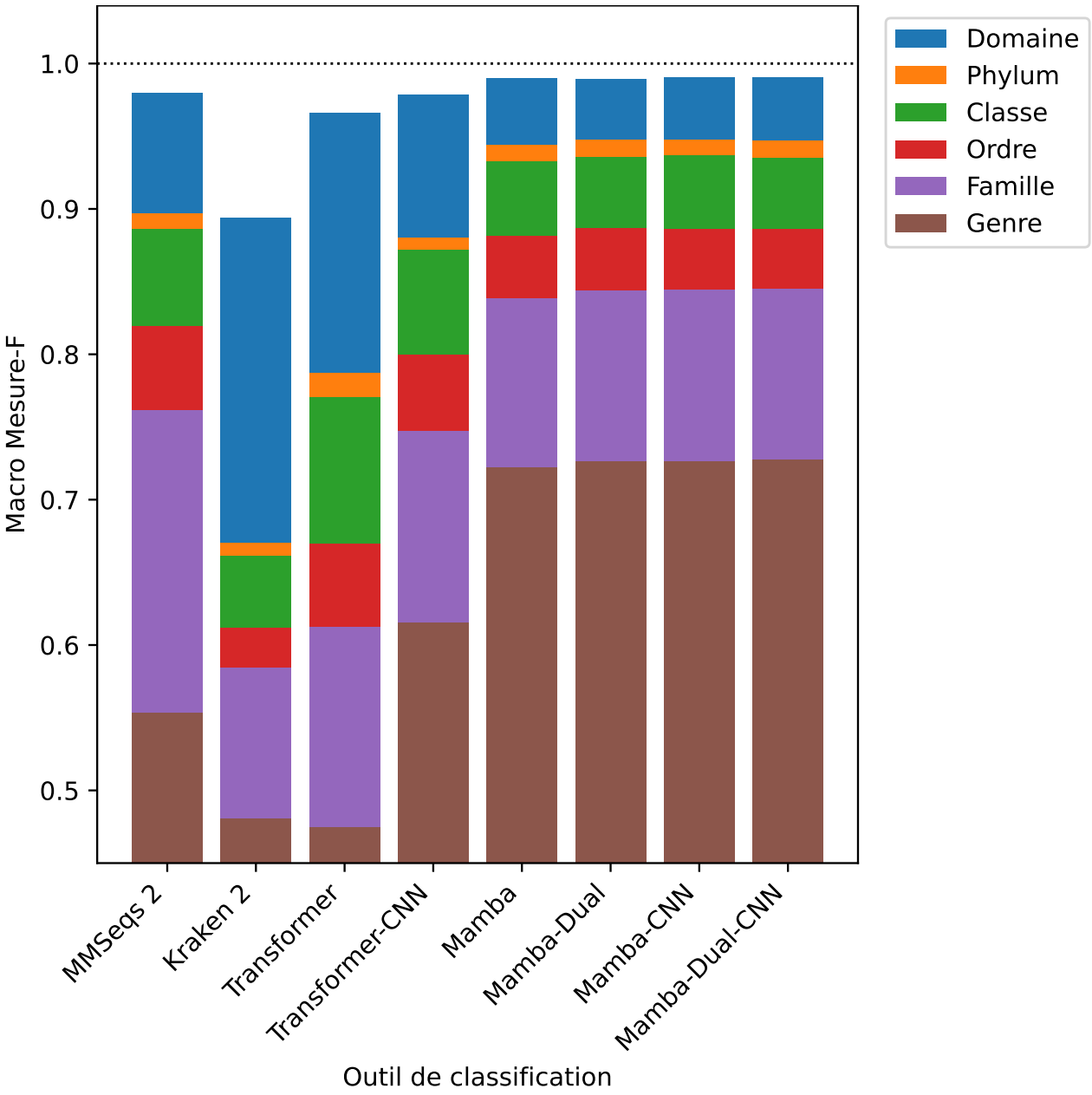

Figure A.3 Mesure-F macro à 6 niveaux taxonomiques pour le jeu de données 2

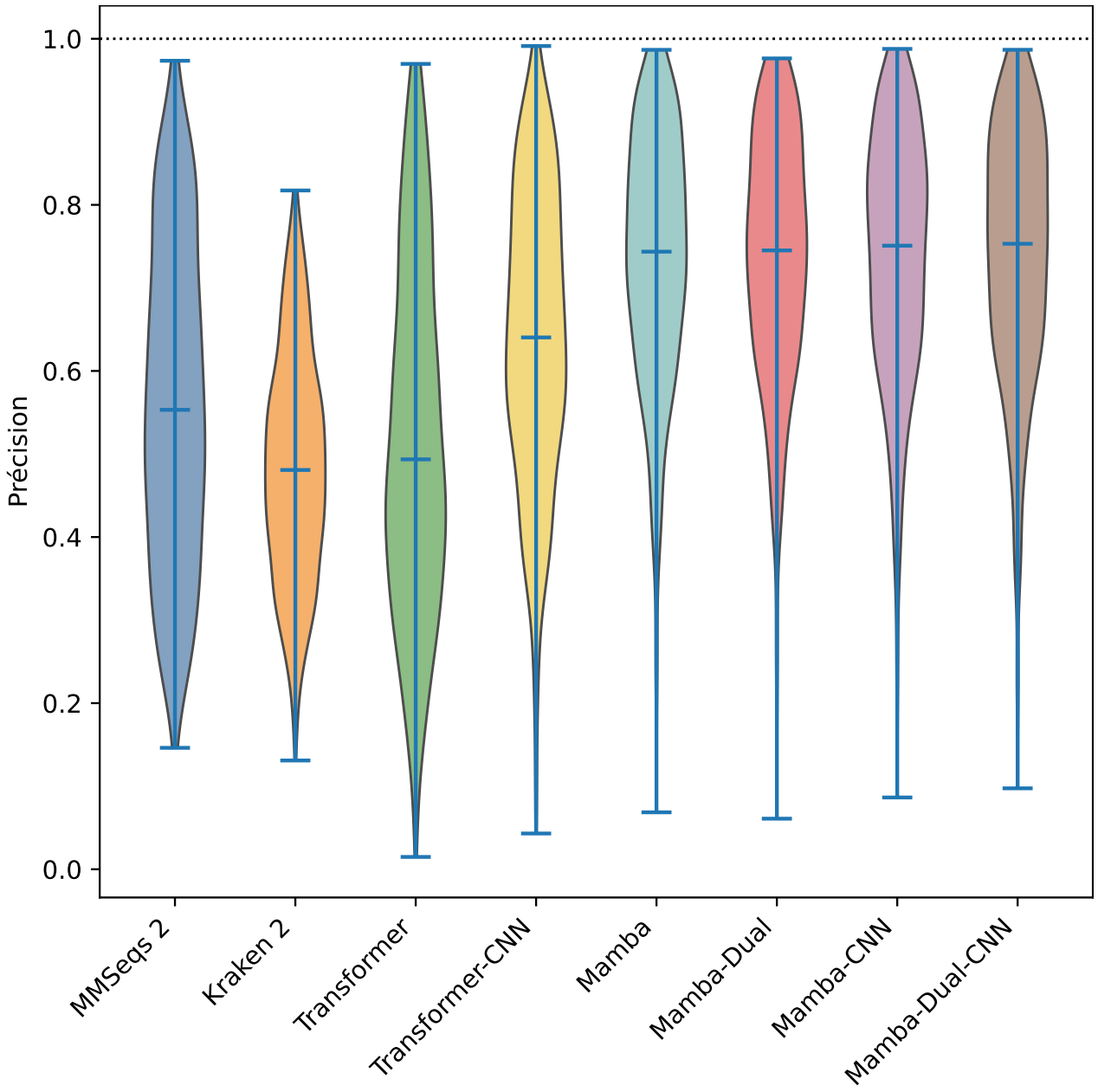

Figure A.4 Diagramme violon de la précision macro pour le jeux de données 2

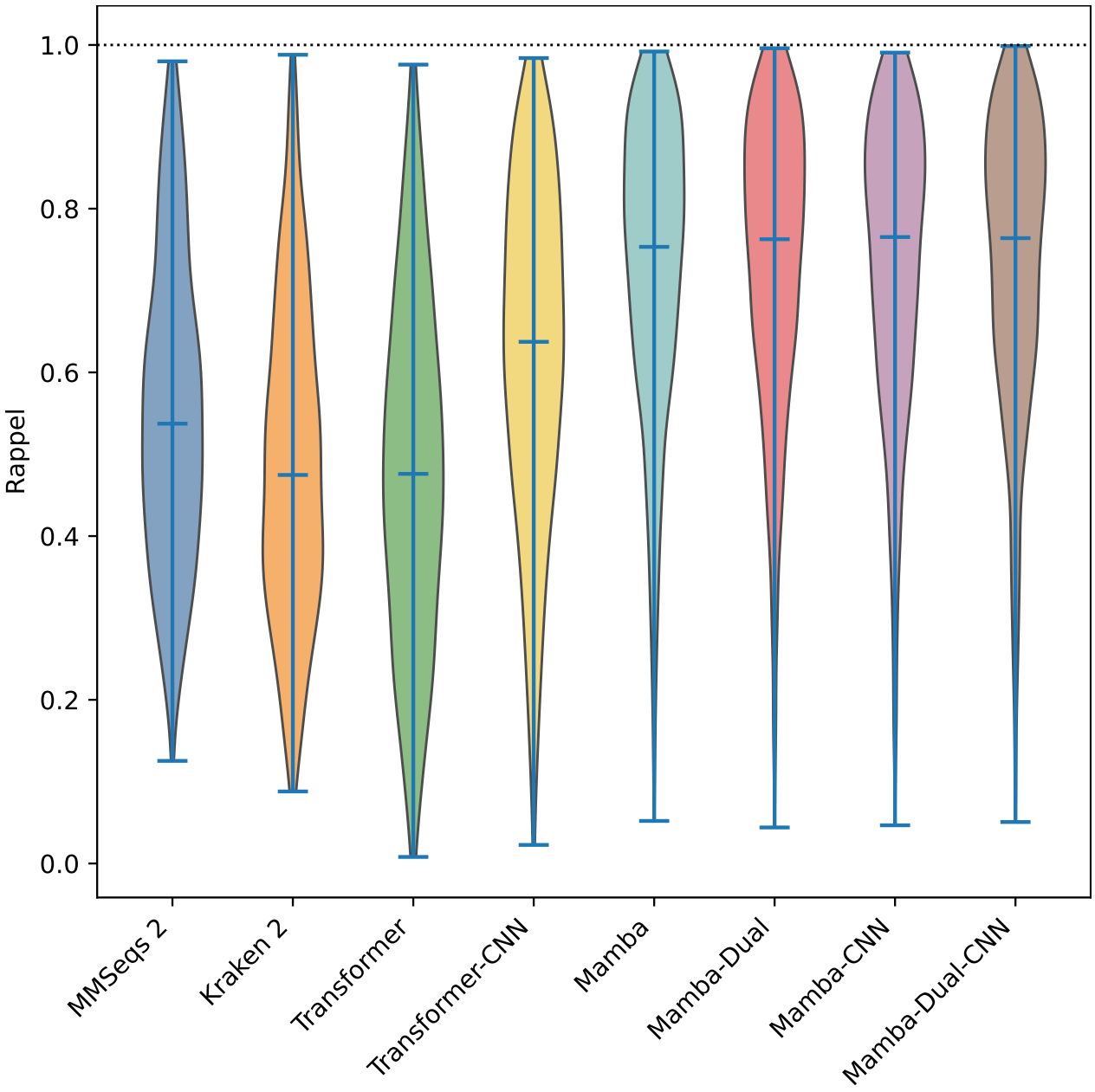

Figure A.5 Diagramme violon du rappel macro pour le jeux de données 2

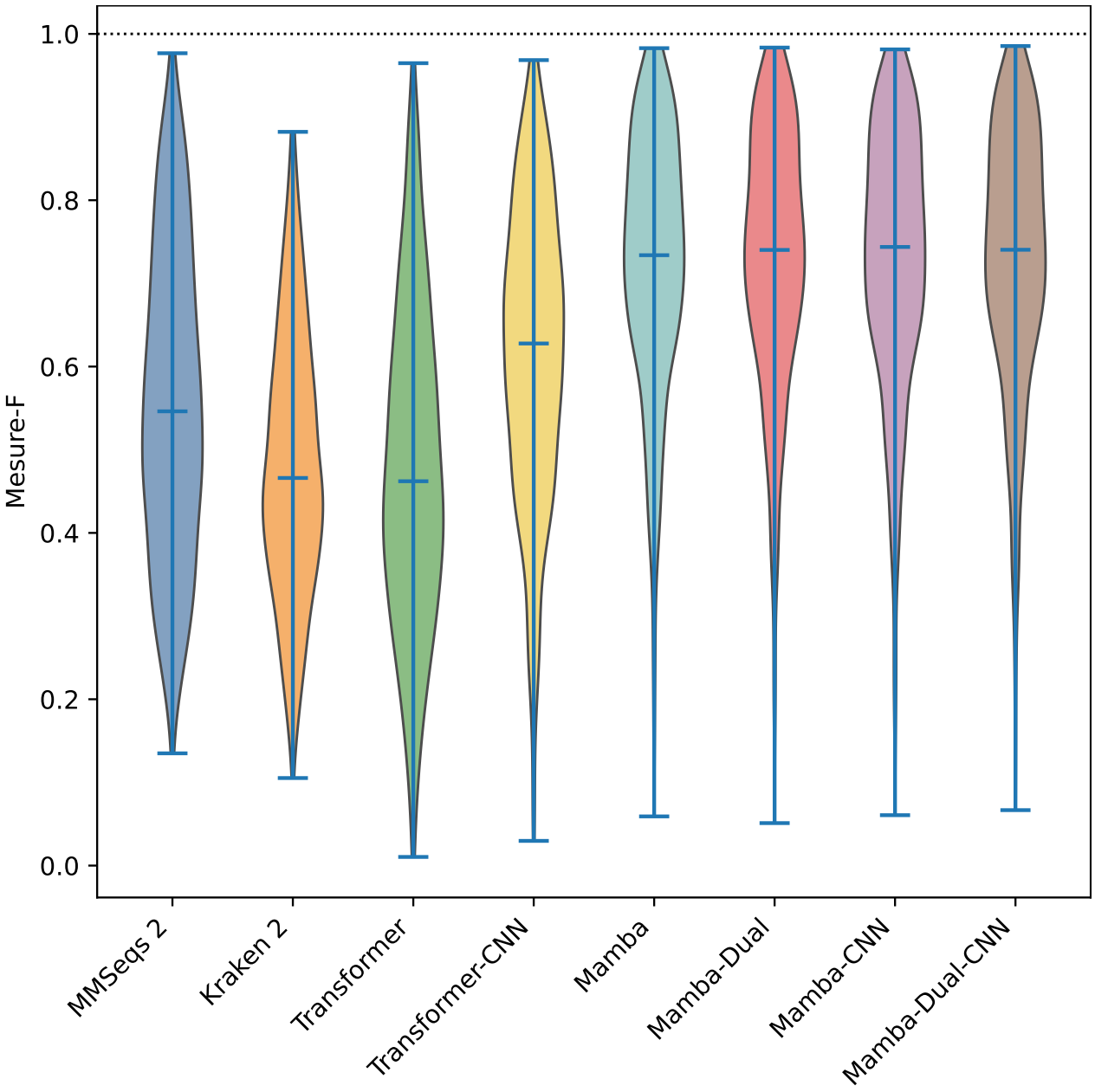

Figure A.6 Diagramme violon de la mesure-F macro pour le jeux de données 2

BIBLIOGRAPHIE

Alcock, B. P., Huynh, W., Chalil, R., Smith, K. W., Raphenya, A. R., Wlodarski, M. A., Edalatmand, A., Petkau, A., Syed, S. A., Tsang, K. K. *et al.* (2023). Card 2023 : expanded curation, support for machine learning, and resistome prediction at the comprehensive antibiotic resistance database. *Nucleic Acids Research*, *51*(D1), D690–D699. http://dx.doi.org/10.1093/nar/gkac920

Altschul, S. F. *et al.* (1997). Gapped BLAST and PSI-BLAST : a new generation of protein database search programs. *Nucleic Acids Research*, *25*(17), 3389–3402. http://dx.doi.org/10.1093/nar/25.17.3389

Arango-Argoty, G., Garner, E., Pruden, A., Heath, L. S., Vikesland, P. et Zhang, L. (2018). Deeparg : a deep learning approach for predicting antibiotic resistance genes from metagenomic data. *Microbiome*, *6*(1), 1–15. http://dx.doi.org/10.1186/s40168-018-0401-z. Récupéré de https://doi.org/10.1186/s40168-018-0401-z

Bank, D., Koenigstein, N. et Giryes, R. (2023). *Autoencoders*, Dans L. Rokach, O. Maimon, et E. Shmueli (dir.). *Machine Learning for Data Science Handbook : Data Mining and Knowledge Discovery Handbook*, (p. 353–374). Springer International Publishing : Cham

Beghini, F. *et al.* (2021). Integrating taxonomic, functional, and strain-level profiling of diverse microbial communities with biobakery 3. *eLife*, *10*, e65088. http://dx.doi.org/10.7554/eLife.65088. Récupéré de https://doi.org/10.7554/eLife.65088

Benítez-Páez, A. *et al.* (2022). Species- and strain-level assessment using rrn long-amplicons suggests donor's influence on gut microbial transference via fecal transplants in metabolic syndrome subjects. *Gut Microbes*, *14*(1), 2078621. PMID : 35604764 ; PMCID : PMC9132484, http://dx.doi.org/10.1080/19490976.2022.2078621

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J. et Sayers, E. W. (2011). Genbank. *Nucleic Acids Research*, *39*(Database issue), D32–D37. Epub 2010 Nov 10, http://dx.doi.org/10.1093/nar/gkq1079

Berg, G. *et al.* (2020). Microbiome definition re-visited : old concepts and new challenges. *Microbiome*, *8*(1), 103. http://dx.doi.org/10.1186/s40168-020-00875-0

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. http://dx.doi.org/10.1023/A:1010933404324. Récupéré de https://doi.org/10.1023/A:1010933404324

Buchfink, B. *et al.* (2015). Fast and sensitive protein alignment using DIAMOND. *Nature Methods*, *12*(1), 59–60. http://dx.doi.org/10.1038/nmeth.3176

Busia, A., Dahl, G. E., Fannjiang, C., Alexander, D. H., Dorfman, E., Poplin, R., McLean, C. Y., Chang, P.-C. et DePristo, M. (2019). A deep learning approach to pattern recognition for short dna sequences. *bioRxiv*. http://dx.doi.org/10.1101/353474. Récupéré de https://www.biorxiv.org/content/early/2019/08/10/353474

Chapelle, O., Schölkopf, B. et Zien, A. (2006). *Semi-supervised learning*. MIT press. http://dx.doi.org/https://doi.org/10.7551/mitpress/9780262033589.001.0001

Chu, Y., Guo, S., Cui, D., Fu, X. et Ma, Y. (2022). Deephagetp : a convolutional neural network framework for identifying phage-specific proteins from metagenomic sequencing data. *PeerJ*, *10*, e13404. http://dx.doi.org/10.7717/peerj.13404. Récupéré de https://doi.org/10.7717/peerj.13404

Clark, K., Khandelwal, U., Levy, O. et Manning, C. D. (2019). What does bert look at ? an analysis of bert's attention. Récupéré de https://arxiv.org/abs/1906.04341

Consortium, T. U. (2025). Uniprot : the universal protein knowledgebase in 2025. *Nucleic Acids Research*, *53*(D1), D609–D617. http://dx.doi.org/10.1093/nar/gkae1010. Récupéré de https://doi.org/10.1093/nar/gkae1010

Dao, T. et Gu, A. (2024). Transformers are ssms : Generalized models and efficient algorithms through structured state space duality. Récupéré de https://arxiv.org/abs/2405.21060

DeepSeek-AI *et al.* (2025). Deepseek-v3 technical report. Récupéré de https://arxiv.org/abs/2412.19437

Devlin, J., Chang, M.-W., Lee, K. et Toutanova, K. (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. Dans J. Burstein, C. Doran, et T. Solorio (dir.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186., Minneapolis, Minnesota. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/N19-1423. Récupéré de https://aclanthology.org/N19-1423/

Dilthey, A. T. *et al.* (2019). Strain-level metagenomic assignment and compositional estimation for long reads with MetaMaps. *Nature Communications*, *10*(1), 3066. http://dx.doi.org/10.1038/s41467-019-10934-2

Doersch, C. et Zisserman, A. (2017). Multi-task self-supervised visual learning. Dans *2017 IEEE International Conference on Computer Vision (ICCV)*, 2070–2079. http://dx.doi.org/10.1109/ICCV.2017.226

Dyer, S. C. *et al.* (2025). Ensembl 2025. *Nucleic Acids Research*, *53*(D1), D948–D957. http://dx.doi.org/10.1093/nar/gkae1071. Récupéré de https://doi.org/10.1093/nar/gkae1071

Fan, J. *et al.* (2021). BugSeq : a highly accurate cloud platform for long-read metagenomic analyses. *BMC Bioinformatics*, *22*(1), 160. http://dx.doi.org/https://doi.org/10.1186/s12859-021-04089-5

Fang, Z., Tan, J., Wu, S., Li, M., Wang, C., Liu, Y. et Zhu, H. (2020). Plasgun : gene prediction in plasmid metagenomic short reads using deep learning. *Bioinformatics*, *36*(10), 3239–3241. http://dx.doi.org/10.1093/bioinformatics/btaa103. Récupéré de https://doi.org/10.1093/bioinformatics/btaa103

Glorioso, P., Anthony, Q., Tokpanov, Y., Whittington, J., Pilault, J., Ibrahim, A. et Millidge, B. (2024). Zamba : A compact 7b ssm hybrid model. Récupéré de https://arxiv.org/abs/2405.16712

Goldfarb, T. *et al.* (2024). Ncbi refseq : reference sequence standards through 25 years of curation and annotation. *Nucleic Acids Research*, *53*(D1), D243–D257. http://dx.doi.org/10.1093/nar/gkae1038. Récupéré de https://doi.org/10.1093/nar/gkae1038

Gondara, L. (2016). Medical image denoising using convolutional denoising autoencoders. Dans *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 241–246. http://dx.doi.org/10.1109/ICDMW.2016.0041

Goodfellow, I., Bengio, Y. et Courville, A. (2016). *Deep Learning*. MIT Press. Récupéré de http://www.deeplearningbook.org

Gu, A. et Dao, T. (2024). Mamba : Linear-time sequence modeling with selective state spaces. Dans *First Conference on Language Modeling*. Récupéré de https://openreview.net/forum?id=tEYskw1VY2

Hasty, T. *et al.* (2009). *The Elements of Statistical Learning*. New York : Springer New York. http://dx.doi.org/https://doi.org/10.1007/978-0-387-84858-7

He, K., Zhang, X., Ren, S. et Sun, J. (2016). Deep residual learning for image recognition. Dans *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. http://dx.doi.org/10.1109/CVPR.2016.90

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257. http://dx.doi.org/https://doi.org/10.1016/0893-6080(91)90009-T. Récupéré de https://www.sciencedirect.com/science/article/pii/089360809190009T

HuggingFace (2026). Byte-pair encoding tokenization. Récupéré de https://huggingface.co/learn/llm-course/chapter6/5

Huson, D. H. *et al.* (2018). MEGAN-LR : new algorithms allow accurate binning and easy interactive exploration of metagenomic long reads and contigs. *Biology Direct*, *13*(1), 6. PMID : 29678199 ; PMCID : PMC5910613, http://dx.doi.org/10.1186/s13062-018-0208-7

Irber, L. *et al.* (2024). sourmash v4 : A multitool to quickly search, compare, and analyze genomic and metagenomic data sets. *Journal of Open Source Software*, *9*(98), 6830. http://dx.doi.org/10.21105/joss.06830

ISO/CEI (2023). ISO/IEC 25010 :2023 - systems and software engineering — systems and software quality requirements and evaluation (square) — product quality model and system in use quality model. Récupéré de https://www.iso.org/standard/78174.html

Johnson, J. S., Spakowicz, D. J., Hong, B.-Y., Petersen, L. M., Demkowicz, P., Chen, L., Leopold, B., Hanson, B. M., Agresta, M. J., Gerstein, M. *et al.* (2019). Evaluation of 16s rrna gene sequencing for species and strain-level microbiome analysis. *Nature Communications*, *10*(1), 5029. http://dx.doi.org/10.1038/s41467-019-13036-1. Récupéré de

https://doi.org/10.1038/s41467-019-13036-1

Kallenborn, F. *et al.* (2025). GPU-accelerated homology search with MMseqs2. *Nature Methods*, *22*(1), 2024–2027. http://dx.doi.org/10.1038/s41592-025-02819-8

Kaplan, J., McCandlish, S., Hernandez, D., Brown, T. B., Thompson, J., Vos, C., Radford, A., Wu, J. et Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv :2001.08361*. Récupéré de https://arxiv.org/abs/2001.08361

Karagöz, M. A. et Nalbantoglu, O. U. (2021). Taxonomic classification of metagenomic sequences from relative abundance index profiles using deep learning. *Biomedical Signal Processing and Control*, *67*, 102539. http://dx.doi.org/https://doi.org/10.1016/j.bspc.2021.102539. Récupéré de https://www.sciencedirect.com/science/article/pii/S1746809421001361

Katharopoulos, A., Vyas, A., Pappas, N. et Fleuret, F. (2020). Transformers are rnns : Fast autoregressive transformers with linear attention. Récupéré de https://arxiv.org/abs/2006.16236

Kim, D. *et al.* (2016). Centrifuge : rapid and sensitive classification of metagenomic sequences. *Genome Research*, *26*(12), 1721–1729. PMID : 27852649 ; PMCID : PMC5131823, http://dx.doi.org/10.1101/gr.210641.116

Kingma, D. P. et Ba, J. (2017). Adam : A method for stochastic optimization. Récupéré de https://arxiv.org/abs/1412.6980

Krizhevsky, A., Sutskever, I. et Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, *60*(6), 84–90. http://dx.doi.org/10.1145/3065386. Récupéré de https://doi.org/10.1145/3065386

Langmead, B. et Salzberg, S. L. (2012). Fast gapped-read alignment with bowtie 2. *Nature Methods*, *9*(4), 357–359. http://dx.doi.org/10.1038/nmeth.1923. Récupéré de https://doi.org/10.1038/nmeth.1923

LeCun, Y., Bengio, Y. et Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. http://dx.doi.org/10.1038/nature14539. Récupéré de https://doi.org/10.1038/nature14539

Liang, Q., Bible, P. W., Liu, Y., Zou, B. et Wei, L. (2020). Deepmicrobes : taxonomic classification for metagenomics with deep learning. *NAR Genomics and Bioinformatics*, *2*(1), lqaa009. http://dx.doi.org/10.1093/nargab/lqaa009. Récupéré de https://doi.org/10.1093/nargab/lqaa009

Lin, T.-Y., Goyal, P., Girshick, R., He, K. et Dollár, P. (2017). Focal Loss for Dense Object Detection. *arXiv e-prints*, p. arXiv :1708.02002. http://dx.doi.org/10.48550/arXiv.1708.02002

Liu, B. et Pop, M. (2009). Ardb—antibiotic resistance genes database. *Nucleic Acids Research*, *37*(suppl_1), D443–D447. http://dx.doi.org/10.1093/nar/gkn656. Récupéré de https://doi.org/10.1093/nar/gkn656

Liu, F., Miao, Y., Liu, Y. et Hou, T. (2022). Rnn-virseeker : A deep learning method for identification of short viral sequences from metagenomes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *19*(3), 1840–1849. http://dx.doi.org/10.1109/TCBB.2020.3044575

Liu, L., Liu, X., Gao, J., Chen, W. et Han, J. (2020). Understanding the difficulty of training transformers. Dans B. Webber, T. Cohn, Y. He, et Y. Liu (dir.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 5747–5763., Online. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/2020.emnlp-main.463. Récupéré de https://aclanthology.org/2020.emnlp-main.463/

Liu, S. *et al.* (2025). Analysis of metagenomic data. *Nature Reviews Methods Primers*, *5*(1), 5. PMID : 40688383 ; PMCID : PMC12276902, http://dx.doi.org/10.1038/s43586-024-00376-6

Lu, J. *et al.* (2017). Bracken : estimating species abundance in metagenomics data. *PeerJ Computer Science*, *3*, e104. http://dx.doi.org/10.7717/peerj-cs.104

Lu, J. *et al.* (2022). Metagenome analysis using the Kraken software suite. *Nature Protocols*, *17*(12), 2815–2839. http://dx.doi.org/10.1038/s41596-022-00738-y

Martin, E. et Cundy, C. (2018). Parallelizing linear recurrent neural nets over sequence length. Dans *International Conference on Learning Representations*. Récupéré de https://openreview.net/forum?id=HyUNwulC-

Matchado, M. S. *et al.* (2024). On the limits of 16s rrna gene-based metagenome prediction and functional profiling. *Microbial Genomics*, *10*(2), 001203. PMID : 38421266 ; PMCID : PMC10926695, http://dx.doi.org/10.1099/mgen.0.001203

Mock, F., Kretschmer, F., Kriese, A., Böcker, S. et Marz, M. (2022). Taxonomic classification of dna sequences beyond sequence similarity using deep neural networks. *Proceedings of the National Academy of Sciences*, *119*(35), e2122636119. http://dx.doi.org/10.1073/pnas.2122636119. Récupéré de https://www.pnas.org/doi/abs/10.1073/pnas.2122636119

Nalbantoglu, O. U., Way, S. F., Hinrichs, S. H. et Sayood, K. (2011). Raiphy : Phylogenetic classification of metagenomics samples using iterative refinement of relative abundance index profiles. *BMC Bioinformatics*, *12*(41). Récupéré de http://dx.doi.org/10.1186/1471-2105-12-41

Navgire, G. S. *et al.* (2022). Analysis and interpretation of metagenomics data : an approach. *Biological Procedures Online*, *24*(1), 18. PMID : 36402995 ; PMCID : PMC9675974. Erratum in : Biol Proced Online. 2024 ;26(1) :8, http://dx.doi.org/10.1186/s12575-022-00179-7

Parks, D. H. *et al.* (2025). Gtdb release 10 : a complete and systematic taxonomy for 715230 bacterial and 17245 archaeal genomes. *Nucleic Acids Research*, *54*(D1), D743–D754. http://dx.doi.org/10.1093/nar/gkaf1040. Récupéré de https://doi.org/10.1093/nar/gkaf1040

Pérez-Cobas, A. E. *et al.* (2020). Metagenomic approaches in microbial ecology : an update on whole-genome and marker gene sequencing analyses. *Microbial Genomics*, *6*(8), mgen000409. PMID : 32706331 ; PMCID : PMC7641418, http://dx.doi.org/10.1099/mgen.0.000409

Pinheiro Cinelli, L., Araújo Marins, M., Barros da Silva, E. A. et Lima Netto, S. (2021). *Variational Autoencoder*, Dans *Variational Methods for Machine Learning with Applications to Deep Networks*, (p. 111–149). Springer International Publishing : Cham

Portik, D. M., Brown, C. T. et Pierce-Ward, N. T. (2022). Evaluation of taxonomic classification and profiling methods for long-read shotgun metagenomic sequencing datasets. *BMC Bioinformatics*, *23*(1), 541. http://dx.doi.org/10.1186/s12859-022-05103-0. Récupéré de https://doi.org/10.1186/s12859-022-05103-0

Ren, J., Song, K., Deng, C., Ahlgren, N. A., Fuhrman, J. A., Li, Y., Xie, X., Poplin, R. et Sun, F. (2020). Identifying viruses from metagenomic data using deep learning. *Quantitative Biology*, *8*(1), 64–77. http://dx.doi.org/https://doi.org/10.1007/s40484-019-0187-4. Récupéré de https://onlinelibrary.wiley.com/doi/abs/10.1007/s40484-019-0187-4

Ren, L., Liu, Y., Lu, Y., Shen, Y., Liang, C. et Chen, W. (2025). Samba : Simple hybrid state space models for efficient unlimited context language modeling. Récupéré de https://arxiv.org/abs/2406.07522

Roy, G., Prifti, E., Belda, E. et Zucker, J.-D. (2024). Deep learning methods in metagenomics : a review. *Microbial Genomics*, *10*(4). http://dx.doi.org/https://doi.org/10.1099/mgen.0.001231. Récupéré de https://www.microbiologyresearch.org/content/journal/mgen/10.1099/mgen.0.001231

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215. http://dx.doi.org/10.1038/s42256-019-0048-x. Récupéré de https://doi.org/10.1038/s42256-019-0048-x

Ruscheweyh, H. *et al.* (2021). Reference genome-independent taxonomic profiling of microbiomes with mOTUs3. bioRxiv pre-print

Sak, H., Senior, A. W. et Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Dans *INTERSPEECH*, 338–342.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229. http://dx.doi.org/10.1147/rd.33.0210. Récupéré de https://ieeexplore.ieee.org/abstract/document/5392560

Satam, H. *et al.* (2023). Next-generation sequencing technology : Current trends and advancements. *Biology*, *12*(7). http://dx.doi.org/10.3390/biology12070997. Récupéré de https://www.mdpi.com/2079-7737/12/7/997

Schoch, C. L., Ciufo, S., Domrachev, M., Hotton, C. L., Kannan, S., Khovanskaya, R., Leipe, D., Mcveigh, R., O'Neill, K., Robbertse, B., Sharma, S., Soussov, V., Sullivan, J. P., Sun, L., Turner, S. et Karsch-Mizrachi, I. (2020). Ncbi taxonomy : a comprehensive update on curation, resources and tools. *Database (Oxford)*, *2020*, baaa062. http://dx.doi.org/10.1093/database/baaa062

Sutton, R. S. et Barto, A. G. (2014). *Reinforcement Learning : An Introduction* (second éd.). MIT press.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York : Springer-Verlag. http://dx.doi.org/10.1007/978-1-4757-3264-1

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. et Polosukhin, I. (2017). Attention is all you need. Récupéré de https://arxiv.org/pdf/1706.03762.pdf

Wood, D. E. *et al*. (2014). Kraken : ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, *15*(3), R46. PMID : 24580807; PMCID : PMC4053813, http://dx.doi.org/10.1186/gb-2014-15-3-r46

Wu, W., Li, Q., Zhang, Y., Zhan, Z., Chen, R., Li, M., Fu, K., Qi, J., Bao, Y., Wang, C., Zhu, Y., Zhang, Z., Tang, J., Feng, F., Ye, J., Liu, Y., Xiong, H. et Wang, Z. (2026). Generator : A long-context generative genomic foundation model. Récupéré de https://arxiv.org/abs/2502.07272

Wu C, Tygert M, L. Y. (2019). A hierarchical loss and its problems when classifying non-hierarchically. *PLoS ONE*. http://dx.doi.org/https://doi.org/10.1371/journal.pone.0226222

Zha, Y., Chong, H., Qiu, H. et et al. (2022). Ontology-aware deep learning enables ultrafast and interpretable source tracking among sub-million microbial community samples from hundreds of niches. *Genome Medicine*, *14*(1), 1–17. http://dx.doi.org/10.1186/s13073-022-01047-5. Récupéré de https://doi.org/10.1186/s13073-022-01047-5

Zhou, Z., Ji, Y., Li, W., Dutta, P., Davuluri, R. et Liu, H. (2024). Dnabert-2 : Efficient foundation model and benchmark for multi-species genome. Récupéré de https://arxiv.org/abs/2306.15006