Vers une Intelligence Artificielle Numérique Responsable

Angela Voinea Ciocan

LA ROCHELLE UNIVERSITÉ

ÉCOLE DOCTORALE EUCLIDE

LABORATOIRE L3i

THÈSE présentée par :

Angela CIOCAN (VOINEA)

soutenue le : **11 Juillet 2024**
pour obtenir le grade de : **Docteur de La Rochelle Université**
Discipline : **Informatique et Applications**

Vers une Intelligence Artificielle Numérique Responsable

Président du jury	Karell BERTET	Professeur des universités	La Rochelle Université
Rapporteurs	Christine FERNANDEZ-MALOIGNE	Professeur des universités	Université de Poitiers
	Denis HAMAD	Professeur des universités	Université du Littoral Côte d'Opale
Direction	Vincent COURBOULAY	Maître de conférences HDR	La Rochelle Université

Thèse réalisée au Laboratoire L3i
La Rochelle Université - Institut LUDI
Avenue Michel Crépeau
17042 La Rochelle cedex 01

Tel : +33 5 46 45 82 62
Web : `http://l3i.univ-larochelle.fr`

Sous la direction de Vincent Courboulay, Maitre de conférences HDR

Financement Financé par le Ministère de l'Enseignement Supérieur
et de la Recherche (MESR) qui en confie la gestion à l'ANRT

Résumé

Le numérique pollue, et l'Intelligence Artificielle tout particulièrement !

La consommation d'énergie est de plus en plus une préoccupation dans de nombreux domaines, y compris pour les domaines de l'IA.

Un constat partagé, la consommation d'énergie de l'IA a considérablement augmenté au cours des dernières années, tant pour la partie matérielle nécessaire pour la construction des solutions que pour les solutions en elles-mêmes, notamment en raison de l'explosion des services basés sur l'IA et de l'énorme quantité de données nécessaire.

Dans le passé, la consommation d'énergie était principalement un sujet lié au matériel utilisé avec sa capacité d'utiliser le moins possible d'énergie pour effectuer des tâches. Aujourd'hui, la partie applicative est aussi importante que le matériel, et elle est également responsable de l' augmentation de l'énergie consommée.

Avec l'émergence du Cloud Computing et l'adoption de plus en plus des infrastructures hébergées dans le Cloud ont modifié la perception des entreprises et des usagés sur la consommation énergétique des applications IA. Avec des ressources matérielles et des services dématérialisées, facilement scalables à travers les plate-formes Cloud, beaucoup d'entreprises tendent à ne pas donner d'importance à l'optimisation énergétique de leurs solutions d'IA.

L'objectif de cette thèse est d'aider les entreprises et les usagés à comprendre et à introduire la conception de solutions d'IA plus responsables, afin de réduire leur consommation d'énergie et de fournir des services plus frugaux en termes de consommation d'énergie. Nous contribuons de cette manière à la conception de solutions IA plus écologiques et plus durables.

Pour le faire, nous commençons par mener un état de l'art sur l'évolution des solutions IA, afin de discuter des multiples facettes auxquels il faut s'intéresser une fois lancé dans une démarche responsable autour du numérique, et particulièrement l'IA.

Pour réduire la consommation énergétique des solutions IA, les entreprises et l'ensemble des usages doivent d'abord être capable de la mesurer et suivre son évolution. Le choix des outils de mesure de ce type de solution reste encore un vrai challenge aujourd'hui du fait que la mesure doit être réalisée sur l'ensemble du cycle de vie des solutions afin de pouvoir mettre en évidence les parties qui nécessitent une optimisation afin de réduire la consommation d'énergie.

Une fois les outils capables de mesurer la consommation d'énergie des applications IA sont en place, le travail d'analyse et d'optimisation pour procéder à la réduction d'énergie et la réalisation des solutions plus responsable de point de vue numérique peut commencer.

Cette thèse propose aux entreprises des solutions concrètes pour rendre l'utilisation des applications IA plus responsable et surtout des solutions sans besoin de remettre en cause la partie matérielle déjà en place.

Mots clés : Intelligence Artificielle, Deep Learning, NLP, Numérique Responsable, Énergie, Consommation, Durable

Towards Sustainability Artificial Intelligence

Abstract

Digital technology pollutes, and Artificial Intelligence (AI) in particular !

Energy consumption is increasingly a concern in many areas, including AI domains.

A shared observation is that AI's energy consumption has significantly increased over the past few years, both for the hardware needed to build solutions and for the solutions themselves, notably due to the explosion of AI-based services and the vast amount of data required.

In the past, energy consumption was mainly a topic related to the hardware used, with its ability to use as little energy as possible to perform tasks. Today, the application part is as important as the hardware, and it is also responsible for the increase in energy consumed.

The goal of this thesis is to help companies and users understand and introduce the design of more responsible AI solutions, in order to reduce their energy consumption and provide more frugal services in terms of energy use. In this way, we contribute to the design of greener and more sustainable AI solutions.

To do this, we begin by conducting a state-of-the-art review on the evolution of AI solutions, in order to discuss the multiple facets that must be considered once embarked on a responsible approach around digital technology, and particularly AI.

To reduce the energy consumption of AI solutions, companies and all users must first be able to measure it and track its evolution. The choice of measurement tools for this type of solution remains a real challenge today because the measurement must be carried out over the entire lifecycle of the solutions in order to highlight the parts that require optimization to reduce energy consumption.

Once the devices (or solutions) capable of measuring the energy consumption of AI applications are in place, the work of analysis and optimization to proceed with energy reduction and the realization of more digitally responsible solutions can begin.

Furthermore, the emergence of cloud computing and the growing adoption of cloud infrastructures have changed companies' perception of energy consumption. With the dematerialized and easily scalable resources offered by the cloud, many companies tend to give less importance to the energy optimization of their AI solutions.

This thesis offers companies concrete solutions to make the use of AI applications more responsible and especially solutions that do not require questioning the already existing hardware part.

Keywords : Artificial Intelligence, Deep Learning, NLP, Responsible Digital, Energy, Consumption, Sustainable

Remerciements

Je souhaite exprimer ma profonde gratitude à tous ceux qui m'ont aidée et soutenue tout au long de ce parcours de thèse, ainsi qu'à tous ceux qui ont montré une sensibilité particulière pour ce sujet.

Tout d'abord, je tiens à remercier Vincent Courboulay, mon directeur de thèse, qui a eu une foi totale en moi en me confiant ce sujet, à une époque où il était encore relativement nouveau et peu exploré. Sa vision et son soutien ont été inestimables pour la réussite de cette recherche.

Je souhaite également remercier l'entreprise qui m'a accueillie, France Travail anciennement Pôle Emploi, en particulier mon premier responsable, Philippe Varela, qui m'a initiée aux mesures énergétiques et à l'importance de la sensibilité dans ce domaine. Un grand merci aussi à Martine Loubriat pour la partie RSE, pour son engagement et son soutien.

Un remerciement spécial à Rozenn Brenin, qui a vu le potentiel de ce sujet et m'a accueillie dans une deuxième structure, m'accompagnant pendant une période importante de ma thèse. Sa sensibilité pour ce sujet m'a guidée vers les algorithmes de reconnaissance de texte, enrichissant ainsi mon travail de recherche.

Je tiens à exprimer ma gratitude à mon jury : Karell Bertet, Christine Fernandez, Denis Hamad, d'une qualité et d'une excellence remarquables, pour avoir accepté de me donner leurs avis éclairés et constructifs. Merci également au laboratoire L3i de La Rochelle Université, et représenté par M. Yacine Ghamri-Doudane, pour son soutien tout au long de ce projet.

Enfin, je ne pourrais jamais assez remercier ma famille. Sans leur soutien indéfectible, cette thèse n'aurait pas été possible. Un merci tout particulier à mon mari, Adrian, qui a été mon conseiller, mon soutien, et a fait preuve d'un grand sacrifice tout au long de cette période. Même si les mots me manquent pour exprimer toute ma gratitude, nous avons toujours fonctionné sans mots pour nous comprendre.

Merci à tous, du fond du cœur.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Sommaire

L'évolution rapide du secteur numérique, notamment avec l'avènement de l'intelligence artificielle, a suscité des préoccupations croissantes concernant son impact environnemental. Une étude recente [ECF22] révèle que le numérique représente déjà une part significative des émissions de gaz à effet de serre à l'échelle mondiale. L'entraînement des modèles d'IA, notamment ceux utilisés dans le traitement du langage naturel, représente un impact environnemental considérable, comme en témoignent les recherches de l'Université du Massachusetts. Face à ces défis, il est impératif de prendre des mesures pour évaluer et minimiser l'impact écologique de l'IA, tout en continuant à exploiter ses avantages.

1.1 Problématique de recherche

Le numérique pollue, et l'Intelligence Artificielle (IA) tout particulièrement ! Un constat de plus en plus pointé du doigt par les institutions publiques, les chercheurs et les médias partout dans le monde.

D'après une étude réalisée conjointement par ADEME [1] et l'Arcep [2] publiée le 19 janvier 2022 [ECF22] [3], « le numérique représenterait aujourd'hui 3 à 4 % des émissions de gaz à effet de serre (GES) [4] dans le monde et 2 % de l'empreinte carbone au niveau national (phases de fabrication et d'utilisation comprises) ».

Selon des chercheurs de l'Université du Massachusetts, aux Etats-Unis, l'IA ne fait que raffiner des données comme l'on raffinerait du pétrole : tout comme l'exploitation de l'or noir, les méthodes d'apprentissage profond ont en effet un impact environnemental considérable. Entraîner un modèle de Deep Learning (DL) [5] pour le traitement du langage naturel (NLP) [6] émet autant de CO_2 qu'un être humain pendant 57 ans, ou que 5 voitures pendant leur durée de vie [SGM20].

Dans le domaine de l'IA, le NLP, au fur et à mesure que les assistants vocaux et les services de traduction instantanée se développent l'empreinte carbone pour la mise en place de nouveaux modèles augmente de manière inquiétante. Pour interpréter une commande vocale adressée à Google Assistant et pour lui apprendre à reconnaître votre voix, pour interpréter des textes complexes, ou encore pour traduire un document, il faut une énorme quantité d'énergie car il faut entraîner en profondeur les algorithmes à partir

1. L'Agence de l'environnement et de la maîtrise de l'énergie (Ademe) est un établissement public à caractère industriel et commercial français créé en 1991. Elle affiche également le nom d'« Agence de la transition écologique ». Elle est placée sous la tutelle des ministères de l'Enseignement supérieur, de la Recherche et de l'Innovation et de la Transition écologique et solidaire. L'Ademe suscite, anime, coordonne, facilite ou réalise des opérations de protection de l'environnement et la maîtrise de l'énergie, avec un budget de 690 millions € en diminution (605 millions prévus en 2019) pour un effectif salarié de 963 équivalents temps-plein.

2. L'Autorité de régulation des communications électroniques, des postes et de la distribution de la presse (Arcep) est une autorité administrative indépendante française chargée de réguler les communications électroniques et postales et la distribution de la presse.

3. Evaluation de l'impact environnemental du numérique en France et analyse prospective, Note de synthèse réalisée par l'ADEME et l'Arcep

4. Les gaz à effet de serre (GES) sont des composants gazeux qui absorbent le rayonnement infrarouge émis par la surface terrestre et contribuent ainsi à l'effet de serre. L'augmentation de leur concentration dans l'atmosphère terrestre est l'un des facteurs à l'origine du changement climatique.

5. DL : L'apprentissage profond, Deep Learning en anglais, est le sous-ensemble des méthodes d'apprentissage automatique basées sur des réseaux de neurones. L'adjectif « profond » fait référence à l'utilisation de plusieurs couches dans le réseau.

6. NLP, pour Natural Language Processing, est une discipline qui porte essentiellement sur la compréhension, la manipulation et la génération du langage naturel par les machines. Ainsi, le NLP est réellement à l'interface entre la science informatique et la linguistique. Il porte donc sur la capacité de la machine à interagir directement avec l'humain.

d'une grande masse de données, et cela des centaines, voire des milliers de fois, pendant des semaines ou des mois, à l'aide des ordinateurs surpuissants.

Ces modèles sont coûteux à former et à développer, tant sur le plan financier, en raison du coût du matériel et de l'électricité ou du temps de calcul, que sur le plan environnemental, en raison de l'empreinte carbone nécessaire pour alimenter le matériel moderne de traitement des algorithmes et c'est aussi l'avis des scientifiques de l'Université du Massachusetts dans leur étude. Ils ont ainsi pu estimer ce que consomment les méthodes de Deep Learning, en équivalent CO_2, sur la base du mix énergétique moyen aux USA (17 % d'énergies renouvelables, 35 % de gaz, 27 % de charbon, et 19 % de nucléaire). Mais ces chiffres ne comprennent que l'impact énergétique et les émissions de gaz à effet de serre associées.

La production de ce matériel nécessite l'extraction de terres rares et minéraux épuisant les ressources de la planète, émettant beaucoup de gaz à effet de serre et consommant beaucoup d'eau, de pétrole et de produits chimiques. Toute cette chaîne de traitement contribue fortement à l'épuisement des ressources au sens général et cela amène un grand risque pour les générations futures.

Pour pouvoir mesurer tous les effets de l'IA sur l'environnement il faut prendre en compte la totalité de la chaîne, à partir de la production du matériel, le stockage des données, leur utilisation à travers des usages, jusqu'à leur destruction. L'approche la plus pertinente, qui est d'ailleurs de plus en plus présente dans les études récentes, reste l'analyse du cycle de vie (ACV) [7] qui se base sur des normes et référentiels publics spécifiques. Son objectif est de rendre le périmètre d'analyse plus complet en décomposant le numérique en trois composants matériels principaux : les terminaux, les réseaux et les centres de données. Cette approche est caractérisée par sa prise en compte multicritères, notamment à travers l'évaluation de l'impact environnemental du numérique à l'aide de 11 indicateurs environnementaux supplémentaires en plus de l'empreinte carbone (épuisement des ressources abiotiques, acidification, écotoxicité, radiations ionisantes, émissions de particules fines, création d'ozone, matières premières, production de déchets, consommation d'énergie primaire, consommation d'énergie finale) [ECF22]. De plus, elle intègre une analyse qui considère les impacts environnementaux générés à chaque étape du cycle de vie de ces trois composants, incluant la fabrication, la distribution, l'utilisation et la fin de vie, ce qui en fait une approche multi-étapes de l'ACV.

Les données, elles, doivent être stockées de manière non volatile sur des différents supports, comme par exemple les disques durs, les disques SSD, les serveurs de fichiers

7. L'analyse du cycle de vie (ACV) est une méthode d'évaluation normalisée (ISO 14040 et 14044) permettant de réaliser un bilan environnemental multicritère et multi-étape d'un système (produit, service, entreprise ou procédé) sur l'ensemble de son cycle de vie. Son but est de connaître et pouvoir comparer les impacts environnementaux d'un système tout au long de son cycle de vie, de l'extraction des matières premières nécessaires à sa fabrication à son traitement en fin de vie (mise en décharge, recyclage...), en passant par ses phases d'usage, d'entretien et de transport.

(NAS) ou les services spécialisés dans le Cloud. Une donnée gardée quelques jours n'influence pas autant qu'une donnée stockée pendant des années, car même inactif, un disque consomme de l'énergie et doit être remplacé après un certain temps. Il faut aussi prendre en compte dans les données leurs modes de stockages mirrorés, répliqués et sauvegardés.

Par exemple, dans la mesure de l'efficience des data centers via le calcul du Power Usage Effectivenness (PUE) [8], mis au point par le Green Grid [9], apporte un indicateur supplémentaire sur la consommation énergétique des services numériques hébergés. En Europe le PUE moyen d'un data center se situe autour de 2.5 kilowatts alors que un data center écologique tend vers un PUE le plus proche possible 1 kilowatt. On voit donc que l'influence de l'IA est vaste et protéiforme. Des minéraux à l'énergie primaire en passant par les déchets et la consommation énergétique des data centers, l'IA a des impactes.

Les chiffres actuels sont déjà énormes, mais leur importance le sera encore plus si l'on considère que les recherches dans ce domaine ne sont pas près de s'arrêter, et donc d'utiliser toujours plus de données et d'énergie et de matériaux, dans un contexte de course à l'IA.

Une solution simple : ***la frugalité****.*

Les travaux réalisés dans cette thèse s'inscrivent dans le cadre d'un dispositif de recherche CIFRE mis en place entre le Laboratoire Informatique Image et Interaction (L3i) de La Rochelle Université et Pôle Emploi, devenu France Travail depuis 2024, qui est un établissement public national à caractère administratif, chargé de l'emploi en France.

Pôle Emploi, engagé dans la démarche de sobriété numérique, s'est fixé comme ambition d'apporter une nouvelle vision à la conception, le développement et l'usage des applications avec un fort impact énergétique. Dans leur patrimoine applicatif, et spécifiquement celui lié aux applications IA, deux natures d'applications sont présentes : une qui est dédiée au fonctionnement et aux usages internes de l'entreprise et l'autre qui couvre plutôt les besoins des clients externes. Nos travaux de recherche vise la partie d'usage interne, avec une représentation dans la Figure 1.1.

Les techniques employées aujourd'hui dans le domaine de l'IA, les méthodes d'apprentissage automatique et d'apprentissage profond, respectivement Machine Learning (ML) et Deep Learning (DL), sont consommatrice car elles nécessitent de plus en plus de données qu'elles agrègent en continu afin de proposer des services de plus en plus performants.

8. L'indicateur d'efficacité énergétique (en anglais PUE ou Power Usage Effectiveness) est utilisé pour qualifier l'efficacité énergétique d'un centre d'exploitation informatique. C'est un des éléments de l'informatique écoresponsable.

9. The Green Grid

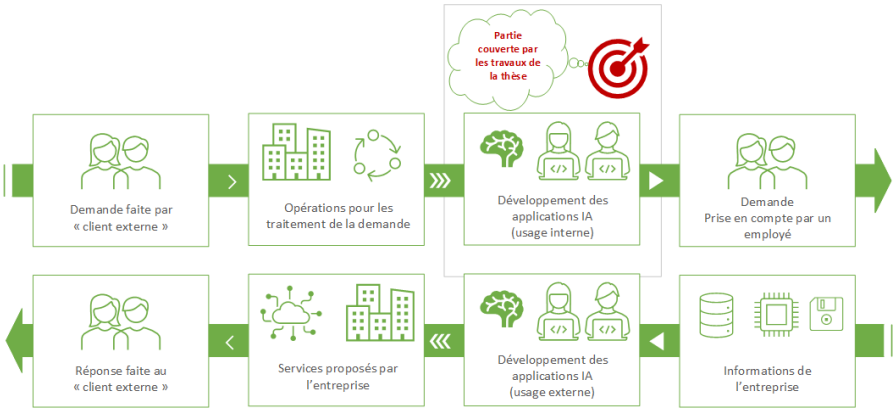

FIGURE 1.1 – Représentation du contexte et du périmètre des travaux

Dans le portefeuille des applications IA avec un usage interne les applications destinées à la classification des mails ont été embarquées dans le périmètre de nos travaux avec la volonté de proposer une démarche de réduction de la consommation des applications IA de type NLP. Le positionnement de ce type d'applications dans le vaste domaine de l'IA est dans la Figure 1.2.

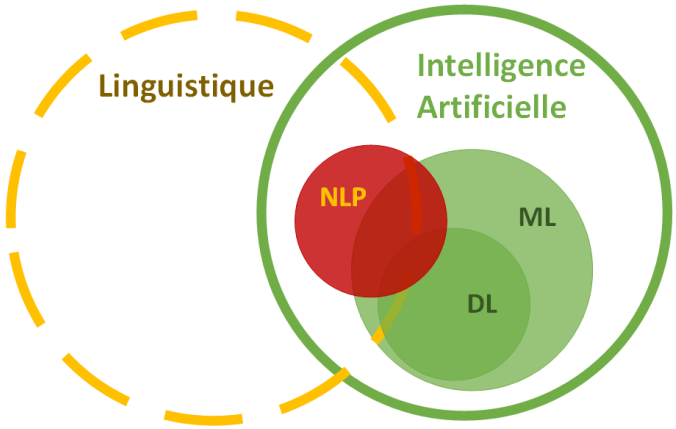

FIGURE 1.2 – La place du NLP dans le domaine IA

Pour réduire l'énergie consommée par ces techniques, nos recherches viseront la mise en place de méthodes afin de réaliser des modèles permettant d'économiser l'énergie nécessaire à leurs utilisation.

L'objectif principal de la thèse est d'introduire de nouvelles méthodes pour aborder la question de la consommation énergétique des services numériques, en mettant l'accent sur les services d'IA, à travers une analyse complète du cycle de vie d'un service numé-

rique IA, et dans une perspective d'alignement à la démarche d'amélioration continue du Numérique Responsable (NR) [10].

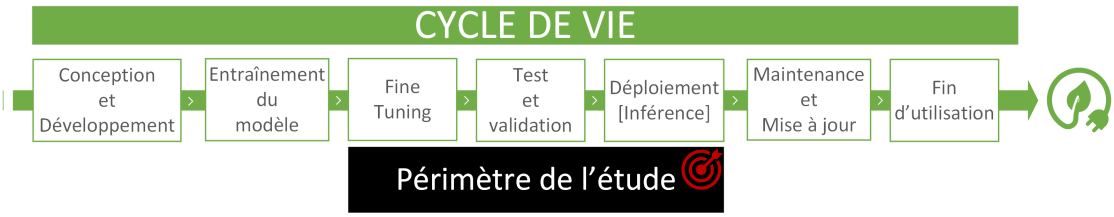

FIGURE 1.3 – Les étapes du cycle de vie

1.2 Plan du mémoire

Le plan de ce mémoire veut être le reflet d'un parcours analytique et exploratoire à travers les différents aspects de l'IA vue sous un regard numérique responsable. La structuration des différents chapitres permet d'aborder de façon exhaustive un spectre large de ce domaine, avec un niveau profond de détails sur certaines parties, les fondements théoriques jusqu'aux implications pratiques avec un regard particulier sur l'utilisation des modèles d'IA sur le terrain, communément appelé la phase d'inférence.

Le mémoire est structuré en plusieurs parties de la manière suivante :

Chapitre 2 : Les Fondements d'une IA Numérique Responsable

Examen approfondi des principes et pratiques de l'IA Numérique Responsable, couvrant l'éthique, la transparence, la durabilité et l'équité. Discussion sur les défis et les implications de l'intégration de ces principes dans les technologies actuelles.

Chapitre 3 : IA Numérique Responsable, un État de l'Art

Ce chapitre se concentrera sur l'état actuel de l'IA Numérique Responsable, en explorant les différentes approches et méthodologies utilisées dans ce domaine. Il mettra

10. Le Numérique Responsable est une démarche d'amélioration continue qui vise à réduire l'empreinte écologique, économique et sociale des technologies de l'information et de la communication, soit autrement dit, une démarche qui vise à minimiser l'impact environnemental et social du numérique tout en favorisant une utilisation plus durable et éthique des technologies dans le sens large du terme. Le périmètre couvert par rapport à l'ensemble du cycle de vie est représenté dans la Figure 1.3 Guide du Numérique responsable.

en lumière les différences entre l'entraînement et l'inférence dans les algorithmes d'IA, avec un accent particulier sur les réseaux neuronaux en raison de leur consommation énergétique élevée.

Un intérêt particulier est accordé aux algorithmes de traitement automatique du langage naturel (NLP), domaine qui a connu une révolution significative avec l'émergence de nouveaux modèles tels que le modèle BERT [XWvD20], [MA21],[AS23]. Cette avancée, rendue possible grâce à l'adoption généralisée des unités de traitement graphique (GPU), a marqué un tournant dans la reconnaissance du texte, offrant des performances remarquables. Cependant, cette innovation s'accompagne d'un coût énergétique important, soulignant un défi majeur pour l'IA Numérique Responsable : la consommation énergétique des technologies embarquées par l'IA.

Le chapitre se penchera sur la phase d'inférence des réseaux de neurones, une étape importante dans le cycle de vie d'une application d'IA. Nous explorerons pourquoi et comment la phase d'inférence se distingue comme un élément qu'il faudrait explorer en termes de performance et d'efficacité énergétique. Il expliquera pourquoi cette étape est importante dans le cycle de vie d'une application d'IA et comment elle impacte la consommation énergétique. Il présentera également des stratégies pour réduire cette consommation, soutenues par des mesures énergétiques et l'introduction d'un outil de mesure dédié. Ce chapitre mettra en évidence les défis associés à l'optimisation de cette phase, dans le contexte des modèles de NLP avancés, qui requièrent des ressources computationnelles importantes.

Chapitre 4 : FlauBERT et les méthodes de compression

Ce chapitre se concentre sur le modèle FlauBERT, un modèle de traitement du langage naturel spécifique à la langue française. Il décrit l'architecture de FlauBERT ainsi que les méthodes de compression choisies, qui préservent la performance du modèle, tout en permettant une inférence sur des infrastructures à base de processeurs de type CPU, avec une réduction de la consommation énergétique.

Chapitre 5 : Contributions

Ce chapitre met en avant nos travaux pratiques ainsi que les résultats obtenus pour la compression et la mesure des performances des modèles de base et des modèles compressés de FlauBERT.

Chapitre 6 : Conclusions et perspectives

Dans cette conclusion, nous résumons nos efforts pour proposer des stratégies visant à diminuer la consommation énergétique des applications d'intelligence artificielle à court terme, sans nécessiter d'investissements majeurs pour les entreprises.

Le plan du mémoire propose une exploration exhaustive de l'IA Numérique Responsable, de ses fondements théoriques à son application pratique, en mettant l'accent sur des méthodes innovantes pour réduire l'impact environnemental, à travers des solutions de réduction de la consommation d'énergie, tout en préservant l'efficacité et la performance des systèmes d'IA.

Dans cette partie introductive, nous avons évoqué brièvement nos principaux objectifs, un premier focus sur le périmètre ainsi qu'une présentation de la structure de ce mémoire.

Nous allons maintenant plonger plus en profondeur dans le domaine de l'intelligence artificielle et explorer un aspect important, à savoir **Les Fondements d'une Intelligence Artificielle Numérique Responsable**. Ce chapitre vise à établir la définition de l'IA Numérique Responsable, en examinant les concepts, en identifiant les défis qui se posent et en retraçant l'historique de ce domaine en pleine expansion.

Synthèse

→ L'essor de l'intelligence artificielle pose des défis majeurs en matière d'impact environnemental, notamment en raison de la consommation énergétique associée à l'entraînement des modèles, en particulier dans le domaine du traitement du langage naturel.

→ Pour évaluer et atténuer ces impacts, une approche basée sur l'analyse du cycle de vie est essentielle, permettant de prendre en compte toutes les étapes, de la fabrication du matériel à la fin de vie des modèles.

→ En intégrant des pratiques d'écoconception et en explorant des méthodes plus efficientes sur le plan énergétique, nous pouvons progressivement minimiser l'empreinte carbone de l'IA tout en continuant à bénéficier de ses avancées technologiques.

Chapitre 2

Les Fondements d'une Intelligence Artificielle Numérique Responsable

Sommaire

Dans ce chapitre, nous explorons le concept d'Intelligence Artificielle Numérique Responsable, mettant en avant les principes et pratiques pour un développement et une utilisation de l'IA éthiques, transparents, et durables. Nous mettons l'accent sur l'importance de l'IA responsable, pour la gestion environnementale, soulignant la nécessité d'une approche qui bénéficie durablement à l'environnement et tient compte des impacts sur les écosystèmes. Nous abordons également les défis liés à l'adoption de l'IA Numérique Responsable, tels que l'intégration de l'éthique, la transparence, et la durabilité environnementale. Ces principes sont fondamentaux pour développer des systèmes d'IA qui profitent à la société tout en réduisant les risques et impacts négatifs.

2.1 Le Concept de l'IA Numérique Responsable

Le concept de IA Numérique Responsable englobe une série de principes et de pratiques visant à assurer que les systèmes d'IA soient développés et utilisés de manière éthique, transparente, responsable et durable. L'IA responsable dans le contexte de la gestion environnementale consiste à développer et utiliser l'IA d'une manière bénéfique pour l'environnement, durable et consciente des impacts à long terme sur les écosystèmes et la biodiversité. Cela nécessite une approche multidisciplinaire, combinant des perspectives de la technologie, de l'éthique, des sciences de l'environnement et des sciences sociales pour créer des systèmes d'IA qui sont à la fois puissants et responsables.

Le passage vers une IA Numérique Responsable nécessite de s'imposer et de respecter par la suite, quelques principes. Ces principes, incluant l'éthique, la transparence, l'inclusion, la protection de la vie privée, la durabilité environnementale, la responsabilité sociale, la sécurité, l'interopérabilité, l'accessibilité, la collaboration, la gouvernance, l'éducation, la sensibilisation et la conformité légale, sont les piliers d'une IA éthique et durable, Figure 2.1.

FIGURE 2.1 – Mots clés du concept d'IA Numérique Responsable

Chacun de ces aspects est fondamental dans le processus de construction des systèmes IA au profit de la société et de l'environnement, tout en minimisant les risques et impacts négatifs.

Éthique : L'intégration de principes éthiques dans le développement, le déploiement et l'utilisation des systèmes d'IA pour assurer le respect des droits fondamentaux, la justice sociale et l'équité [1]. Intégrer l'éthique dans l'usage de l'IA est essentiel pour plusieurs raisons. Tout d'abord, cela permet de garantir que le développement et l'utilisation de l'IA respectent les valeurs et les droits fondamentaux de l'humanité. Sans une base éthique, l'IA pourrait être utilisée de manière irresponsable, entraînant des conséquences négatives pour les individus et la société.

De plus, l'intégration de l'éthique dans l'IA vise à prévenir les biais algorithmique, la discrimination et les injustices. Les systèmes d'IA sont souvent formés sur des données historiques qui peuvent refléter des préjugés existants. Une approche éthique cherche à atténuer ces biais et à garantir une utilisation équitable de la technologie [SBF+19].

Enfin, l'éthique dans l'IA contribue à établir la confiance du public. Les utilisateurs sont plus enclins à adopter et à soutenir les technologies d'IA si elles sont développées et utilisées de manière transparente, responsable et respectueuse des valeurs morales. L'intégration de l'éthique dans l'IA est essentielle pour préserver nos valeurs, éviter les abus potentiels et établir une confiance durable dans l'utilisation de cette technologie.

Transparence : Fournir des explications claires sur le fonctionnement des systèmes d'IA afin de favoriser la compréhension, la confiance et la responsabilité.

Explicabilité : Assurer que les décisions prises par les systèmes d'IA peuvent être comprises et expliquées de manière claire et compréhensible aux utilisateurs finaux.

Inclusion : Garantir que les systèmes d'IA sont conçus de manière inclusive, en évitant les biais discriminatoires et en assurant l'équité dans l'accès et les résultats.

Protection de la vie privée : Mettre en œuvre des mécanismes de protection de la vie privée pour assurer que les données personnelles sont traitées de manière responsable et conforme aux réglementations en vigueur.

Durabilité environnementale : Adopter des pratiques de développement durable, en minimisant la consommation d'énergie, en utilisant des technologies éco-énergétiques des systèmes d'IA.

Responsabilité sociale : Encourager la responsabilité sociale des entreprises et des développeurs d'IA envers la société, en prenant en compte les conséquences sociales de leurs applications.

Sécurité : Mettre en place des mesures de sécurité robustes pour prévenir les abus, les

1. Les « 23 principes d'Asilomar ».

cyberattaques et les atteintes à la sécurité liées à l'IA.

Interopérabilité et Accessibilité : Les systèmes d'IA doivent être conçus de manière à favoriser l'interopérabilité et l'accessibilité. Cela garantit que ces technologies peuvent être utilisées de manière équitable et inclusive, en évitant la création de barrières technologiques.

Collaboration et Gouvernance : Favoriser la collaboration entre les parties prenantes, y compris le gouvernement, les entreprises, la société civile et les chercheurs, pour élaborer des normes et des réglementations visant à guider l'utilisation responsable de l'IA.

Éducation et Sensibilisation : Promouvoir l'éducation et la sensibilisation sur les implications de l'IA, tant du point de vue des utilisateurs que des concepteurs, pour une utilisation plus informée et responsable.

Conformité légale et normative : Se conformer aux lois et aux normes réglementaires en vigueur pour garantir la légalité et la responsabilité dans le développement et l'utilisation de l'IA.

FIGURE 2.2 – Principes pour le passage vers une IA Numérique Responsable

Pour concevoir une IA Numérique Responsable, il est impératif de considérer l'impact

environnemental de ces technologies, Figure 2.2. Cela implique d'intégrer des pratiques de durabilité dès la conception de l'IA et de s'assurer de leur efficience tout au long du cycle de vie technologique [FC22]. L'intégration des recommandations sur l'éthique de l'IA est également un aspect à prendre en considération [Gab23].

L'IA et l'Environnement : Un Équilibre à Trouver

L'impact environnemental dans le contexte de l'IA responsable implique une réduction de l'empreinte écologique des technologies d'IA et une utilisation proactive pour résoudre les défis environnementaux. Cette démarche nécessite une approche qui prend en compte l'impact environnemental à chaque phase du cycle de vie de l'IA, depuis sa conception jusqu'à son utilisation finale [SGM20].

Dans cette optique, l'IA Numérique Responsable Environnementale [FAH+20] vise à minimiser l'impact écologique des opérations d'IA par des pratiques de réduction de la consommation énergétique des centres de données, d'optimisation du cycle de vie du matériel d'IA et d'utilisation de matériaux durables. Des solutions pour réduire l'impact environnemental de l'IA doivent être explorées. L'application de pratiques d'IA responsables et durables est essentielle pour s'assurer que les avantages de la technologie ne soient pas annulées par ses coûts environnementaux.

La Consommation Énergétique : Un Défi de Taille

La consommation énergétique de l'IA, souvent sous-estimée, est une problématique majeure. La majorité des études réalisées sur la consommation énergétique de l'IA se concentrent sur la phase d'entraînement [VSC23], durant laquelle les algorithmes fonctionnent pendant des périodes allant de plusieurs jours à plusieurs semaines pour assimiler des règles statistiques à partir d'un important jeux de données. L'étude réalisée en juin 2019 et portée par des chercheurs de l'Université du Massachusetts, [SGM20] a révélé que l'entraînement de certains modèles d'IA populaires peut produire environ 284 000 kg de dioxyde de carbone, équivalent à près de 300 vols aller-retour entre New York et San Francisco, Figure 2.3 adaptée de [SGM20].

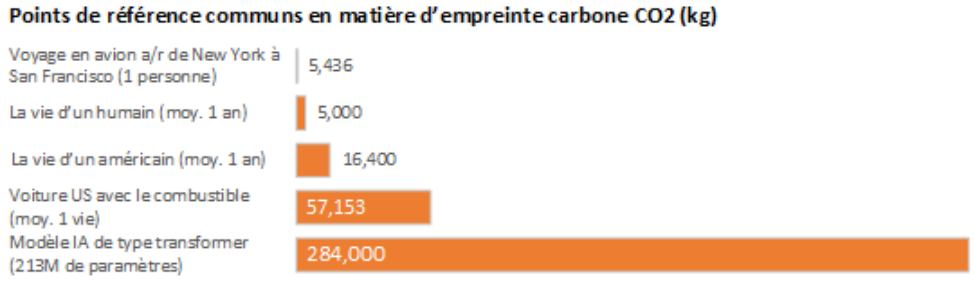

FIGURE 2.3 – Points de référence communs en matière d'empreinte carbone [SGM20]

Elle portait sur la phase d'entraînement de quatre modèles de langage : BERT, conçu par Google en 2018, GPT-2 (OpenAI, 2019), Transformer (Google Brain, 2017) et ELMo (Allen Institute for AI, 2018). Cette étude a été complétée par une autre réalisée par trois chercheurs au sein du département informatique de l'Université de Copenhague en 2020, avec la conclusion qu'une phase d'entraînement pour le modèle ChatGPT-3 nécessite autant d'énergie électrique que le besoin de 126 maisons danoises pendant un an qui se traduit en l'équivalent de CO_2 emis pour parcourir 700 000 km avec un automobile clasique [AKS20].

De plus, les déchets électroniques générés par la technologie d'IA posent un défi environnemental sérieux, car ils contiennent des produits chimiques dangereux qui peuvent contaminer les sols et les ressources en eau [RDK+19], [RDK+23].
L'enjeu est de taille car la multiplication des usages et applications, toujours plus énergivores, risque encore de faire augmenter la consommation énergétique.

Bien que l'IA présente un potentiel incontestable pour contrer les problématiques environnementales, sa propre empreinte écologique ne doit pas être négligée. Elle doit incarner un mouvement global, prônant une transformation profonde et durable à travers le cycle de vie des technologies d'IA, et intégrant les principes éthiques, de transparence, d'inclusion, de protection de la vie privée, de durabilité environnementale, de responsabilité sociale, de sécurité, d'interopérabilité et de gouvernance collaborative.

Jusqu'à là nous avons abordé les préoccupations majeures liées à la consommation énergétique de l'IA et aux défis environnementaux qu'elle pose. Dans notre quête d'une IA Numérique Responsable, nous nous engageons sur un chemin marqué par l'évolution de la conscience collective. Pour mieux comprendre comment nous en sommes arrivés là et comment nous pouvons tracer la voie vers un avenir plus responsable, il est important d'examiner l'histoire de l'IA Numérique Responsable, depuis ses modestes débuts jusqu'à son rôle aujourd'hui. Dans ce contexte, nous explorerons son évolution, en commençant par les contributions au domaine de la cryptanalyse et de l'informatique, comme le travail d'Alan Turing [2] sur le décryptage de la machine Enigma pendant la Seconde Guerre mondiale, jusqu'aux avancées contemporaines en intelligence artificielle.

2.2 Histoire et évolution de l'IA vers une IA Numérique Responsable

L'histoire et l'évolution de l'IA Numérique Responsable s'inscrit dans le contexte plus large du développement de l'IA et de la prise de conscience croissante des impacts envi-

2. Alan Turing mathématicien et théoricien de l'informatique britannique. Il est largement reconnu comme l'un des pères de l'informatique théorique et de l'intelligence artificielle. Turing est célèbre pour son rôle dans le déchiffrement des codes nazis pendant la Seconde Guerre mondiale, notamment le code de la machine Enigma.

ronnementaux et sociaux de la technologie. La Figure 2.4 représente le chemin parcouru jusqu'à aujourd'hui pour aller vers une IA de plus en plus responsable et durable.

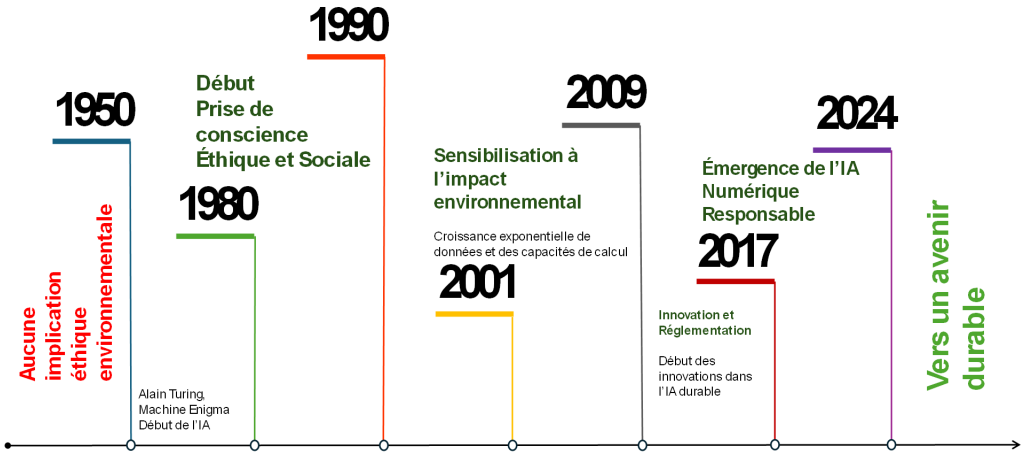

FIGURE 2.4 – Frise chronologique IA Numérique Responsable

Elle commence dans les années 1950, avec les inovations des chercheurs comme Alan Turing, John McCarthy [3] et Marvin Lee Minsky [4] explorant les concepts de machine pensante et d'intelligence artificielle. À cette époque, l'accent était mis sur la création de machines capables de simuler l'intelligence humaine, sans une attention particulière pour les implications éthiques ou environnementales, comme le montre la Figure 2.5 [5].

3. John McCarthy est le principal pionnier de l'intelligence artificielle avec Marvin Lee Minsky. À la fin des années 1950 il a créé avec Fernando Corbató la technique du temps partagé, qui permet à plusieurs utilisateurs d'employer simultanément un même ordinateur. Il reçoit le prix Turing en 1971 pour ses travaux en intelligence artificielle.

4. Marvin Lee Minsky est un scientifique américain. Il a travaillé dans le domaine des sciences cognitives et de l'intelligence artificielle. Il est également cofondateur, avec l'informaticien John McCarthy, du Groupe d'intelligence artificielle du Massachusetts Institute of Technology (MIT) et auteur de nombreuses publications aussi bien en intelligence artificielle qu'en philosophie comme La Société de l'esprit (1986). Son dernier ouvrage, The Emotion Machine (2006), non publié en français, propose de nouveaux développements sur ces théories.

5. Scopus : base de données transdisciplinaire de résumés et de citations de publications scientifiques lancée par l'éditeur scientifique Elsevier en 2004. Scopus référence environ 25 000 journaux scientifiques (y compris 1 200 titres en open access), et intègre chaque année près de 3 millions de nouvelles références : articles scientifiques, publications industrielles, collections d'ouvrages, actes de conférence.

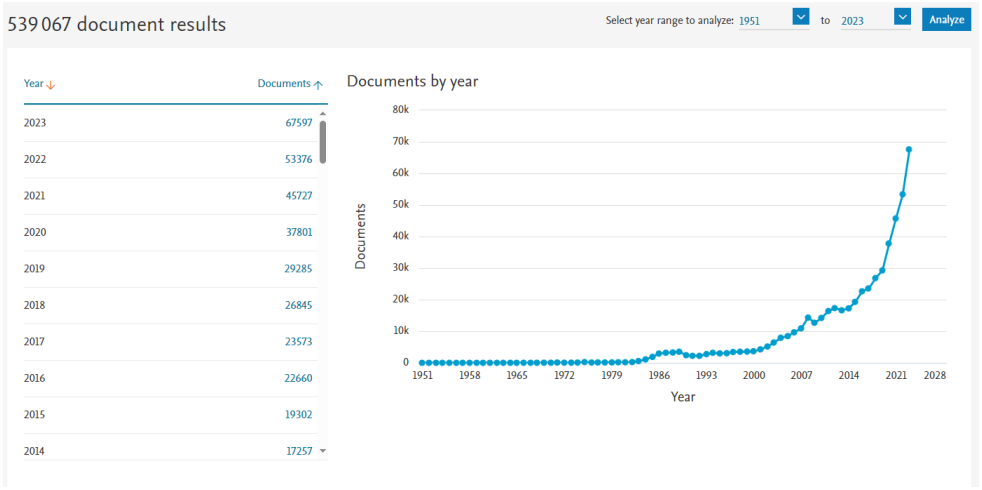

FIGURE 2.5 – Évolutions des publications scientifique dans le domaine de l'IA [Source Scope]

Dans les années 1960-1970, des progrès sont réalisés dans les domaines de la logique symbolique, de la compréhension du langage naturel et des systèmes experts, cette période est aussi appelée **l'âge de l'optimisme**. Parallèlement à ces avancées technologiques, les années 1970 ont également été marquées par une prise de conscience croissante des problèmes environnementaux. Des événements tels que le premier **Jour de la Terre** [6] en 1970 ont mobilisé des millions de personnes à travers le monde pour sensibiliser aux défis environnementaux et promouvoir des actions visant à protéger la planète. Une progression de ce mouvement au fil du temps est affichée dans la Figure 2.6 [7]. Cette sensibilisation accrue à l'égard de l'environnement a influencé les discours publics et a contribué à l'émergence de préoccupations environnementales dans divers secteurs, y compris celui de la technologie.

6. Le Jour de la Terre est une importante célébration environnementale par la société civile. Célébré le 22 avril, le Jour de la Terre est un événement annuel mondial où plusieurs manifestations qui soutiennent la protection de l'environnement sont effectuées et coordonnées grâce au Earth Day Network. Le Jour de la Terre marque tous les ans l'anniversaire de la naissance du mouvement environnemental le plus important de la planète.

7. Source : Jour de la Terre

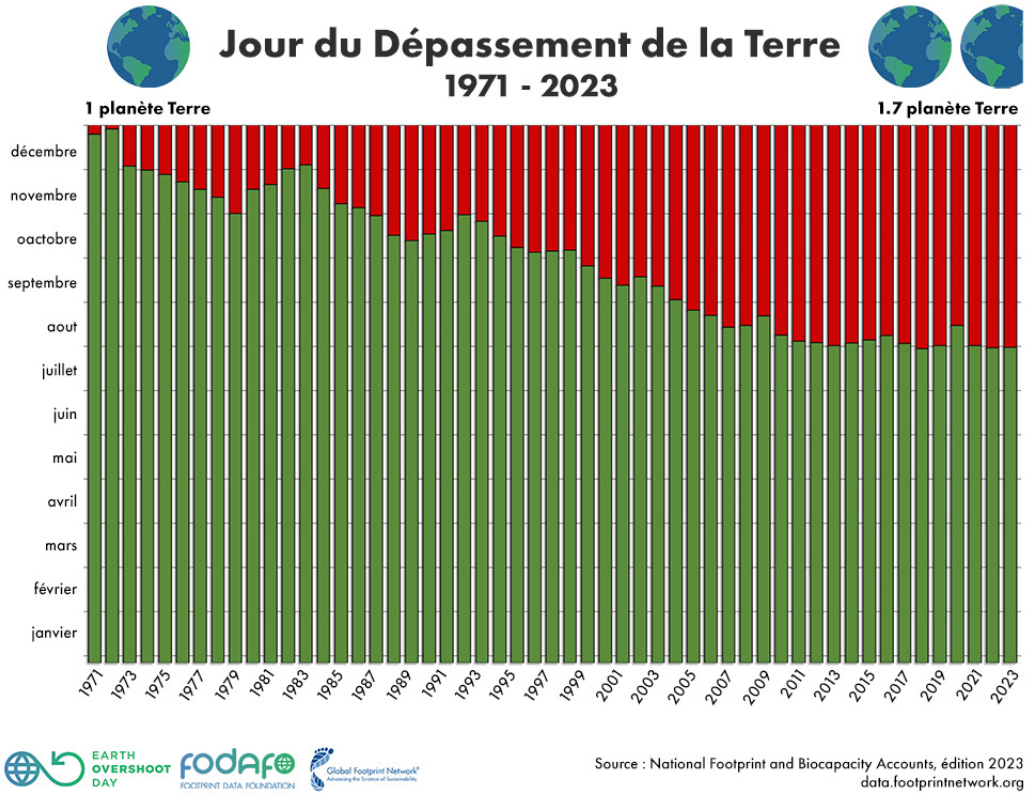

FIGURE 2.6 – Progression du Jour du Dépassement de la Terre au fil des années [Source overshootday.org]

Il suit une période de scepticisme et de financement réduit en raison de la réalisation que les attentes initiales de progrès rapides en IA étaient trop optimistes. Les progrès sont plus lents que prévu, ce qui conduit à une baisse d'intérêt et de financement pour l'IA, surnommée **l'hiver de l'IA** qui s'étale entre 1970 et 1980. Pendant cette période, alors que l'IA traversait une phase de stagnation, l'attention mondiale se tournait de plus en plus vers les préoccupations environnementales. Les années 1980 ont été marquées par une sensibilisation croissante aux questions telles que la déforestation, la pollution atmosphérique et la conservation de la biodiversité. Des événements tels que la publication

du Rapport Brundtland en 1987[8],[9], qui a popularisé le concept de développement durable, ont contribué à sensibiliser davantage le public aux enjeux environnementaux et à encourager les gouvernements et les entreprises à prendre des mesures pour protéger la planète. Cette prise de conscience environnementale a continué à influencer les politiques et les comportements tout au long des années 1980, jetant les bases pour des initiatives plus vastes en faveur de la durabilité environnementale dans les décennies suivantes.

Les années 1980 arrivent avec la renaissance de l'IA du fait de l'émergence de nouvelles approches telles que les réseaux neuronaux et les algorithmes d'apprentissage automatique, l'utilisation croissante des ordinateurs et des algorithmes pour des tâches d'IA, comme la reconnaissance de la parole et la vision par ordinateur, et le développement des systèmes experts avec une adoption industrielle dans des domaines comme la médecine et la finance. Cette période est marquée par les travaux de Yann LeCun[10] sur les réseaux de neurones convolutifs (CNN), une architecture fondamentale dans le domaine de la vision par ordinateur. Ses travaux ont eu un impact majeur sur des applications telles que la reconnaissance d'images, la classification d'objets et la segmentation sémantique. Parallèlement à ces avancées technologiques, ces années ont été marquées par une prise de conscience croissante encore plus forte des problèmes environnementaux. Des événements tels que la Conférence des Nations Unies sur l'environnement et le développement en 1992, également connue sous le nom de Sommet de la Terre de Rio[11], ont mis en lumière l'importance de la durabilité environnementale à l'échelle mondiale. Cette sensibilisation accrue aux enjeux environnementaux a influencé les discours publics et les politiques, incitant de nombreux secteurs, y compris celui de la technologie, à prendre en compte les implications environnementales de leurs activités.

A partir des années 2000, avec l'explosion de la puissance de calcul et la disponibi-

8. Le Rapport Brundtland est le nom communément donné à une publication, officiellement intitulée Notre avenir à tous (Our Common Future), rédigée en 1987 par la Commission mondiale sur l'environnement et le développement de l'Organisation des Nations Unies, présidée par la Norvégienne Gro Harlem Brundtland. Utilisé comme base au Sommet de la Terre de 1992, ce rapport utilise pour la première fois l'expression de « sustainable development », traduit en français par « développement durable », et il lui donne une définition : « Le développement durable est un mode de développement qui répond aux besoins des générations présentes sans compromettre la capacité des générations futures de répondre aux leurs. Deux concepts sont inhérents à cette notion : le concept de « besoins », et spécialement des besoins essentiels des plus démunis, à qui il convient d'accorder la plus grande priorité, et l'idée des limitations que l'état de nos techniques et de notre organisation sociale impose sur la capacité de l'environnement à répondre aux besoins actuels et à venir. »

9. Rapport Brundtland sur l'environnement.

10. Yann LeCun est un chercheur en informatique franco-américain, considéré comme l'un des pères fondateurs de l'apprentissage profond, reconnu pour ses travaux sur les réseaux de neurones convolutifs (CNN), une architecture qui a révolutionné le domaine de la vision par ordinateur.

11. La conférence des Nations unies sur l'environnement et le développement, plus connue sous le nom de sommet de la Terre de Rio de Janeiro ou sommet de Rio, s'est tenue à Rio de Janeiro au Brésil du 3 au 14 juin 1992, réunissant 120 chefs d'État et de gouvernements et 189 pays. Maurice Strong en était le secrétaire général. Environ 2 400 représentants d'organisations non gouvernementales (ONG) étaient présents, tandis que plus de 17 000 personnes assistaient au Forum des ONG qui se tenait parallèlement au sommet.

lité de grandes quantités de données alimentent l'avènement de l'apprentissage profond. Grâce à ces nouveaux moyens de calcul, les réseaux de neurones profonds deviennent capables de performances impressionnantes dans des domaines tels que la reconnaissance d'images et la traduction automatique, évolutions grandement facilitées par les travaux de Yoshua Bengio [12] qui est considéré comme le pionner de l'apprentissage profond et l'un des principaux chercheurs à avoir contribué à sa renaissance. En parallèle à ces développements technologiques, les années 2000 ont été marquées par une prise de conscience accrue de la nécessité de protéger l'environnement. Des initiatives telles que le Protocole de Kyoto en 2005 [13] ont mis en évidence l'urgence d'agir contre le changement climatique et ont incité de nombreux secteurs à réévaluer leurs pratiques en matière de durabilité. Cette prise de conscience environnementale a également influencé le domaine de la technologie, incitant à rechercher des approches plus respectueuses de l'environnement dans le développement et l'utilisation des nouvelles technologies, y compris l'intelligence artificielle.

Cet essor s'accompagne d'un impact environnemental significatif lié aux technologies numériques, y compris celui de l'intelligence artificielle, et l'impact se manifeste à plusieurs niveaux :

Consommation énergétique : Les Data Centers jouent un rôle central dans l'infrastructure technologique moderne en hébergeant les données et les serveurs de calcul nécessaires au fonctionnement des applications de l'IA et d'autres services en ligne. Cependant, ils sont également responsables d'une consommation massive d'énergie et d'une empreinte environnementale significative, principalement en raison de la nécessité de maintenir des conditions de température adéquates pour le bon fonctionnement de l'équipement. Bien que des progrès aient été réalisés pour les rendre plus efficaces sur le plan énergétique, leur consommation d'énergie élevée et leur impact environnemental persistent en raison de la demande croissante de services numériques, y compris ceux basés sur l'IA.

Émissions de CO_2 : La consommation d'énergie par les technologies numériques se traduit par des émissions de dioxyde de carbone, contribuant ainsi au réchauffement

12. Yoshua Bengio, né à Paris en France, est un chercheur québécois d'origine marocaine, spécialiste en intelligence artificielle, et pionnier de l'apprentissage profond. Depuis 2020, il est professeur au département d'informatique et de recherche opérationnelle de l'Université de Montréal. Il est le fondateur et directeur scientifique de Mila. Il est récipiendaire du Prix Acfas Urgel-Archambault 2009, du prix Turing 2018, membre de l'Ordre du Canada et de la Royal Society (depuis 2020).

13. Le protocole de Kyoto est un accord international visant à la réduction des émissions de gaz à effet de serre et qui vient s'ajouter à la Convention-cadre des Nations unies sur les changements climatiques dont les pays participants se rencontrent une fois par an depuis 1995. Signé le 11 décembre 1997 lors de la troisième conférence des parties à la convention (COP 3) à Kyoto, au Japon, il est entré en vigueur le 16 février 2005 « au quatre-vingt dixième jour après la date à laquelle au moins 55 parties à la Convention, incluant les parties « Annexe I » qui comptaient en 1990 un total d'au moins 55 % des émissions de CO_2 de ce groupe, avaient déposé leurs instruments de ratification, d'acceptation, d'approbation ou d'accession ». « Au 14 janvier 2009, 184 États avaient déposé leurs instruments de ratification, d'accession, d'approbation ou d'acceptation »

climatique. Les calculs complexes nécessaires pour l'entraînement des modèles d'IA, en particulier les réseaux de neurones profonds, exigent d'importantes ressources informatiques qui, selon la source d'énergie utilisée, peuvent entraîner des émissions significatives de CO_2.

Utilisation des ressources : La fabrication des composants électroniques nécessaires aux infrastructures numériques (serveurs, disques durs, circuits intégrés) consomme des ressources non renouvelables et rares, telles que les métaux précieux et les terres rares. L'extraction et le traitement de ces matériaux ont des conséquences environnementales, y compris la dégradation des écosystèmes et la pollution.

Déchets électroniques : Le cycle de vie relativement court des équipements informatiques contribue à une accumulation rapide de déchets électroniques, ou e-déchets, qui posent des défis en termes de recyclage et de gestion des déchets. Les substances toxiques contenues dans les e-déchets peuvent entraîner des problèmes de santé publique et de contamination environnementale.

Face à ces défis, des efforts sont déployés pour réduire l'impact environnemental des technologies numériques et de l'IA. Ces efforts incluent :

— le développement de méthodes de calcul plus efficaces,
— l'amélioration de l'efficacité énergétique des centres de données,
— l'utilisation accrue des énergies renouvelables,
— le recyclage des composants électroniques et
— la conception de modèles d'IA moins gourmands en ressources.

En réponse à ces préoccupations, le concept d'IA Numérique Responsable a commencé à émerger de manière significative, principalement à mesure que les préoccupations concernant les implications éthiques, sociales et environnementales de l'intelligence artificielle ont gagné en importance.

Bien que le terme puisse sembler récent, l'idée d'une technologie associée à la responsabilité trouve ses racines dans les années 1970 et 1980, période où les préoccupations environnementales ont commencé à émerger [HE75] . À cette époque, les questions liées à la surconsommation énergétique des data centers et à l'obsolescence programmée des équipements électroniques ont commencé à susciter des inquiétudes, stimulant ainsi les premières réflexions sur une utilisation plus raisonnable et éthique du numérique.

Aujourd'hui, alors que la technologie est omniprésente dans notre vie quotidienne, le concept de numérique responsable revêt une importance capitale. Il souligne la nécessité urgente de repenser nos pratiques technologiques pour répondre aux défis contemporains

tout en préservant l'environnement et en promouvant le bien-être de chacun. Dernièrement, les innovations dans l'IA durable, telles que les algorithmes économes en énergie et les data centers écologiques, ont commencé à se développer. Parallèlement, le gouvernement et les organismes internationaux ont commencé à mettre en place des réglementations et des normes pour encourager une approche plus responsable de l'IA, comme par exemple :

La norme ISO [14] **CEI 30134-2 de 2016** établit le Power Usage Effectiveness (PUE) comme l'indicateur de référence pour évaluer l'efficacité énergétique des centres de données informatiques à l'échelle mondiale. Ce standard a bénéficié de la contribution des professionnels français du secteur, coordonnés par l'AFNOR. [15].

Le Règlement Général sur la Protection des Données (RGPD) : adopté par l'Union Européenne en 2016 et entré en vigueur en 2018, vise à protéger les données personnelles des individus et à réglementer leur traitement, y compris par les algorithmes d'intelligence artificielle. Il impose des obligations strictes en matière de transparence, de consentement, et de protection des données, afin d'assurer que les données personnelles sont utilisées de manière éthique et responsable.

AI Act [16] : proposée initialement le 21 avril 2021 par la Commission européenne, cette réglementation vise à établir des normes communes pour le développement et l'utilisation de l'IA dans l'UE, en mettant l'accent sur la transparence, la responsabilité, et la sécurité. Elle comprend des exigences telles que l'évaluation de l'impact social et environnemental des systèmes d'IA, ainsi que l'interdiction de certaines applications d'IA jugées risquées.

14. L'Organisation internationale de normalisation (en anglais : International Organization for Standardization), est un organisme de normalisation international composé de représentants d'organisations nationales de normalisation de 167 pays, selon le principe d'un membre par pays. L'ISO est le plus grand organisme de normalisation au monde et demeure une organisation non gouvernementale. Référentiel des normes ISO.

15. Le groupe Afnor est un groupe français de services autour de la normalisation et de la certification1 issu de la fusion des associations Association française de normalisation (Afnor) et Association française pour l'assurance de la qualité (AFAQ) et qui comprend quatre métiers : la normalisation, la certification, l'édition spécialisée et la formation.

16. La réglementation de l'intelligence artificielle est l'élaboration de politiques et de lois du secteur public pour promouvoir et réglementer l'intelligence artificielle (IA). Elle est donc liée à la réglementation des algorithmes. Le paysage réglementaire et politique de l'IA est un problème émergent dans les juridictions du monde entier, y compris dans l'Union européenne (qui dispose d'un pouvoir de réglementation gouvernemental) et dans les organismes supranationaux comme l'IEEE, l'OCDE (qui n'en ont pas) et d'autres. Depuis 2016, une série de lignes directrices en matière d'éthique de l'IA ont été publiées afin de maintenir le contrôle social sur cette technologie. La réglementation est considérée comme nécessaire à la fois pour encourager l'IA et pour gérer les risques associés. Outre la réglementation, les organisations déployant l'IA jouent un rôle central dans la création et le déploiement d'une IA digne de confiance, et dans l'atténuation des risques. La réglementation de l'IA par le biais de mécanismes tels que les commissions d'examen peut également être considérée comme un moyen social d'aborder le problème du contrôle de l'IA.

Selon AI Index de Stanford [17], le nombre annuel de lois liées à l'IA adoptées dans les 127 pays étudiés est passée d'une seule loi en 2016 à 37 lois adoptées en 2022, Figure 2.7.

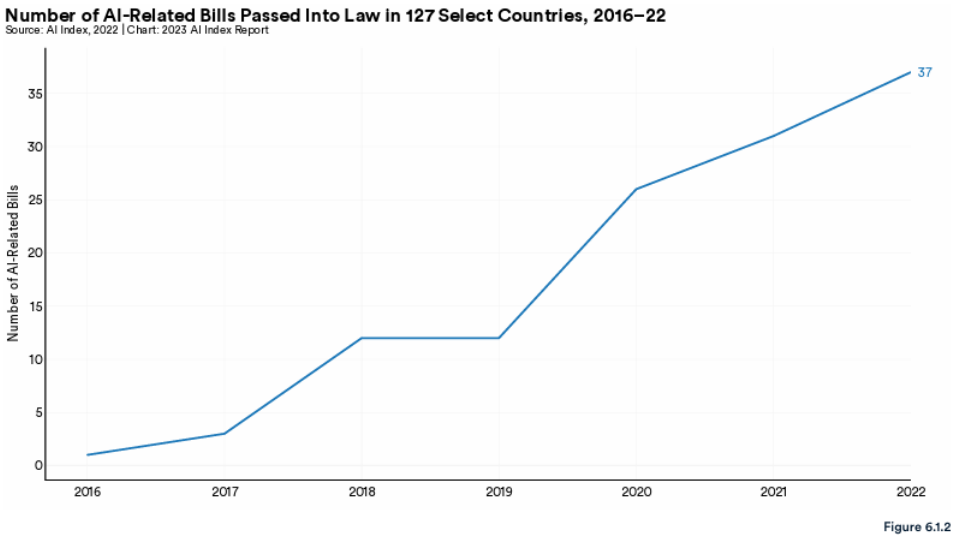

FIGURE 2.7 – Nombre de projets de loi adoptés relatifs à l'IA [Standford AI Index]

De plus en plus, les entreprises intègrent la responsabilité numérique dans leur plan stratégique et dans leurs façons d'utiliser l'IA, reconnaissant que la durabilité est non seulement bénéfique pour l'environnement, mais aussi pour leur image de marque et leur viabilité à long terme.

Cette partie du mémoire a tracé l'évolution de l'IA, depuis ses débuts avec des visionnaires comme Alan Turing jusqu'à l'essor de l'IA Numérique Responsable. Nous avons observé une progression depuis la simple simulation de l'intelligence humaine vers une prise de conscience des impacts éthiques et environnementaux de l'IA.

Aujourd'hui, l'IA Numérique Responsable est en train de devenir un élément central de l'industrie technologique. Cependant, cette nouvelle ère de l'IA ne vient pas sans son lot de défis et d'enjeux complexes. Dans la section suivante, nous explorerons ces défis et enjeux, en examinant les obstacles techniques, éthiques, sociaux et réglementaires qui se dressent sur le chemin de l'IA Numérique Responsable.

17. Standford AI Index

2.3 Enjeux et Défis de l'IA Numérique Responsable

L'intégration de l'IA Numérique Responsable dans les organisations et sociétés contemporaines représente un jalon déterminant dans notre progression vers un avenir technologique éthique et durable. Cependant, cette transition n'est pas sans défis. Cette partie se penche sur les enjeux associés à l'implémentation de l'IA Responsable, allant de l'équilibre entre l'innovation et l'éthique à la gestion de la consommation d'énergie, en passant par les questions d'accessibilité, d'inclusivité, et de cadre politique et réglementaire. Chacun de ces aspects requiert une analyse approfondie pour comprendre comment naviguer ces défis et réaliser le plein potentiel de l'IA dans un cadre responsable.

Équilibre entre Innovation et Éthique : L'un des plus grands défis est de trouver le juste équilibre entre l'innovation rapide dans le domaine de l'IA et le respect des principes éthiques. Cela inclut la protection de la vie privée, la non-discrimination, et la transparence des algorithmes. Cette question est abordée par Deloitte Insights [18], qui souligne l'importance de la transparence et de la protection de la vie privée dans le développement de l'IA, ainsi que la nécessité d'établir des structures de responsabilité appropriées.

Gestion de la Consommation d'Énergie et Durabilité : Les technologies d'IA sont énergivores, notamment les centres de données et les infrastructures informatiques nécessaires pour former et exécuter des modèles d'IA. La réduction de l'empreinte carbone de ces technologies nécessite des investissements dans des sources d'énergie renouvelable, l'optimisation de l'efficacité énergétique et le développement de solutions de refroidissement innovantes. En outre, il y a le défi de concevoir des modèles d'IA qui sont à la fois efficaces et économes en énergie.

Accessibilité et Inclusivité : Assurer que les avantages de l'IA soient accessibles à tous est un autre défi majeur. Cela implique de lutter contre le **fossé numérique**, qui peut exclure certaines populations de l'accès aux technologies d'IA. De plus, il est important que les modèles d'IA soient conçus de manière inclusive, en tenant compte des diverses perspectives et besoins, et en évitant les biais qui peuvent se manifester dans les algorithmes.

Cadre Politique et Réglementaire : Trouver le bon équilibre entre favoriser l'innovation et garantir une utilisation éthique et durable de l'IA est essentiel. Des organisations comme AI Empower soulignent l'importance de la collaboration dans le développement

18. Deloitte Insights est une plateforme qui fournit des perspectives, des analyses et des recherches sur un large éventail de sujets liés aux affaires, à la technologie, à l'économie et à d'autres domaines pertinents pour les entreprises et les décideurs. Leurs publications comprennent des rapports de recherche, des analyses de tendances, des études de cas et des points de vue d'experts, visant à aider les entreprises à comprendre les défis actuels et émergents auxquels elles sont confrontées et à prendre des décisions éclairées.

de directives, de meilleures pratiques et de cadres réglementaires qui favorisent à la fois l'innovation et une utilisation responsable de l'IA. Les gouvernements et les régulateurs ont un rôle dans l'établissement de politiques et de directives pour s'attaquer aux problèmes éthiques liés à l'IA. Par exemple, le Règlement Général sur la Protection des Données (RGPD) de l'Union européenne exige que les organisations puissent expliquer les décisions prises par leurs algorithmes.

L'adoption de cadres législatifs pertinents, couplée à des actions spécifiques au sein des différents secteurs et à une meilleure prise de conscience collective, est nécessaire pour libérer pleinement le potentiel de l'IA Numérique Responsable.

Après avoir exploré les enjeux et les défis liés à l'implémentation de l'IA Numérique Responsable, nous nous concentrons sur la définition précise et les contours de ce concept. La prochaine section, qui conclut cette section dédiée aux Fondements de l'IA Numérique Responsable, se penchera sur la **Définition de l'IA Numérique Responsable**. Cette partie vise à établir une compréhension claire et concise de ce que signifie exactement l'IA Numérique Responsable dans le contexte contemporain.

2.4 Définition de l'IA Numérique Responsable

Pour mieux comprendre la notion de l'Intelligence Artificielle Numérique Responsable, cette section explique les différents principes et pratiques qui sous-tendent ce concept, en mettant l'accent sur son aspect environnemental. En se basant sur des références telles que Deloitte Insights et TechTarget [19], ainsi que sur des outils innovants comme InterpretML, nous analyserons en détail des aspects tels que l'éthique, la transparence, la durabilité environnementale, et l'équité dans le cadre de l'IA. Cette exploration vise à fournir une compréhension holistique de ce que signifie développer et utiliser l'IA de manière responsable dans notre société actuelle, en tenant compte de ses impacts sur l'environnement. La notion d'IA Numérique Responsable repose sur des principes reconnus tels que la sécurité, la fiabilité, l'éthique, la transparence et la réduction des biais. TechTarget souligne l'importance de prendre en compte les aspects éthiques et légaux dans l'IA, mettant en avant le besoin de développer des technologies d'IA qui soient sûres, fiables, éthiquement responsables, et respectueuses de l'environnement.

Sous un angle assez standard, et adopté de plus en plus, l'IA Responsable se concentre sur la création de **systèmes équitables, transparents et écologiquement durables**, capables de prendre des décisions justes et impartiales, en étant compréhensibles pour

19. TechTarget est une entreprise qui fournit des informations spécialisées dans le domaine de la technologie. Elle offre des données, des outils et des services en ligne pour les professionnels de l'informatique et du marketing technologique, avec un accent sur les aspects éthiques et légaux dans l'IA.

les utilisateurs. Ces principes sont importants dans des domaines affectant significativement les écosystèmes et la durabilité. La transparence des systèmes d'IA, devenue aussi possible à travers des outils comme par exemple InterpretML [20], est fondamentale pour expliquer clairement les processus de décision des modèles d'IA, qu'ils soient transparents ou opaques. Par ailleurs, il est important de communiquer sur les performances et les comportements des modèles d'IA, ainsi que sur leurs limites et incertitudes, pour une meilleure efficacité, tout en minimisant leur empreinte environnementale dans la durée.

2.5 Conclusion

Dans ce chapitre dédié aux Fondements de l'IA Numérique Responsable, nous avons exploré les multiples dimensions qui composent le cadre de développement et d'application des technologies d'IA dans une perspective responsable. En mettant en lumière les principes d'éthique, de transparence, d'inclusion, et de durabilité environnementale, nous avons souligné comment ces piliers sont essentiels pour orienter l'IA vers des contributions positives à la société, tout en respectant les droits fondamentaux et en opérant dans le respect de notre environnement. Nous avons examiné les implications de l'IA sur l'écosystème, la nécessité de réduire sa consommation énergétique, et l'importance d'adopter des pratiques de développement durable pour atténuer son impact environnemental.

Ce chapitre a également abordé l'évolution historique de l'IA, soulignant une prise de conscience croissante de ses impacts et la montée d'une approche plus responsable dans son développement. Nous avons évoqué des défis et des enjeux liés à l'intégration de l'IA Numérique Responsable dans la société, notamment l'équilibre entre innovation et éthique, la gestion de la consommation d'énergie, l'accessibilité, et l'élaboration d'un cadre politique et réglementaire adapté. Ce chapitre a posé les bases pour une compréhension approfondie de ce que signifie développer et utiliser l'IA de manière responsable dans notre ère contemporaine, marquant le chemin vers une innovation qui est non seulement technologiquement avancée, mais aussi équitable, sécurisée, et durable pour tous.

Après avoir posé les bases de l'IA Numérique Responsable et examiné ses principaux enjeux et défis, notre attention se porte à présent sur l'état de l'art de l'IA Numérique Responsable. Poursuivant notre exploration, nous aborderons les solutions actuellement disponibles, issues tant de la recherche scientifique que de la mise en application concrète, visant à réduire la consommation énergétique associée à l'IA.

20. Outil open source développé par Microsoft, conçu pour aider à l'interprétation des modèles de Machine Learning.

Synthèse

→ Renforcer l'éthique et la transparence algorithmique pour éliminer les biais et les discriminations, augmentant ainsi la confiance publique dans ces technologies.

→ Adopter des pratiques durables pour atténuer ces impacts et utiliser l'IA dans la lutte contre les problèmes environnementaux.

→ Favoriser la collaboration entre gouvernements, industries, académies et communautés pour le développement d'une IA responsable.

→ Intégrer progressivement des considérations éthiques et écologiques dans son développement.

Chapitre 3

Intelligence Artificielle Numérique Responsable, un État de l'Art

Sommaire

Ce chapitre explore l'IA sous un angle Numérique Responsable. Organisé en six sections, il débute par une exploration du périmètre de l'IA. Il examine ensuite les réseaux de neurones et leur impact environnemental. Une attention spécifique est dédiée au traitement automatique du langage naturel et aux méthodes de compression qui visent à diminuer la consommation de ressources. Les types de processeurs, ainsi que leur contribution à l'efficience énergétique des modèles d'IA, sont analysés, mettant en lumière les forces et faiblesses des architectures matérielles variées. En conclusion, cet état de l'art traite des mesures énergétiques et des indicateurs de performance, essentiels pour évaluer et améliorer l'efficacité énergétique. Nous accentuons l'importance des techniques de compression, notamment durant la phase d'inférence des algorithmes de reconnaissance de texte, soulignant leur rôle dans l'optimisation de l'efficacité énergétique. Nous soulignons la nécessité de mesurer la consommation énergétique lors du déploiement des technologies d'intelligence artificielle.

L'IA gagne progressivement en importance et s'impose de plus en plus dans le monde des entreprises. Les algorithmes de l'IA, capables de reconnaître des images, de classifier des textes et des mails, ou encore de détecter des fraudes, se sont imposés comme des outils incontournables. Cependant, ce n'est que récemment, face à l'urgence climatique, que l'adoption de l'énergie en tant que métrique a commencé à être sérieusement envisagée.

Historiquement, les data scientistes et les entreprises ne se sont pas concentrés sur l'évaluation de la consommation énergétique de leurs algorithmes et applications, souvent par manque de familiarité avec ces approches. Pourtant, la consommation d'énergie a fait l'objet de vastes études dans le domaine de l'architecture informatique depuis plusieurs décennies. Les chercheurs de ce secteur s'intéressent depuis longtemps à la conception de processeurs à haute efficacité énergétique. Le mérite du premier processeur conçu spécifiquement pour réduire la consommation d'énergie est souvent attribué au processeur Intel Atom, qui a été lancé en 2008 [Dom08]. Ce processeur a été développé pour répondre à la demande croissante de dispositifs mobiles tels que les netbooks, les tablettes et les smartphones, qui nécessitaient une faible consommation d'énergie pour prolonger la durée de vie de la batterie. L'architecture du processeur Intel Atom a été conçue avec des caractéristiques visant à minimiser la consommation d'énergie, telles qu'une conception de circuits plus efficace et une réduction de la fréquence d'horloge, permettant ainsi une efficacité énergétique supérieure par rapport aux processeurs conventionnels de l'époque. Dans le domaine des processeurs serveurs, Intel prévoit de segmenter son portefeuille en deux à partir de 2024. L'entreprise proposera des solutions dotées exclusivement de cœurs **haute performance**, désignés sous le nom de P-cores, ainsi que d'autres équipées de cœurs à **haute efficacité énergétique**, nommés E-cores.

Les récentes percées en matière de matériel et de méthodologie dédiées à l'entraînement des réseaux de neurones ont facilité l'émergence d'une nouvelle génération de modèles de grande envergure, entraînés sur des jeux de données massifs. Ces modèles ont nettement accru leur précision dans de multiples domaines du NLP. Toutefois, cette amélioration de la précision est tributaire de ressources computationnelles d'une grande ampleur, ce qui engendre une consommation énergétique significative. Le coût financier lié à l'entraînement de ces modèles est donc élevé, en raison à la fois du matériel requis et du temps de calcul nécessaire sur des infrastructures On Premise [1] ou Cloud. De surcroît, la question de la consommation énergétique associée à l'utilisation de ces ressources est préoccupante.

L'étude [SGM20], une des premières études importantes qui met en évidence la consommation énergétique considérable des modèles NLP, présente une évaluation approximative des coûts financiers et environnementaux engendrés par l'entraînement de divers modèles de réseaux de neurones performants dans le domaine du NLP. À la lu-

1. On-Premise désigne un modèle de licence et d'utilisation pour les logiciels et les programmes informatiques basés sur serveur que le client ou le licencié installe dans son propre environnement informatique.

mière de ces résultats, des recommandations tangibles sont avancées afin de réduire les coûts et de favoriser l'équité dans la recherche et la pratique en NLP et de ce fait faire un premier pas vers une recherche plus durable.

Le cadre pour un reporting systématique de l'empreinte énergétique de l'IA est initié dans les travaux [HHR+20], avec un point d'attention sur l'importance de la transparence dans la recherche en IA pour le climat.

Le travail [SDSE20] publié dans « Communications of the ACM » [2], introduit le concept de « **Green AI** » [3], mettant l'accent sur la recherche en IA qui priorise l'efficacité énergétique et la réduction de l'empreinte carbone, par opposition à la « **Red AI** » [4] qui se concentre uniquement sur la performance sans tenir compte de l'impact environnemental. Une vision encore plus large est la « **AI for Green** », vision qui met l'accent sur l'utilisation de l'IA pour résoudre les problèmes environnementaux et promouvoir le développement durable. Cette approche utilise l'IA pour analyser les données environnementales, modéliser les systèmes écologiques, optimiser les processus industriels pour réduire les déchets et les émissions, et faciliter la transition vers une économie plus verte. Ce concept englobe à la fois le développement de technologies d'IA vertes et l'application de l'IA pour relever les défis environnementaux et contribuer à la préservation de la planète [YMC21], [Lin21], [BLZ+22]. Avec une vision plus sociale, James Hodson [Hod15], pense qu'il est difficile de comprendre les impacts de l'IA sur la société, et les pressions continues de l'automatisation nécessiteront des adaptations sociétales et des efforts politiques dans les années à venir. En 2014, il est à l'origine de l'initiative « **AI for Good** », avec Stefano Pacifico et Michael Witbrock.

Une autre démarche est celle qui consiste à quantifier les émissions de carbone CO_2 associées à l'entraînement de divers modèles d'apprentissage automatique, travail évoqué dans [LLSD19] offrant un aperçu concret de l'impact environnemental de l'IA. Toujours dans une même approche, dans [CHSV17], les auteurs explorent des techniques de quantification pour réduire la consommation d'énergie des modèles d'apprentissage profond

2. Communications of the ACM (CACM) est la principale revue mensuelle de l'Association for Computing Machinery (ACM). Créé en 1957, CACM est envoyé à tous les membres de l'ACM, environ 80 000 actuellement. Les articles sont écrits pour des lecteurs ayant des connaissances dans tous les domaines de l'informatique et des systèmes d'information. L'accent est mis sur les applications concrètes des découvertes sur les technologies de l'information et sur les problèmes de gestion des systèmes d'information. L'ACM publie aussi de nombreuses autres revues plus théoriques.

3. « Green AI » est une approche de l'IA qui intègre des considérations environnementales et cherche à réduire l'empreinte carbone et l'utilisation des ressources. Cette approche vise à développer des modèles d'IA plus efficaces en termes d'énergie, en optimisant les algorithmes, les architectures de modèle et les méthodes d'entraînement pour minimiser la consommation de ressources. Les initiatives de « Green AI » encouragent également l'utilisation de sources d'énergie renouvelables et la mise en place de pratiques de développement durable dans le domaine de l'IA.

4. Le concept de « Red AI » se réfère à une approche de l'IA qui met l'accent sur l'obtention de performances et de résultats impressionnants, souvent au détriment de l'efficacité énergétique et environnementale. Le terme « RED » vient de l'idée de « redlining », évoquant une utilisation excessive ou abusive des ressources, ce qui peut avoir des conséquences néfastes sur l'environnement.

sans compromettre significativement leur précision.

Malgré une émergence de conscience dans le domaine de l'apprentissage automatique, la plupart des efforts de recherche demeurent concentrés sur l'optimisation de la précision des modèles, souvent sans prendre en compte les contraintes computationnelles. Cette tendance est marquée dans l'apprentissage profond, où l'objectif est de développer des modèles toujours plus profonds et précis, sans restriction en termes de ressources de calcul. Ces modèles, souvent gourmands en calcul (mesurés en gigaFLOPS [5]) et en mémoire (avec des millions de paramètres ou de poids), requièrent une importante puissance de calcul pendant la phase d'entraînement et peuvent être utilisés à de multiples reprises lors du déploiement. Néanmoins, une certaine prise de conscience concernant la consommation énergétique commence à émerger au sein de la communauté de recherche en apprentissage automatique, comme d'autres branches d'ailleurs. Afin de relever cette problématique, différentes approches sont actuellement à l'étude pour estimer la consommation d'énergie, tant à un niveau global de l'IA que pour les applications spécifiques de l'apprentissage automatique. Ces initiatives de recherche s'efforcent de concevoir des approches qui conjuguent progrès technologiques et préservation environnementale.

Pour approfondir notre compréhension de l'état de l'art de l'IA Numérique Responsable, il est évident de commencer par définir les périmètres de l'IA, du Machine Learning et du Deep Learning. Cette démarche nous permet d'identifier précisément où et pourquoi les algorithmes consomment d'importantes ressources aujourd'hui. En clarifiant ces concepts, nous pourrons mieux cerner les domaines d'action et les raisons sous-jacentes à la consommation énergétique élevée des technologies d'IA, jetant ainsi les bases pour explorer les solutions visant à réduire leur empreinte écologique.

3.1 Intelligence Artificielle en quelques mots

L'IA est un domaine interdisciplinaire de l'informatique qui se concentre sur la création et le déploiement de systèmes informatiques capables d'effectuer des tâches qui nécessitent normalement des processus cognitifs humains, tels que la perception, la compréhension du langage, la résolution de problèmes, la prise de décision et l'apprentissage à partir de l'expérience. Comme le souligne Stuart Russell et Peter Norvig dans leur ouvrage de référence, [RN10], [RN21] (édition actualisée en 2021), l'IA est conçue pour imiter et même surpasser les capacités humaines en termes de raisonnement et d'analyse.

5. FLOPS : Le nombre d'opérations en virgule flottante par seconde (en anglais : floating-point operations per second) est une unité de mesure de la rapidité de calcul d'un système informatique et donc d'une partie de sa performance. Les opérations en virgule flottante (additions ou multiplications) sont des opérations qui permettent des calculs représentant de très grands et de très petits nombres représentés par une mantisse et un exposant. De telles opérations prennent plus de temps de calcul que des opérations sur les nombres entiers et sont utilisées dans certains types d'applications.

L'IA repose sur des algorithmes complexes, souvent inspirés de modèles mathématiques et statistiques, ainsi que sur l'utilisation de vastes ensembles de données pour entraîner et améliorer les performances des systèmes. Les principaux domaines de l'IA incluent l'apprentissage automatique, le traitement du langage naturel, la vision par ordinateur, la robotique et la recherche opérationnelle. Nous les représentons sous forme de diagramme d'Euler en Figure 3.1, convention largement présente dans la literature.

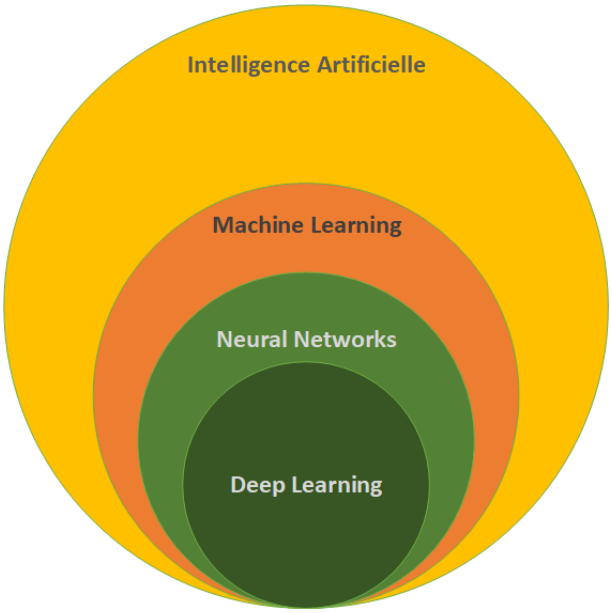

FIGURE 3.1 – Diagramme d'Euler des domaines de l'Intelligence Artificielle [SCYE17]

Le **Machine Learning** (ML) se distingue comme une branche importante de l'IA. Le ML est défini par Tom M. Mitchell [6] dans [Mit97], une référence dans ce domaine, comme la capacité d'un système à apprendre et à s'améliorer à partir de l'expérience sans être explicitement programmé pour cela. Le ML englobe diverses techniques et algorithmes, allant de l'apprentissage supervisé à l'apprentissage non supervisé, en passant par l'apprentissage par renforcement. Des applications typiques du ML incluent les systèmes de recommandation, comme ceux utilisés par Netflix ou Amazon, pour suggérer des produits ou des films en fonction des préférences des utilisateurs.

Entre le ML et le **Deep Learning (DL)**, le concept de **réseaux de neurones**

6. Tom Michael Mitchell, né le 9 août 1951 à Blossburg en Pennsylvanie, est un informaticien américain et professeur à l'Université de Carnegie Mellon (CMU). Mitchell est connu pour ses contributions à l'avancement de l'apprentissage automatique, l'intelligence artificielle et les neurosciences cognitives et est l'auteur du manuel Machine Learning. Il est membre de la Académie nationale d'ingénierie des États-Unis depuis 2010. Il est également membre de l'Association Américaine pour l'Avancement des Sciences et de l'Association pour le Progrès de l'Intelligence Artificielle.

(Neural Networks-NN) occupe une place fondamentale. Les réseaux de neurones sont des modèles inspirés par le fonctionnement du cerveau humain, composés d'unités de calcul, ou neurones, organisés en couches.

Ces modèles sont capables d'apprendre des représentations de données à travers une méthode appelée "apprentissage par backpropagation", qui ajuste les poids des connexions neuronales en fonction de l'erreur de sortie du réseau.

David E. Rumelhart, Geoffrey E. Hinton, et Ronald J. Williams, ont été les pionniers dans la formalisation de l'algorithme de **rétropropagation du gradient**, à travers les travaux [RHW86] publiés dans leur article fondateur en 1986, posant ainsi les bases du succès des réseaux de neurones dans les applications d'IA. Avant l'introduction de la rétropropagation, l'ajustement des poids dans les réseaux de neurones était largement inefficace et imprécis, limitant la complexité des problèmes que ces réseaux pouvaient résoudre. Cet algorithme a apporté une méthode efficace pour l'entraînement des réseaux de neurones profonds, en permettant de calculer le gradient de la fonction de perte par rapport à tous les poids du réseau. Cette avancée a permis de développer des réseaux de neurones capables de traiter des tâches de classification, de régression et de reconnaissance de motifs avec une efficacité sans précédent.

L'introduction de l'algorithme de rétropropagation du gradient a fourni une méthode efficace pour l'entraînement des **réseaux de neurones profonds (DNN)**, permettant à ces architectures complexes d'apprendre à partir de **grandes quantités de données**.

Les DNNs sont des architectures d'apprentissage en couches qui imitent le fonctionnement du cerveau humain pour traiter les données et qui permets de créer des modèles prédictifs. Elles sont caractérisées par leur architecture multicouche comportant un grand nombre de couches de neurones interconnectées. Ces réseaux sont appelés "profonds" en raison de leur profondeur, c'est-à-dire du nombre élevé de couches entre l'entrée et la sortie du réseau.

Ian Goodfellow, Yoshua Bengio et Aaron Courville, dans [GBC16], décrivent comment le DL a révolutionné des domaines tels que la vision par ordinateur et le traitement automatique du langage naturel. Un exemple marquant de DL reste le système AlphaGo [7] de DeepMind [8], qui a battu plusieurs champions du monde du jeu de go [9], un jeu qui a longtemps été considéré comme un défi majeur pour l'intelligence artificielle en raison de sa complexité extrêmement élevée.

Après avoir exploré quelques notions sur les bases de l'intelligence artificielle, notamment du Deep Learning, du Machine Learning et des réseaux de neurones, nous

7. AlphaGo
8. Google DeepMind
9. Histoire du go

abordons maintenant les aspects fondamentaux des processus d'entraînement et d'inférence qui rendent ces technologies opérationnelles. Notre attention se concentre sur les méthodes de classification et de régression qui sont la base de nombreuses applications d'intelligence artificielle. Nous examinerons également les fonctions de perte qui guident l'optimisation des poids du réseau pendant l'entraînement.

Nous explorerons les techniques d'entraînement, notamment l'apprentissage supervisé et non supervisé, qui permettent respectivement aux modèles d'apprendre à partir de données étiquetées et non étiquetées. De plus, nous examinerons les stratégies d'optimisation telles que le Momentum et Adam, qui améliorent la capacité des modèles à ajuster leurs paramètres pour minimiser la fonction de perte, accélérant ainsi le processus d'entraînement et améliorant leur capacité à généraliser à partir des données (il s'agit de la capacité du modèle à appliquer ce qu'il a appris à partir de son ensemble d'entraînement à des situations qu'il n'a pas explicitement rencontrées pendant sa phase d'entraînement).

Nous discuterons également de l'importance du batching [10] dans l'efficacité de l'entraînement, en permettant le traitement des données par groupes pour une utilisation plus efficace des ressources de calcul.

3.2 Entraînement et Inférence

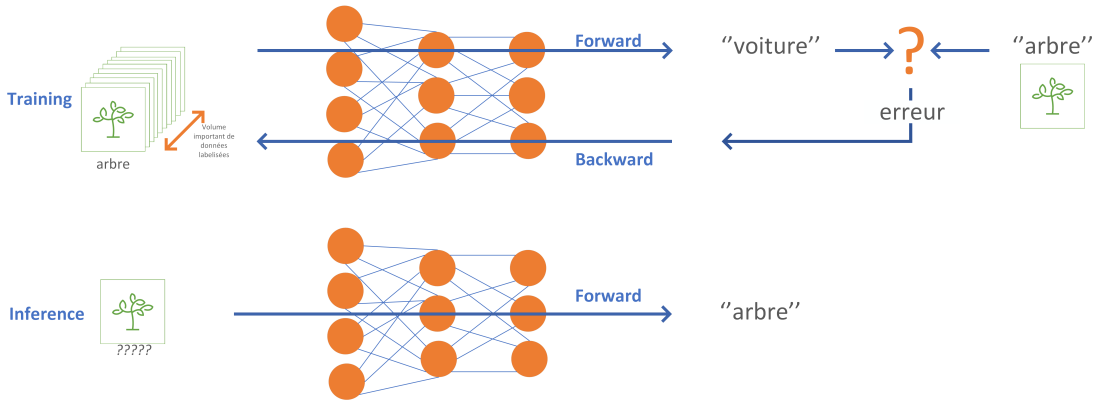

FIGURE 3.2 – Les étapes de l'entraînement et de l'inférence (figure inspiré de NVIDIA)

10. batching, dans le contexte de l'apprentissage automatique et de l'entraînement de modèles d'IA, désigne la pratique de diviser l'ensemble des données d'entraînement en plusieurs petits groupes ou "batches". Au lieu de mettre à jour les poids du modèle après chaque exemple de données (entraînement stochastique) ou après avoir traité l'ensemble complet des données (entraînement par lots), l'entraînement se fait sur ces petits lots.

Cette section explore deux concepts essentiels dans le domaine des réseaux neuronaux : l'apprentissage et l'inférence. Une représentation des grandes étapes est montrée dans la Figure 3.2.

L'Entraînement

L'objectif de l'entraînement des DNNs est de trouver un ensemble de poids W et de biais b qui optimisent la fonction de coût L, minimisant ainsi la différence entre les prédictions du modèle $\hat{y}_i(W, b)$ et les valeurs réelles y_i, tout en maximisant la probabilité d'identifier correctement la classe et minimisant celle d'assigner incorrectement les classes. Ce processus implique la propagation avant (forward propagation) pour calculer les prédictions et la rétropropagation (backward propagation) pour ajuster les paramètres afin de réduire la perte moyenne sur un vaste ensemble d'entraînement, en se basant sur les classes correctes souvent spécifiées dans l'ensemble d'apprentissage [LBH15], [WLCS18].

Il existe plusieurs méthodes pour entraîner les poids des modèles. L'approche la plus courante est appelée **apprentissage supervisé**, où tous les échantillons d'apprentissage sont étiquetés, Figure 3.3.

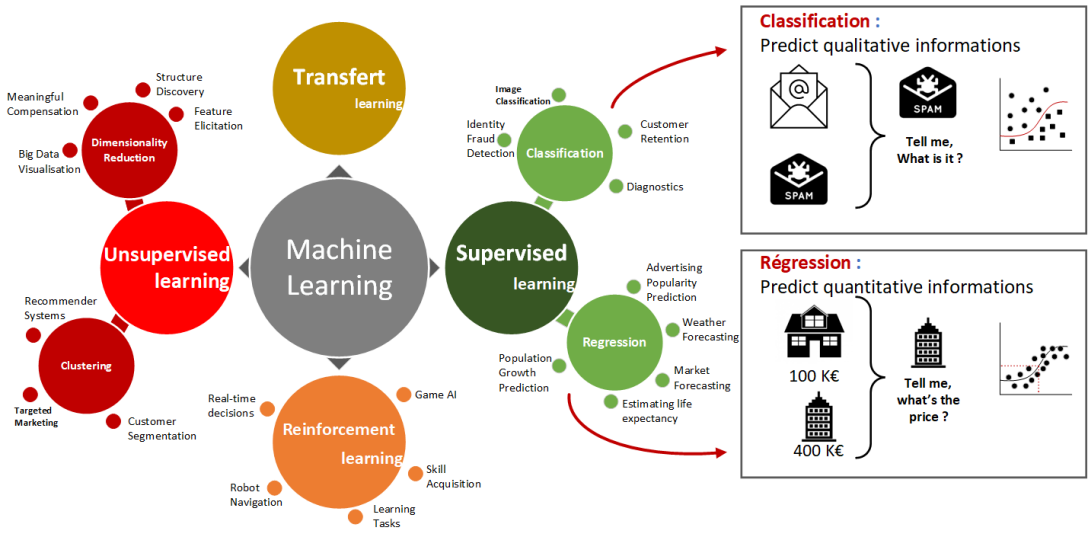

FIGURE 3.3 – Illustration des Algorithmes de Machine Learning [BHV19]

L'**apprentissage non supervisé** est une autre approche, où aucun échantillon d'apprentissage n'est étiqueté. L'**apprentissage semi-supervisé** utilise un ensemble de données étiquetées et non étiquetées (par exemple, utilisez données non étiquetées pour définir les limites du cluster et utiliser la petite quantité de données étiquetées pour étiqueter les clusters), [Zhu05], [Zhu08], [vEH20]. Enfin, l'**apprentissage par renforcement**

[SB17] peut être utilisé pour entraîner les poids de telle sorte qu'étant donné l'état de l'environnement actuel, le DNN peut indiquer l'action que l'agent doit entreprendre pour maximiser les récompenses attendues ; cependant, les récompenses peuvent ne pas être disponibles immédiatement après une action, mais seulement après une série d'actions.

Une autre approche couramment utilisée pour déterminer les poids est le **réglage fin** (fine-tuning), des poids entraînés étant disponibles et sont utilisés comme point de départ, puis ces poids sont ajustés pour un nouvel ensemble de données (l'**apprentissage par transfert**). Cela permet un entraînement plus rapide que de commencer à partir d'un point de départ aléatoire, et peut parfois donner une meilleure précision, [PY10], [TLBH16]. En ajustant les poids du modèle pré-entraîné nous évitons ainsi le coût et le temps associés à l'entraînement d'un modèle à partir de zéro en bénéficiant de l'apprentissage déjà effectué. Cette méthode est utile dans des contextes où les données disponibles pour le nouvel ensemble de tâches sont limitées, car elle permet de tirer parti des motifs et des caractéristiques apprises sur un ensemble de données plus vaste et potentiellement plus général.

L'Inférence

L'inférence fait référence à l'utilisation d'un modèle déjà entraîné pour prédire la sortie basée sur de nouvelles entrées. Pendant l'inférence, les poids et les biais du modèle restent fixes, seul le processus de propagation avant est utilisé pour calculer les prédictions. Dans ce processus, l'entrée est transmise à travers les différentes couches du réseau, chaque couche appliquant ses poids, biais et fonctions d'activation pour transformer les données d'entrée jusqu'à produire une sortie. La sortie peut être une prédiction de classe dans les tâches de classification, une valeur continue dans les cas de régression, ou tout autre type de données spécifiques au problème traité.

L'inférence est importante pour évaluer la performance réelle du modèle dans des conditions réelles d'utilisation, où les données n'ont pas été vues par le modèle durant son entraînement. Elle permet de mesurer la capacité du modèle à généraliser à partir de ses apprentissages à de nouvelles données. La rapidité et l'efficacité du processus d'inférence sont également importantes, surtout dans les applications où le temps de réponse est critique, comme dans les systèmes de recommandation en temps réel, la détection d'objets en vision par ordinateur, ou encore dans les applications de traduction automatique.

Pour une nouvelle entrée x, la prédiction $\hat{y}$ est calculée en utilisant la fonction f, représentant le réseau de neurones, avec les paramètres W et b déjà fixés après l'entraînement. Mathématiquement, cela se présente comme suit :

$$\hat{y} = f(x; W, b) \tag{3.1}$$

où f est la fonction du réseau de neurones, x est la nouvelle entrée, W représente les poids du réseau neuronal, et b représente les biais du réseau neuronal. Ces poids et biais

sont fixés après l'entraînement du modèle sur un ensemble de données d'entraînement. Lors de l'inférence, le modèle utilise ces paramètres fixes pour calculer la prédiction pour la nouvelle entrée x.

Fonctions de Perte en Apprentissage Supervisé : Régression, Classification Binaire et Multiclasse

Dans le cas de la **régression**, où l'on cherche à prédire des valeurs continues, la fonction de coût est souvent formulée comme la somme des carrés des écarts entre les valeurs prédites et réelles, divisée par le nombre total d'exemples dans l'ensemble d'entraînement (*Erreur Quadratique Moyenne (Mean Squared Error - MSE)*) :

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{3.2}$$

où y_i est la valeur réelle et $\hat{y}_i$ est la valeur prédite par le modèle, avec N étant le nombre d'exemples.

Une autre méthode utilisée est *Erreur Absolue Moyenne (MAE)* mesure la moyenne des valeurs absolues des différences entre les prédictions et les valeurs réelles. La formule de MAE est la suivante :

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{3.3}$$

En plus de MSE et MAE, l'*Erreur Quadratique Moyenne Racine (Root Mean Squared Error - RMSE)* est également fréquemment utilisée dans la régression pour mesurer la différence entre les valeurs prédites par un modèle et les valeurs réelles. La RMSE est la racine carrée de la MSE, offrant ainsi une échelle qui est compatible et directement comparable aux valeurs réelles et prédites. La formule de RMSE est la suivante :

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{3.4}$$

La MSE est couramment privilégiée pour sa capacité à souligner les grandes erreurs à travers la mise en carré des écarts, ce qui facilite son intégration avec des méthodes d'optimisation basées sur le gradient. Cependant, cette caractéristique peut aussi rendre la MSE moins intuitive, amplifiant disproportionnellement les grandes erreurs au détriment des petites et masquant ainsi leur impact dans l'évaluation totale. À l'inverse, la MAE offre une robustesse contre les valeurs aberrantes et fournit une compréhension directe en mesurant la moyenne des différences absolues entre prédictions et valeurs réelles, traitant

toutes les erreurs de manière égale sans privilégier les grandes erreurs. La RMSE, prenant la racine carrée de la MSE, conserve la sensibilité de la MSE aux grandes erreurs tout en alignant l'échelle des erreurs à celle des valeurs prédites et réelles, rendant ses résultats plus aisément interprétables et offrant une perspective intuitive sur la magnitude réelle des erreurs.

Après avoir introduit les fonctions de perte telles que MSE, MAE, et RMSE pour évaluer la performance des modèles de régression, il est également essentiel de considérer le coefficient de détermination R^2. Le R^2 mesure la part de la variance des données expliquée par le modèle, offrant une perspective sur la qualité de l'ajustement du modèle aux données observées. Un R^2 proche de 1 indique que le modèle explique une grande part de la variance, tandis qu'un R^2 proche de 0 suggère que le modèle n'explique pas efficacement la variance des données.

La relation entre MAE, RMSE, et la taille de l'échantillon n est exprimée par l'encadrement suivant :

$$\text{MAE} \leq \text{RMSE} \leq \sqrt{n}\text{MAE} \tag{3.5}$$

Cet encadrement illustre comment MAE et RMSE se comparent en termes de magnitude de l'erreur. La première inégalité confirme que RMSE est toujours au moins aussi grande que MAE, reflétant la surpondération des grandes erreurs par RMSE et MSE. La seconde inégalité met en évidence que l'écart entre RMSE et MAE peut augmenter avec la taille de l'échantillon n. Dans le cas extrême où toutes les erreurs proviennent d'une seule observation, RMSE atteint sa valeur maximale de $\sqrt{n}$MAE. Cette dynamique souligne comment la distribution des erreurs affecte la sélection de la métrique d'évaluation la plus appropriée.

Pour la **classification binaire**, où les sorties attendues sont des catégories (par exemple, 0 ou 1), la fonction de coût est généralement exprimée comme la somme négative du logarithme des probabilités prédites, ajustée pour correspondre à la catégorie réelle de chaque exemple. Cette formulation, connue sous le nom de log loss ou *entropie croisée*, pénalise les prédictions incorrectes plus sévèrement que les prédictions proches de la vérité, encourageant ainsi le modèle à produire des probabilités de prédiction précises.

$$\text{BinaryCrossEntropy}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right) \tag{3.6}$$

où y_i est la valeure réelle pour l'exemple i (0 ou 1), $\hat{y}_i$ est la probabilité prédite par le modèle que l'exemple i appartienne à la classe 1, N est le nombre total d'exemples d'entraînement.

Hinge Loss : Cette fonction de perte est souvent utilisée avec les SVM (Support Vector Machines) [11] pour les problèmes de classification binaire. Elle mesure la marge entre les classes et encourage une marge plus large entre les exemples de différentes classes.

$$\text{HingeLoss}(y, \hat{y}) = \max(0, 1 - y \cdot \hat{y}) \tag{3.7}$$

où $\hat{y}$ est la prédiction du modèle, et y est la valeur réelle (1 pour la classe positive et -1 pour la classe négative), favorisant une marge positive entre les scores attribués aux classes positive et négative, avec une marge minimale de 1.

Ces fonctions de perte sont couramment utilisées dans l'apprentissage supervisé pour entraîner des modèles de classification. Le choix entre elles dépend souvent de la nature spécifique de la tâche et des caractéristiques des données. Selon le problème spécifique, d'autres fonctions de perte peuvent également être envisagées, comme la perte de log-vraisemblance négative pour les tâches de régression logistique.

Dans le contexte de la **classification multiclasse**, où un exemple d'entrée peut appartenir à l'une de plusieurs classes distinctes, la fonction de coût nécessite une adaptation pour gérer cette complexité. La classification multicritère doit évaluer la probabilité qu'une entrée appartienne à chacune des classes possibles. Pour cela, la fonction de coût d'*entropie croisée multiclasse* est souvent utilisée. Elle est définie comme suit :

$$\text{CategoricalCrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log(\hat{y}_{ij}) \tag{3.8}$$

où y est le vecteur des étiquettes réelles encodées en one-hot [12], $\hat{y}$ est le vecteur des prédictions de probabilité pour chaque classe, produites par le modèle, N est le nombre total d'exemples d'entraînement, M est le nombre total de classes.

Sparse Categorical Cross-Entropy : Cette fonction de perte est similaire à l'entropie croisée catégorielle, mais elle est utilisée lorsque les étiquettes de classe sont fournies sous forme d'entiers plutôt que de représentations one-hot.

11. classe de modèles d'apprentissage supervisé utilisés pour la classification, la régression et la détection des valeurs aberrantes. Développées dans les années 1990, les SVM sont reconnues pour leur capacité à créer des frontières de décision complexes, même dans des espaces où les données ne sont pas linéairement séparables.

12. L'encodage "one-hot" est une méthode de représentation des étiquettes de classe sous forme de vecteurs binaires où un seul élément est à 1 pour indiquer la classe spécifique de l'exemple, et tous les autres éléments sont à 0, facilitant ainsi la classification précise dans l'apprentissage automatique.

$$\text{SparseCategoricalCrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N}\sum_{i=1}^{N}\log(\hat{y}_{i,y_i}) \tag{3.9}$$

où y est le vecteur des étiquettes de classe réelles (entiers), $\hat{y}$ est le vecteur des prédictions de probabilité produites par le modèle pour chaque classe, N est le nombre total d'exemples d'entraînement. Cette fonction pénalise les écarts entre les probabilités prédites et les vraies étiquettes de classe, en encourageant le modèle à estimer correctement la probabilité de chaque classe pour chaque exemple d'entraînement.

Dans le contexte de classification multicritère, l'optimisation de la fonction de coût nécessite d'ajuster les poids et les biais pour réduire l'erreur totale, tout en veillant à ce que le modèle distingue précisément la classe la plus probable pour chaque entrée, parmi un éventail étendu d'options.

Optimisation des Réseaux de Neurones Profonds : Descente de Gradient Stochastique et Rétropropagation

Pour la minimisation de la fonction de perte [GBC16], l'**algorithme de descente de gradient** est largement utilisé. L'optimisation par descente de gradient stochastique (SGD) et ses variantes, comme Adam, RMSprop, sont des méthodes d'optimisation qui permettent de mettre à jour les poids du réseau de manière itérative en se basant sur le gradient de la fonction de perte.

Dans le cadre de l'apprentissage profond, deux concepts importants dans l'optimisation des modèles de réseaux de neurones sont : la descente de gradient stochastique (SGD) et la rétropropagation.

La descente de gradient stochastique (SGD) permet la mise à jour itérative des paramètres du modèle θ, les ajustant ainsi pour minimiser la fonction de perte $L(\theta)$. Cette mise à jour est réalisée à chaque étape d'itération t selon la formule :

$$\theta_{t+1} = \theta_t - \eta\nabla L(\theta_t)$$

où η représente le taux d'apprentissage, qui contrôle la taille du pas effectué dans la direction opposée au gradient, et $\nabla L(\theta_t)$ est le gradient de la fonction de perte par rapport aux paramètres θ_t, indiquant ainsi la direction de la pente la plus raide.

La rétropropagation est employée pour calculer ces gradients nécessaires à la mise à jour des paramètres via SGD. Ce processus exploite la règle de la chaîne pour propager les erreurs calculées à la sortie du réseau (pendant l'avant-propagation) à travers toutes

les couches intermédiaires. Ainsi, pour chaque poids w_{ij}^l dans le réseau, où l désigne la couche concernée, le changement $\Delta w_{ij}^l = -\eta \frac{\partial E}{\partial w_{ij}^l}$ est appliqué, E représentant l'erreur totale du réseau. Le calcul du gradient $\frac{\partial E}{\partial w_{ij}^l}$ à travers les couches permet d'ajuster les poids pour minimiser l'erreur.

Ce processus de rétropropagation est précédé par la propagation avant, au cours de laquelle les entrées traversent le réseau couche par couche pour générer une sortie. L'erreur entre cette sortie et la sortie attendue est calculée à l'aide de la fonction de coût spécifique, ce qui sert de fondement au calcul des gradients lors de la rétropropagation. En combinant la descente de gradient stochastique et la rétropropagation, il devient possible d'ajuster les paramètres du réseau dans le but de réduire l'erreur de prédiction.

Une variante de SGD est la SGD mini-batch, où les mises à jour des paramètres sont calculées sur un petit sous-ensemble aléatoire des données d'entraînement à chaque itération, ce qui accélère le processus d'optimisation tout en conservant la capacité à généraliser [Bot10].

L'utilisation de mini-batch présente plusieurs avantages :

— Amélioration de l'efficacité du calcul : Calculer le gradient sur de petits mini-batchs de données est plus rapide que de le faire sur l'ensemble complet des données.

— Stabilisation des mises à jour des paramètres : En moyennant les gradients sur un mini-batch, les mises à jour des paramètres sont moins sensibles aux fluctuations bruitées des gradients individuels.

— Utilisation efficace de la mémoire : Les mini-batchs permettent de travailler avec des données en mémoire de manière efficace.

— La taille du mini-batch est un hyperparamètre réglable, et le choix d'une taille appropriée peut avoir un impact significatif sur la vitesse de convergence et la qualité de l'optimisation.

En plus de l'algorithme de descente de gradient stochastique, il existe plusieurs autres méthodes d'optimisation couramment utilisées en apprentissage automatique et en apprentissage profond : Adam, RMSProp, Adagrad, Adadelta, et NAG.

— Adam : est un algorithme d'optimisation qui combine les avantages de l'optimisation par descente de gradient stochastique avec ceux de l'optimisation par momentum. Il adapte le taux d'apprentissage pour chaque paramètre du modèle individuellement, en fonction des estimations du premier et du deuxième moments des gradients [KB15].

— RMSProp (Root Mean Square Propagation) : RMSProp est un algorithme d'optimisation qui ajuste le taux d'apprentissage pour chaque paramètre du modèle en fonction de la moyenne mobile de l'accumulation des gradients au carré [HSS12].

— Adagrad (Adaptive Gradient Algorithm) : Adagrad est un algorithme d'optimisation qui adapte le taux d'apprentissage pour chaque paramètre du modèle en fonction de l'historique des gradients accumulés pour ce paramètre [DHS11].

— Adadelta : Adadelta est une extension d'Adagrad qui vise à résoudre certains de ses problèmes, notamment la diminution du taux d'apprentissage au fil du temps. Au lieu de maintenir un historique des gradients accumulés, Adadelta utilise une moyenne mobile des mises à jour des paramètres [JAZ+18].

— Nesterov Accelerated Gradient (NAG) : NAG est une variante de la descente de gradient avec momentum qui calcule le gradient non pas aux paramètres actuels, mais aux paramètres "prédits" après avoir pris une étape de momentum [SMDH13].

Diverses techniques sont utilisées pour améliorer l'efficacité et la robustesse de l'entraînement d'un modèle. Souvent, la perte de plusieurs entrées est calculée avant qu'une seule passe de mise à jour des poids ne soit effectuée. C'est ce qu'on appelle le **dosage(batching**), qui permet d'accélérer et de stabiliser le processus. L'efficacité de l'entraînement des réseaux de neurones dépend également de la méthode d'optimisation utilisée.

D'autres méthodes d'optimisation existent, comme le **Momentum**, qui accélère la SGD dans des directions pertinentes tout en réduisant les fluctuations pour une convergence plus rapide et la **Batch Normalization**, qui normalise les entrées de chaque couche pour faciliter un entraînement plus stable et rapide des réseaux de neurones profonds.

L'Importance des Fonctions d'Activation dans l'Entraînement des Réseaux de Neurones

Les fonctions d'activation sont essentielles dans les réseaux de neurones, car elles permettent à ces modèles d'apprendre et de modéliser des relations non linéaires complexes entre les données d'entrée et de sortie. En l'absence de ces fonctions, un réseau de neurones serait réduit à un simple modèle linéaire, limitant sa capacité à résoudre des problèmes complexes qui nécessitent une compréhension nuancée des données. Les fonctions d'activation les plus fréquemment utilisées sont ReLU (Rectified Linear Unit), Sigmoïde, Tanh (Tangente Hyperbolique), et Softmax. ReLU est préférée pour les couches cachées en raison de sa simplicité et de son efficacité à accélérer la convergence de l'entraînement, tandis que la fonction Softmax est idéale pour la couche de sortie dans les tâches de classification multiclasse. Sélectionner une fonction d'activation peut contribuer à réduire

le problème de la disparition des gradients, un obstacle fréquent lors de l'apprentissage des DNNs, en garantissant que les gradients conservent une amplitude adéquate pour permettre une actualisation efficace des poids sur plusieurs couches.

3.3 DNN, Enjeux Environnementaux

Les DNNs, caractérisés par leur architecture multicouche, représentation dans la Figure 3.4, se distinguent par leur capacité à effectuer un apprentissage automatique à partir de données brutes. Le domaine du traitement automatique du langage naturel a notamment bénéficié des avancées avec l'émergence de nouveaux modèles comme BERT, un modèle bidirectionnel basé sur des transformateurs (Bidirectional Encoder Representations from Transformers)[PNI+18], [DCLT18], [CGRS19], [SLL19], par exemple.

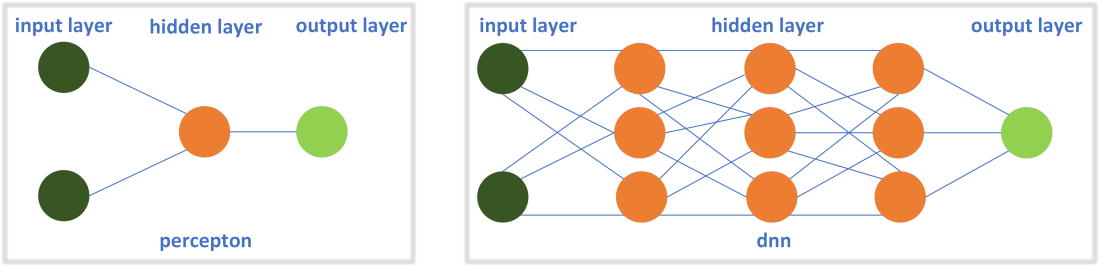

FIGURE 3.4 – Diagramme d'une structure de réseau de neurones profond DNN

Cependant, l'amélioration de la précision des modèles de NLP et d'autres applications de l'IA s'accompagne d'une augmentation significative de la consommation énergétique. Cette tendance soulève des préoccupations environnementales et met en lumière le besoin de développer des approches plus efficaces et économes en énergie pour l'entraînement et le déploiement de ces modèles avancés.

Entraînement et Inférence, l'Impact Environnemental

Les avancées technologiques et matérielles récentes dédiées à l'entraînement des DNNs ont conduit à des améliorations notables de leur précision [BCB15], [LSL+15], [DM17], [VSP+17], [NJPS21]. Les modèles nécessitant des calculs plus complexes ont progressivement obtenu des meilleurs résultats de précision [PNI+18], [DCLT18], [RWC+19], [SLL19]. Ces progrès, qui ont également été appliqués à l'entraînement des DNNs, ont contribué aussi à des améliorations de la précision des modèles de type NLP [PNI+18], [DCLT18], [SLL19].

Depuis maintenant plus de 10 ans, la puissance de calcul utilisée pour la mise en place des grands modèles d'IA a augmentée de plus de 300 000 fois selon [DD18], ce qui n'est pas ni viable à long terme, ni raisonnable de point de vue environnemental. Selon [BMR+20], le nombre de paramètres d'un modèle IA est étroitement lié à la puissance de calcul nécessaire pour former ces modèles de réseaux de neurones, et le taux augmente d'environ 10 fois par an.

La fin de la loi de Moore [13] prévue pour 2030 [14] soulève des préoccupations quant à la capacité des technologies commerciales conventionnelles à répondre aux exigences de calcul croissantes. En effet, l'entraînement de modèles de dernière génération nécessite des ressources de calcul substantielles, entraînant une consommation d'énergie importante et des coûts financiers et environnementaux associés.

Des études ont montré que les émissions de carbone générées lors de l'entraînement de modèles NLP peuvent être plusieurs fois supérieures à celles d'une voiture sur toute sa durée de vie [SGM20], [BGNL21]. La communauté de l'IA s'est souvent concentrée sur la précision des modèles sans prendre en compte leur coût environnemental. Il est temps d'examiner de manière critique l'impact de notre travail sur l'environnement [SDSE20], [Dha20], [VSC23], [MXL+23].

On estime qu'il faut réduire les émissions de carbone de moitié au cours de la prochaine décennie pour décourager l'escalade des catastrophes naturelles [15], l'entraînement et le développement du modèle constituent une part substantielle des émissions de gaz à effet de serre attribuées à de nombreux entraînements des algorithmes de NLP.

Atteindre des réseaux de neurones efficaces en temps réel avec une précision optimale nécessite de repenser la conception, l'entraînement et le déploiement des modèles [PPA18], [AAH+23]. Dans ce sens, de nombreuses méthodes ont été introduites pour alléger les modèles et les rendre plus compacts. Une des approches pour réduire la consommation d'énergie consiste à appliquer des **techniques de compression** qui réduisent considérablement le nombre de paramètres et de calculs des modèles d'apprentissage profond, tels que l'élagage (pruning), la distillation et la quantification à faible précision, [MGD20], [MPMF23], [LLM23], tout en veillant simultanément à ce que la précision du

13. Les lois de Moore sont des lois empiriques qui ont trait à l'évolution de la puissance de calcul des ordinateurs et de la complexité du matériel informatique.

La première de ces lois est émise par le docteur Gordon E. Moore en 1965, lorsque celui-ci postule sur une poursuite du doublement de la complexité des semi-conducteurs tous les ans à coût constant. Dix ans plus tard Moore ajusta sa prédiction à un doublement du nombre de transistors présents sur une puce de microprocesseur tous les deux ans. Ce second postulat se révéla exact, et popularisa le terme de « loi de Moore », si bien que ce dernier a fini par s'étendre au doublement d'une capacité quelconque en un temps donné.

14. Conclusion du livre blanc « More Moore » publié par le groupe de travail IRDS de IEEE sous la direction de Paolo Gardini, ancien directeur de la technologie chez Intel.

15. le rapport spécial du Groupe d'experts intergouvernemental sur l'évolution du climat (GIEC) sur le réchauffement global de 1,5 °C

modèle ne soit pas significativement affectée.

Cependant, de nouvelles méthodes doivent continuer à être étudiées pour repousser les limites des modèles. *Pourquoi des modèles plus légers ?*

Lorsque l'on compare le mouvement des données dans le stockage à celui des unités de calcul, l'accès aux données de stockage consomme beaucoup plus d'énergie que le traitement des données dans l'unité de calcul, Figure 3.5, [SLGKAE20].

Opération	Energie [pJ]	Cout relatif
32 bit DRAM Memory	**640**	**6400**
32 bit SRAM Cache	5	50
32 bit float MULT	3,7	37
32 bit int MULT	3,1	31
32 bit Register File	1	10
32 bit float ADD	0,9	9
32 bit int ADD	0,1	1

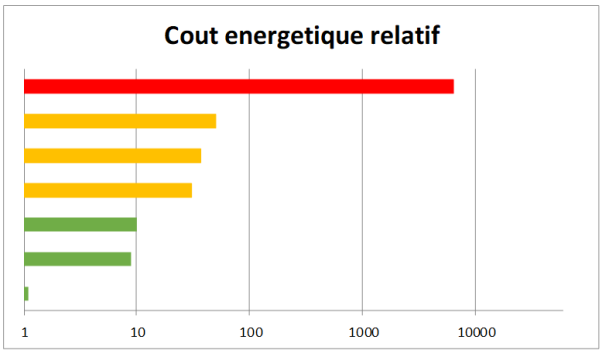

FIGURE 3.5 – Coût relatif d'accès à la mémoire. Source [Hor14]

Les réseaux de neurones de taille réduite présentent des avantages en termes d'efficacité mémoire et de capacité de calcul par rapport à leurs homologues plus volumineux. Une taille réduite permet à ces réseaux de neurones de s'adapter à la mémoire SRAM embarquée, qui offre un accès plus rapide que la mémoire DRAM externe. Cette adaptation à une mémoire plus rapide réduit les cycles nécessaires pour récupérer le modèle, ce qui contribue à améliorer la vitesse d'exécution des inférences.

De plus, les modèles de réseaux de neurones de taille plus petite requièrent moins d'opérations arithmétiques et de cycles de calcul [SEKH23]. Cette réduction des calculs se traduit par une diminution de la consommation d'énergie et une amélioration de l'efficacité énergétique lors de l'exécution des tâches d'inférence. En réduisant le nombre de cycles de référence mémoire nécessaires, les modèles de taille réduite optimisent également l'utilisation de la mémoire, ce qui contribue à une meilleure gestion des ressources matérielles.

De ce fait, les réseaux de neurones plus petits sont adaptés aux applications embarquées ou aux dispositifs avec des contraintes de puissance ou de temps de réponse. Leur capacité à effectuer des prédictions plus rapidement et avec une utilisation moindre des ressources matérielles les rend attrayants pour une utilisation dans des environnements où l'efficacité énergétique et la vitesse d'exécution sont préoccupantes.

A première vue, le coût énergétique de l'entraînement des modèles d'IA peut sembler plus élevé en raison de la nécessité de vastes ressources de calcul pour traiter de grandes quantités de données et optimiser les paramètres du modèle. Cependant, une fois que le modèle est entraîné et déployé en production, les coûts d'inférence peuvent dépasser les coûts d'entraînement pour plusieurs raisons :

Échelle de déploiement : En production, les modèles d'IA peuvent être déployés pour servir un grand nombre d'utilisateurs ou de sollicitations (requêtes), ce qui nécessite des infrastructures informatiques conséquentes pour gérer simultanément de multiples demandes d'inférence. Cette mise à l'échelle peut entraîner des coûts énergétiques importants, surtout si les charges de travail d'inférence sont variables et imprévisibles.

Utilisation en continu : Contrairement à l'entraînement, qui peut être effectué de manière périodique ou intermittente, l'inférence doit souvent être disponible en continu pour répondre aux demandes des utilisateurs ou des systèmes en temps réel. Cela signifie que les infrastructures doivent fonctionner en permanence pour garantir la disponibilité des services, ce qui peut entraîner des coûts énergétiques continus.

Complexité des opérations d'inférence : Certaines tâches d'inférence, telles que la classification d'images haute résolution ou le traitement du langage naturel complexe, peuvent nécessiter des calculs intensifs qui consomment davantage d'énergie que des tâches d'entraînement moins complexes. La complexité des opérations d'inférence peut donc contribuer à des coûts énergétiques plus élevés.

A titre d'exemple, et pour confirmer cette hypothèse, Amazon Web Services a révélé que : " l'inférence représente en réalité la majorité du coût et de la complexité de l'exécution de l'apprentissage automatique en production (pour chaque dollar dépensé en phase d'entraînement, neuf sont dépensés en inférence)" [16].

L'empreinte carbone est répartie de manière égale entre l'entraînement et l'inférence, pour les cas d'utilisation de recommandation [XLC+21] .

Il existe plusieurs études sur le calcul de l'entraînement et son impact environnemental [DD18], [CCP17], mais il en existe très peu axées sur les coûts d'inférence et leur consommation d'énergie associée [XLC+21]. Ces algorithmes nécessitent des niveaux élevés de puissance de calcul pendant l'entraînement, car ils doivent être formés sur de grandes quantités de données, tandis que lors du déploiement, ils peuvent être utilisés plusieurs fois.

16. Amazon Web Services

La sur-paramétrisation

Les réseaux de neurones, devenus plus précis grâce à un nombre croissant de paramètres, entraînent une augmentation significative des besoins en calcul et en énergie. Bien que la sur-paramétrisation facilite l'entraînement avec des méthodes comme la descente de gradient stochastique, elle engendre une consommation énergétique augmentée. Des approches comme la compression des modèles peuvent offrir une voie pour minimiser l'empreinte énergétique sans compromettre la précision [SYCE19], [YCF+23], [GFF+22], [KMH+20], [BGMSS18], [BGMSS18], [NLB+18], [AZLS19]

Au cours des dernières années il y a eu des améliorations significatives de la précision des réseaux de neurones pour des tâches très variées et assez souvent bien spécifiques, régulièrement obtenues grâce à des modèles sur-paramétrés [17]. Dans de nombreux cas, la sur-paramétrisation est utilisée délibérément pour permettre au modèle d'apprendre des représentations complexes des données et d'obtenir de meilleures performances sur des ensembles de données d'entraînement. En revanche, une sur-paramétrisation excessive peut entraîner aussi un sur-apprentissage (overfitting), où le modèle mémorise les données d'entraînement au lieu de généraliser des motifs qui s'appliquent à de nouvelles données. Il existe une observation intéressante selon laquelle les modèles sur-paramétrés sont souvent plus faciles à entraîner avec des algorithmes d'optimisation tels que la descente de gradient stochastique que des modèles plus compacts [YCF+23].

L'idée sous-jacente est que les modèles avec un grand nombre de paramètres ont une plus grande capacité à capturer la complexité présente dans les données, ce qui signifie qu'ils peuvent mieux s'adapter aux nuances et aux détails des données d'entraînement. Cela peut se traduire par des surfaces de coût plus lisses et moins de minima locaux dans l'espace des paramètres, rendant l'optimisation plus efficace. Les techniques de descente de gradient et de la régularisation permettent un entraînement optimal des réseaux sur paramétrés tout en assurant une bonne capacité de convergence [GFF+22], [SYCE19].

Des travaux théoriques récents, [NLB+18], [AZLS19], [HLX+23], offrent un éclairage sur les mécanismes sous-jacents qui influencent la dynamique d'entraînement et la capacité de généralisation des réseaux de neurones surparamétrés. En intégrant ces connaissances dans la conception et l'entraînement des modèles, il est possible de mieux comprendre et exploiter les avantages de la sur-paramétrisation tout en évitant ses inconvénients, comme le sur-apprentissage.

Les techniques pour régulariser les modèles sur paramètres consiste à intro-

17. La sur-paramétrisation (en anglais, overparameterization) dans le contexte des réseaux de neurones fait référence à une situation où le nombre de paramètres (ou poids) dans un modèle de réseau neuronal est plus élevé que nécessaire pour résoudre une tâche donnée. En d'autres termes, le modèle a plus de capacité d'apprentissage qu'il n'en a besoin pour représenter la relation entre les données d'entrée et de sortie.

duire un terme de pénalité dans la fonction de coût du modèle pour contrôler la complexité du modèle et encourager la simplicité des poids. Parmi les méthodes de régularisation les plus courantes, on trouve :

— *Régularisation L1* : connue sous le nom de Lasso (Least Absolute Shrinkage and Selection Operator) [Tib11], est une technique de régularisation utilisée pour prévenir le sur-apprentissage en imposant des pénalités sur la magnitude absolue des coefficients de régression. En ajoutant la somme des valeurs absolues des poids, multipliée par un paramètre de régularisation λ, à la fonction de coût, la régularisation L1 encourage le modèle à maintenir les poids aussi petits que possible. La fonction de coût régularisée peut s'écrire comme suit pour la régression linéaire :

$$L(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{3.10}$$

où y_i est la valeur cible, x_{ij} est la valeur du j-ième prédicteur pour l'i-ème observation, β_j sont les coefficients de régression, et λ est le paramètre de régularisation qui contrôle l'ampleur de la pénalité.

La régularisation L1 présente l'avantage de produire des modèles parcimonieux [18]. Cela est dû au fait que, pour des valeurs suffisamment élevées de *lambda*, certains coefficients peuvent être poussés exactement à zéro, effectuant ainsi une sélection automatique des caractéristiques en éliminant les prédicteurs les moins importants du modèle. Cette action est utile dans des situations où le nombre de prédicteurs est très élevé par rapport au nombre d'observations, ou lorsque quelques variables explicatives sont présumées avoir un impact significatif sur la variable réponse.

— *Régularisation L2* : Contrairement à L1, la régularisation L2, également connue sous le nom de Ridge [HTJW21], ajoute la somme des carrés des poids multipliée par un paramètre de régularisation λ. Elle a tendance à distribuer l'erreur entre tous les termes, conduisant à des modèles avec des poids petits mais non nuls.

La fonction de coût modifiée par la régularisation L2 pour la régression linéaire peut être exprimée comme suit :

$$L(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{3.11}$$

où y_i est la valeur cible, x_{ij} est la valeur du j-ième prédicteur pour l'i-ème observation, β_j sont les coefficients de régression, et λ est le paramètre de régularisation.

18. Modèles avec un nombre réduit de coefficients non nuls.

L2 réduit la magnitude des poids mais ne les rend généralement pas nuls. La régularisation L2 est utile pour les situations où il est préférable de conserver tous les prédicteurs dans le modèle mais avec des poids de petite taille pour atténuer le sur-apprentissage.

— *Dropout* : Cette méthode consiste à éteindre aléatoirement une partie des neurones (mettre leurs sorties à zéro) à chaque étape d'entraînement [SHK+14], ce qui empêche les neurones de devenir trop dépendants les uns des autres pendant l'apprentissage. Cette indépendance forcée entre les neurones amène le réseau à développer une représentation plus robuste des données, car il ne peut pas compter sur la présence de neurones spécifiques, améliorant ainsi sa capacité à généraliser à de nouvelles données inédites.

  Le Dropout agit en simulant l'entraînement de multiples configurations de réseau en parallèle, bien qu'utilisant un ensemble commun de poids. Bien que ces configurations partielles puissent individuellement offrir des performances sub-optimales, l'agrégation de leurs résultats conduit généralement à une meilleure capacité de généralisation et à une efficacité accrue lorsqu'elles sont confrontées à de nouvelles données. L'application du Dropout est simple et efficace, rendant les DNNs moins susceptibles de sur-apprentissage sans nécessiter de modifications l'architecture ou de l'algorithme d'entraînement.

Les régularisations Lasso et Ridge aident à minimiser la taille des poids au sein des modèles. Moins de poids ou des poids plus petits signifient moins de données à stocker en mémoire et moins d'informations à récupérer lors de chaque opération de calcul, réduisant ainsi la charge sur la mémoire externe et, par conséquent, la consommation d'énergie associée à l'accès à la mémoire.

D'autre part, la technique du Dropout diminue la complexité des modèles en désactivant aléatoirement certains neurones durant l'entraînement. Cette approche limite la dépendance du réseau à des entrées ou des chemins neuronaux spécifiques. Elle conduit au développement de modèles plus robustes et performants.

Les méthodes de régularisation optimisent des grands modèles de DNN non seulement pour prévenir le sur-apprentissage mais aussi pour **réduire leur consommation énergétique.** Ces techniques de régularisation peuvent influencer l'efficacité énergétique des DNNs en réduisant la complexité du modèle, ce qui a un impact direct sur la quantité de données à stocker et à récupérer de la mémoire externe, par exemple la DRAM.

3.4 NLP, Numérique Responsable

Dans cette section nous aborderons en profondeur les différentes techniques de réduction de la complexité des modèles, notamment la compression de modèles. Ces approches visent à réduire la taille et les exigences en ressources des modèles de NLP sans compromettre leurs performances.

3.4.1 Techniques de Compression : Principes et Approches

L'évolution des techniques de compression des modèles de NLP a suivi la croissance exponentielle de la taille de ces modèles.

Au début, les modèles de langage étaient relativement simples et petits, mais avec l'arrivée de modèles comme BERT [DCLT18] et GPT [BMR+20], leur taille a beaucoup augmenté, nécessitant ainsi de nouvelles méthodes de compression, comme le montre la Figure 3.6.

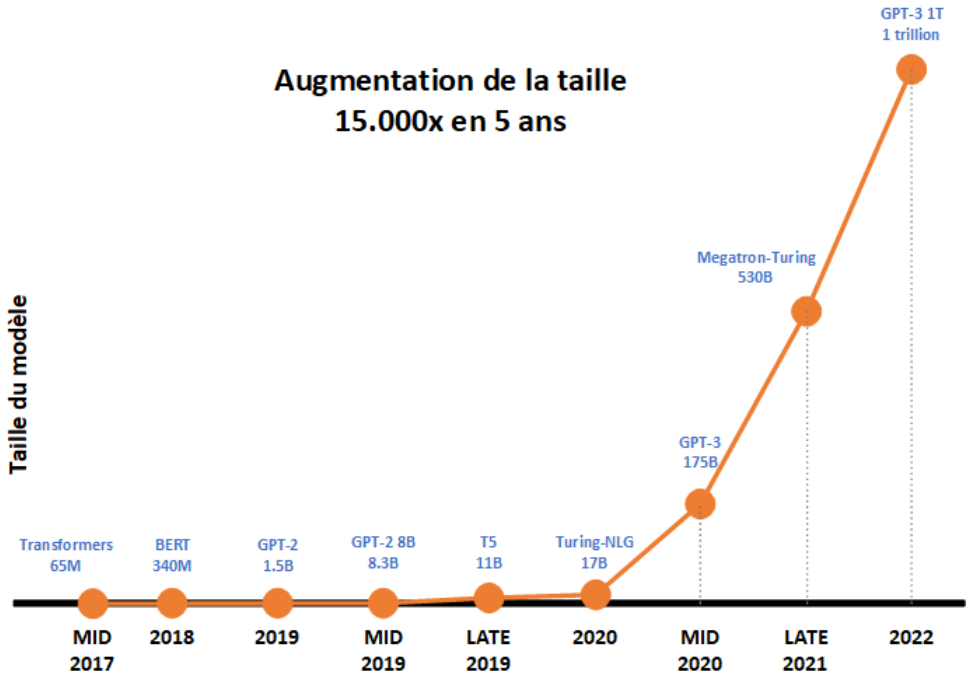

FIGURE 3.6 – Evolution de la taille de modèles avec le temps [GZS+21]

La nécessité de compresser ces modèles sans altérer leur performance a conduit à des avancées significatives dans les techniques de compression [CWZZ17], [JYS+20].

Principes de Base des Techniques de Compression

La compression dans le domaine du NLP se réfère à l'ensemble des méthodes visant à réduire la taille des modèles de langage. L'objectif principal est de diminuer la consommation de ressources computationnelles et de mémoire, tout en préservant, autant que possible, la qualité et la précision des modèles. Comme [HMD16] le soulignent la compression vise à rendre les modèles de NLP plus efficaces et accessibles, comme pour les applications nécessitant une faible latence ou fonctionnant sur des appareils avec des capacités de calcul limitées.

Les principaux principes sur lesquels reposent ces techniques de compression incluent :

Réduction de la dimensionnalité La réduction de la dimensionnalité consiste à réduire le nombre de paramètres du modèle tout en préservant au mieux sa capacité à généraliser. Cette approche peut être réalisée en utilisant des techniques telles que la factorisation de matrices.
Parcimonie des modèles La parcimonie consiste à identifier et à éliminer les paramètres du modèle qui ont peu d'impact sur sa performance globale. Cela permet de réduire la complexité tout en maintenant des niveaux acceptables de précision. Les méthodes de régularisation, telles que la régularisation L1, sont souvent utilisées pour encourager la parcimonie.
Efficacité computationnelle La compression des modèles vise également à améliorer l'efficacité computationnelle en réduisant les besoins en puissance de calcul pour l'inférence. Cela peut être accompli en quantifiant les poids du modèle ou en utilisant des architectures plus légères.

Approches de Compression

Il existe deux approches principales : la compression pendant l'entraînement et celle effectuée juste avant la phase d'inférence, après l'entraînement.

Compression durant l'entraînement : Cette approche vise à réduire la taille du modèle dès le processus d'entraînement. Elle peut impliquer l'utilisation de techniques telles que l'élagage (pruning) des poids moins importants, l'utilisation de réseaux plus compacts, ou encore l'utilisation de la régularisation pour encourager la parcimonie des modèles :

— Quantification de poids : Réduit la précision des poids du modèle (par exemple, de 32 bits en virgule flottante à 8 bits en virgule fixe) tout en préservant les performances du modèle [ZHMD16]
— Élagage (Pruning) : Suppression des poids du modèle qui ont une faible valeur absolue ou une faible contribution à la sortie. Cela peut être basée sur la magnitude ou sur d'autres critères [HPTD15].

— Distillation des connaissances (Knowledge Distillation) : Entraîner un modèle plus petit (l'étudiant) pour reproduire les sorties d'un modèle plus grand (le professeur). Cela peut réduire la complexité du modèle tout en maintenant les performances [HVD15].

Compression après l'entraînement : À l'inverse, la compression après l'entraînement intervient après que le modèle a été complètement formé à sa taille initiale. Une fois que le modèle a acquis sa performance maximale, des techniques de compression sont appliquées pour réduire sa taille tout en préservant sa capacité à bien fonctionner. Cela peut inclure des méthodes :

— Quantification post-entraînement : Réduire la précision des poids d'un modèle déjà entraîné, généralement sans ré-entraîner le modèle [JKC+18].
— Distillation de modèle : Ré-entraîner un modèle avec des données compressées ou synthétiques pour obtenir un modèle plus petit tout en préservant les performances [SDCW19].
— Factorisation de matrice : Décomposer les poids du modèle en matrices de facteurs pour réduire la taille du modèle [LGR+15].

Certaines situations peuvent nécessiter une compression dès le début pour économiser des ressources computationnelles, tandis que d'autres peuvent privilégier la compression après l'entraînement pour conserver la qualité du modèle initial. Il est essentiel de comprendre ces deux approches pour prendre des décisions éclairées lors de la compression des modèles de langage.

Initialement, la compression des modèles de langage était principalement axée sur des méthodes simples comme la réduction de la précision des poids (quantification). Cependant, avec l'augmentation de la complexité des modèles, des techniques plus sophistiquées ont été développées, telles que l'élagage, la factorisation de rang faible, et d'autres méthodes de réduction dimensionnelle. Ces techniques ont évolué pour s'adapter non seulement aux contraintes de taille et de calcul mais aussi pour optimiser les modèles pour des tâches spécifiques de NLP [HVD15].

3.4.2 Techniques de Compression : Focus sur NLP

Le domaine de la compression des modèles NLP est apparu comme un domaine de recherche et d'application critique. Cela permet un déploiement plus efficace et ouvre la voie à une adoption généralisée dans divers domaines (Figure 3.7). "Les techniques de compression de modèle offrent une solution pragmatique au dilemme des ressources posé par les grands modèles NLP", [SDCW19], [KCK21].

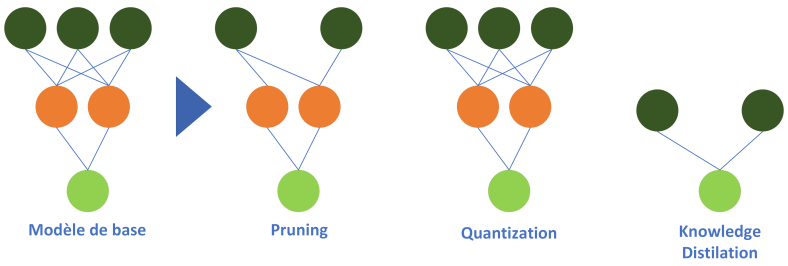

FIGURE 3.7 – Représentation des Méthodes de Compression [KCK21]

Le choix de la technique de compression dépend de la taille souhaitée du modèle, la perte acceptable de performances et les caractéristiques de l'ensemble de données. Il est possible de choisir la combinaison de plusieurs techniques pour atteindre le niveau de compression souhaité tout en maintenant les performances de la tâche[HMD16].

Sparsification

La sparsification représente une approche importante pour réduire la complexité et les coûts associés aux réseaux de neurones et fait référence à une approche visant à améliorer l'efficacité computationnelle et la performance des modèles d'apprentissage automatique, notamment des réseaux de neurones, en réduisant le nombre de calculs nécessaires pour effectuer des opérations importantes. Cette stratégie s'inspire de l'observation selon laquelle tous les éléments d'un modèle ne contribuent pas également à sa sortie, ce qui signifie que de nombreuses opérations de calcul peuvent être évitées sans sacrifier la précision du modèle.

La sparsification représente la pratique d'augmenter la proportion de zéros au sein des poids d'un réseau de neurones, rendant ainsi le modèle plus "sparse". Cette technique peut réduire significativement la quantité de mémoire nécessaire pour stocker le modèle et peut diminuer le nombre d'opérations de calcul requises lors de l'inférence, en exploitant l'efficacité des opérations sur des matrices sparse. La sparsification peut être atteinte de manière statique (en fixant certains poids à zéro de manière arbitraire ou selon un critère défini avant l'entraînement) ou dynamiquement (au cours de l'entraînement, ajustant la structure du réseau pour augmenter sa sparsité) [LWF+15], [GBB11], [LWTL17], [MMS+18].

L'**élagage des poids** et l'**attention clairsemée** (Sparse Attention) [BPC20] sont des exemples concrets de la mise en œuvre de la sparsification. L'élagage des poids implique la suppression des poids inutiles ou de minime importance, ce qui permet de compresser les modèles clairsemés et de les affiner ensuite pour restaurer les performances. L'attention clairsemée cible une optimisation spécifique au sein des mécanismes d'attention des modèles, limitant les schémas d'attention à un sous-ensemble de jetons

(tokens) ou de positions, ce qui diminue le nombre d'interactions et de paramètres nécessaires. Cette approche peut être appliquée de manière statique, en établissant a priori quels poids seront réduits à zéro, ou dynamiquement, en ajustant la structure du réseau pendant l'entraînement pour accroître sa sparsité.

L'**élagage des têtes d'attention** [KMH+20] offre une méthode raffinée d'élagage en ciblant des têtes d'attention spécifiques au sein d'une couche, qui ont un moindre impact sur la performance du modèle.

Elagage

L'élagage consiste à éliminer délibérément certains neurones ou connexions du réseau de neurones, permettant ainsi d'obtenir un modèle plus léger tout en préservant ses performances. Cette stratégie est pertinente dans le contexte de la réduction de la consommation énergétique des réseaux de neurones.

L'idée de "pruning" n'est pas nouvelle et remonte aux années 1980 [Kar90], [GT95] mais elle a connu une explosion d'intérêt au cours de la dernière décennie, principalement en raison de la montée en puissance des DNNs [BOFG20], [BRSH22]. Les modèles "pruned" présentent moins de paramètres, et par conséquent, moins de coûts de calculs théoriques. Cependant, le défi majeur réside dans la manière de maintenir l'exactitude du modèle tout en éliminant une partie de ses composants.

Les réseaux de neurones de taille réduite exigent moins d'espace de stockage pour leurs poids, diminuant ainsi la bande passante mémoire requise pour accéder aux paramètres du modèle. Cette réduction de la demande en mémoire peut conduire à des gains significatifs en termes d'efficacité énergétique, car l'accès à la mémoire est l'un des principaux contributeurs à la consommation d'énergie [Hor14]. De plus, les DNNs plus petits impliquent moins d'opérations arithmétiques et de cycles de calcul, ce qui se traduit par une diminution de la consommation d'énergie liée aux calculs.

L'élagage peut prendre différentes formes, notamment le "channel pruning," le "filter pruning," le "connection pruning," et le "layer pruning" [SWR20]. Les réseaux plus volumineux offrent généralement davantage de possibilités de l'élagage, ce qui explique pourquoi la tendance à la compression se poursuit à mesure que les architectures deviennent plus grandes.

Des études ont montré que les réseaux élagués peuvent généraliser tout aussi bien, voire mieux, que leurs homologues denses originaux. Dans certains cas, l'élagage peut même améliorer la précision, surtout pour des niveaux de compression relativement faibles [HPTD15], [SAM+18], [SAM+20].

Au-delà de l'élagage, d'autres stratégies comme les **techniques de factorisation**

[DP19] et le **regroupement des matrices de poids** [ZNM+20] contribuent également à la réduction de la taille des modèles. La factorisation permet de décomposer les matrices de poids ou d'attention pour minimiser leur taille tout en conservant l'essentiel de l'information. Parallèlement, le regroupement de matrices de poids consolide les poids similaires, réduisant le nombre de valeurs uniques nécessaires.

L'**élagage des couches entières** [VTM+20], en éliminant les parties du modèle contribuant peu à la performance, représente une autre approche pour simplifier les architectures DNN tout en optimisant leur efficacité énergétique.

Réduction de la Précision : Optimiser les Représentations

La réduction de la précision des poids et des activations des réseaux de neurones tout en maintenant la précision du modèle est une autre approche pour optimiser la consommation d'énergie dans le domaine de l'apprentissage automatique. Cette stratégie vise à minimiser la quantité de bits nécessaires pour représenter les paramètres et les activations des réseaux de neurones, tout en préservant leur capacité à effectuer des tâches avec précision. La Figure 3.8 représente quelques options de précision utilisées pour la construction d'un modèle IA.

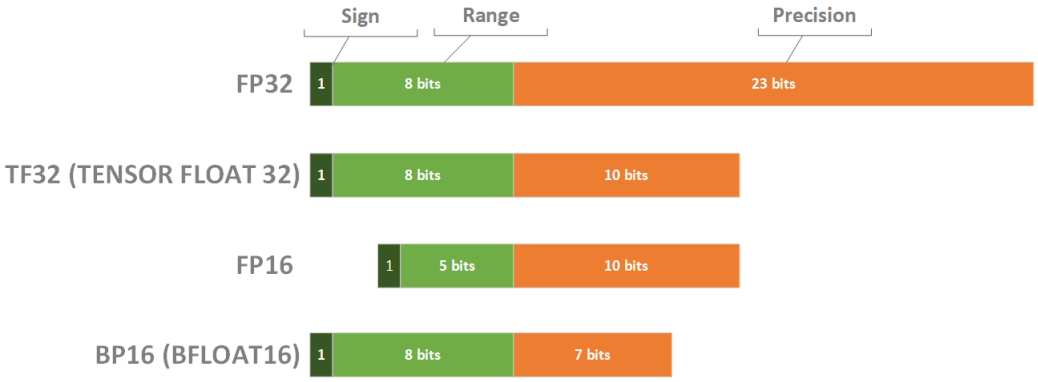

FIGURE 3.8 – Options de précision utilisées pour l'entraînement d'un modèle IA [Kha20]

L'une des raisons de la réduction de la précision réside dans l'optimisation des opérations en virgule flottante. Les opérations en virgule flottante traditionnelles utilisent des nombres à haute précision (par exemple, 32 bits) pour représenter les valeurs, ce qui peut être excessif dans de nombreuses applications. Réduire la précision des données à un niveau inférieur (par exemple,16 bits, 8 bits, 4 bits) [Hor14] peut réduire la consommation d'énergie tout en maintenant des performances acceptables.

Cette transition vers une représentation à plus faible précision a un impact significatif

sur la bande passante mémoire, la consommation d'énergie et l'espace de stockage.

Des techniques telles que la quantification à faible précision (par exemple, 16 bits ou moins) ont été largement explorées dans la littérature scientifique [LGW+21]. Les résultats de ces techniques de quantification ont montré que des réductions de 4x à 8x en bande passante mémoire et en énergie sont souvent réalisables en pratique pour ces applications [GKD+21]. Une illustration de la quantification à faible précision est donnée dans la Figure 3.8.

Certaines études ont démontré que le passage des représentations en virgule flottante à des valeurs entières de 4 bits peut réduire considérablement l'empreinte mémoire et la latence, souvent jusqu'à un facteur de 16x, [GKD+21]. Cela permet d'obtenir de l'efficacité en termes de calcul, de stockage et d'énergie, sans compromettre de manière significative la précision du modèle [HABN+21].

La quantification des poids et la généralisation de la quantification [HCS+18] à l'ensemble du modèle permettent des réductions significatives de la taille du modèle sans compromettre de manière substantielle la précision.

Réseaux Denses Plus Petits

La condensation de modèles vise à créer des réseaux de neurones plus petits et plus compacts tout en maintenant leur capacité à résoudre des tâches complexes.

L'une des techniques les plus répandue dans ce domaine est la **Distillation de Connaissances** [HVD15]. Elle consiste à transférer la connaissance d'un modèle plus grand, souvent qualifié de modèle **enseignant**, vers un modèle plus petit, appelé modèle **élève**. Cette méthode permet au modèle élève d'apprendre de manière plus efficace en utilisant les informations du modèle enseignant, ce qui se traduit par une réduction significative du nombre de paramètres tout en préservant la performance [PPA18]. En d'autres termes, la distillation de connaissances permet de créer des réseaux de neurones plus petits sans sacrifier la qualité des prédictions.

Cette méthode est complétée par d'autres approches comme l'**Adaptation des connaissances**, où le réglage fin sur des ensembles de données spécifiques permet d'optimiser les modèles pré-entraînés pour des tâches ciblées, et le **Plongement de sous-mots**, qui minimise la taille du vocabulaire et de la matrice de plongement (embedding) [PY10].

La **Distillation de modèle** [BCNM06] et l'**Apprentissage par transfert** sont d'autres techniques importantes, permettant de créer des modèles plus légers et performants en s'appuyant sur la connaissance préalable. De même, le **Partage de paramètres** [LPM15] favorise l'efficience en évitant la redondance dans le modèle.

Recherche d'Architecture Neuronale

Une autre approche prometteuse est la "Recherche d'Architecture Neuronale" [EMH19], qui vise à découvrir des architectures de modèle plus compactes tout en maintenant leur efficacité. Cette méthode consiste à explorer un espace de recherche d'architectures pour trouver des configurations qui minimisent la complexité tout en maximisant la performance. Ces architectures neuronales optimisées permettent de créer des modèles plus petits tout en conservant leur capacité à effectuer des tâches complexes.

Ce qui rend ces méthodes encore plus attrayantes, c'est qu'elles ont montré la résilience face à la compression agressive. Contrairement à certaines techniques de quantification et d'élagage qui peuvent entraîner une perte de précision significative avec une compression importante, la distillation de connaissances et la recherche d'architecture neuronale ont tendance à maintenir une performance de haut niveau même avec une compression substantielle, [GKD+21].

Il est également important de noter que la combinaison de la distillation de connaissances avec d'autres techniques telles que la quantification et l'élagage a montré une grande réussite [PPA18]. Cette approche globale permet d'améliorer la vitesse d'inférence et de réduire les coûts d'inférence par rapport à une simple réduction de la profondeur du modèle.

Adoption de DistilBERT et Variantes Simplifiées

Une autre approche pour réduire la consommation d'énergie consiste à choisir des modèles spécifiquement conçus pour l'efficacité. DistilBERT est une version condensée du modèle BERT d'origine. Il conserve en grande partie les performances de BERT tout en réduisant sa taille et ses exigences computationnelles. Ce choix permet une inférence efficace sur des matériels aux ressources limitées [SDCW19].

Transformers efficients

Plusieurs architectures basées sur des transformers efficaces ont été introduites, conçues pour équilibrer la taille du modèle et ses performances. Des modèles tels que MobileBERT et TinyBERT sont spécifiquement conçus pour fonctionner efficacement sur des appareils mobiles et des périphériques. Ces modèles utilisent des techniques de distillation des connaissances et de distillation du modèle pour conserver leur efficacité tout en économisant de l'énergie [SYS+20].

Nous avons exploré l'évolution et l'importance croissante des méthodes de compression dans le domaine du NLP, un secteur caractérisé par des algorithmes de plus en plus gourmands en ressources. Nous avons commencé par un aperçu historique, soulignant comment le développement des techniques de compression a progressé en parallèle avec l'augmentation de la taille et de la complexité des modèles de NLP. Nous avons

ensuite examiné les différents principes de compression : la sparsification, l'élagage, et la réduction des modèles à des versions plus compactes se sont révélés être des stratégies efficaces pour atteindre cet objectif. Chacune de ces méthodes offre une voie pour alléger les charges de calcul et de stockage, tout en maintenant, voire parfois en améliorant, la performance des modèles.

Alors que précédemment nous avons mis en avant l'importance de choisir des modèles NLP plus simples pour diminuer l'empreinte énergétique et améliorer l'efficacité, nous abordons désormais la question sous un angle plus quantitatif et technique, notamment les indicateurs de mesure de la consommation d'énergie des modèles.

Dans une démarche d'optimisation de la consommation d'énergie des modèles NLP un élément très important dans la phase de validation et d'observation est la mesure. Pour obtenir ces indicateurs il faut se faire accompagner des outils spécialisés et adaptés tels que les wattmètres, les compteurs d'énergie électrique et des logiciels de mesure et de surveillance de la consommation d'énergie. Ces outils permettent de quantifier la consommation énergétique des modèles de NLP pendant l'inférence.

3.5 IA, regard sur la consommation d'énergie

Dans le contexte des applications IA, il est essentiel de quantifier la consommation d'énergie à travers la mesure afin de pouvoir identifier les parties à améliorer avec l'objectif de réduire leur impact énergétique.

Les chercheurs ont déjà commencé à s'intéresser à cette question et aujourd'hui il existe des estimations de la consommation énergétique pour certaines tâches spécifiques d'IA. La partie la plus couverte reste la phase d'entraînement des modèles du fait d'une forte conviction et médiatisation sur ce point précis.

A titre d'exemple, pour évaluer la performance des applications d'IA, des benchmarks ont été effectués afin de comparer les différents types de processeurs (CPU, GPU et TPU) en se basant sur des critères comme les opérations en virgule flottante par seconde (FLOPS) et la consommation totale d'énergie. Ces benchmarks sont essentiels pour comprendre comment différents processeurs se comportent lorsqu'ils exécutent des tâches d'IA et quel est leur impact énergétique.

Il est tout de même important de noter qu'il n'existe pas encore des normes écrites pour mesurer l'impact énergétique des applications d'IA, néanmoins, la prise de conscience de la nécessité de telles normes est de plus en plus forte, et aujourd'hui plusieurs organismes s'intéressent à la normalisation de la mesure de l'efficacité énergétique.

L'IEEE [19], par le biais de son Technology Climate Center (ITCC [20]), s'engage dans des discussions et des actions visant à évaluer l'impact environnemental des technologies numériques et de l'IA. Ils organisent des événements pour aborder des sujets tels que la contribution de la technologie aux objectifs climatiques, la durabilité et l'électrification. Cette attention croissante à l'impact environnemental de la technologie reflète la prise de conscience croissante de la responsabilité environnementale dans le secteur technologique et la nécessité de mesurer et de réduire cet impact.

La sensibilisation à l'éco-conception des applications d'IA et la nécessité de mesurer leur impact énergétique nous amènent naturellement à explorer les outils de mesure dédiés aux applications d'IA.

3.5.1 Outils de mesure pour les processeurs

Sous l'angle technologique, parmi les premiers à avoir mis à disposition des moyens de mesure de la consommation d'énergie sont les fabriquants des composants matériels, comme par exemple les fabriquants de processeurs CPU (unités de traitement central) ou de processeurs graphiques GPU (unités de traitement graphique).

Intel

Intel, largement connu pour la fabrication des processeurs CPU, offre une interface appelée Running Average Power Limit (RAPL), qui est une fonctionnalité intégrée aux processeurs Intel et qui permet de surveiller et de limiter la consommation d'énergie des processeurs.

L'interface RAPL est une fonctionnalité introduite à partir de l'architecture Intel Sandy Bridge et qui offre principalement deux fonctionnalités différentes. Une qui permet de mesurer la consommation d'énergie avec une très grande précision et un taux d'échantillonnage élevé et l'autre qui elle permet de limiter (ou de plafonner) la consommation électrique moyenne de différents composants à l'intérieur du processeur, ce qui limite également la production de chaleur du processeur [WJK+12].

Elle permet également aux utilisateurs d'interroger des registres non architecturaux spécifiques au modèle de processeur pour obtenir des informations liées notamment à la puissance CPU, la mesure clé de la consommation d'énergie.

19. IEEE : Institute of Electrical and Electronics Engineers, association professionnelle qui a pour but de promouvoir la connaissance dans le domaine de l'ingénierie électrotechnique, y compris électronique.
20. ITCC : IEEE Technology Climate Center

NVIDIA

NVIDIA, reconnu pour ses GPU, largement utilisés dans le domaine de l'IA et spécifiquement celui du Deep Learning pour leurs capacités de calcul parallèle, fournit un utilitaire appelé NVIDIA System Management Interface (NVIDIA SMI).

Cet utilitaire permet de surveiller et de gérer les cartes graphiques de manière générale, y compris les cartes de type GPU, en fournissant des informations comme la consommation d'énergie. Cette fonctionnalité est importante pour optimiser les performances et l'efficacité énergétique lors de l'utilisation d'applications d'IA qui exploite [NVIDIA System Management Interface (nvidia-smi) Documentation]

D'autres outils de mesure de la consommation d'énergie des processeurs peuvent être cités. Certains de ces outils sont spécifiques à certaines marques de processeurs ou de cartes mères, tandis que d'autres sont plus génériques et peuvent être utilisés avec différents types de processeurs.

Voici quelques exemples de ces outils de mesure de la consommation d'énergie des processeurs :

— **Intel Power Gadget** : Intel propose un outil appelé "Intel Power Gadget" qui permet de surveiller la consommation d'énergie des processeurs Intel. Il fournit des informations en temps réel sur la consommation d'énergie, la fréquence du processeur, etc.

— **Intel VTune Profiler** : Cet outil ne se limite pas à la mesure de l'énergie, mais offre une analyse détaillée des performances des applications, ce qui inclut indirectement des informations sur la consommation d'énergie.

— **AMD Ryzen Master** : Pour les processeurs AMD Ryzen, l'outil "Ryzen Master" fourni par AMD permet de surveiller et de gérer divers paramètres du processeur, y compris la consommation d'énergie.

— **Open Hardware Monitor** : Open Hardware Monitor est un logiciel open source qui permet de surveiller différents capteurs matériels, y compris la consommation d'énergie du CPU. Il prend en charge une variété de processeurs et de cartes mères.

— **CPU-Z** : CPU-Z est un outil utilisé pour obtenir des informations sur le matériel du système, y compris la consommation d'énergie du processeur. Il est plus axé sur la fourniture d'informations sur le matériel que sur la surveillance en temps réel.

— **Linux PowerTOP** : PowerTOP est un utilitaire spécialement conçu pour les systèmes Linux. Il permet de surveiller la consommation d'énergie du CPU et d'identifier les processus et les réglages du système qui peuvent contribuer à une consommation excessive.

Les benchmarks comme MLPerf [21] fournissent un ensemble standardisé de mesures de performances pour différentes plates-formes matérielles et logicielles. Cela permet aux entreprises de comparer objectivement leurs solutions avec celles de leurs concurrents. En ayant des métriques de performances communes, les entreprises peuvent évaluer la qualité de leurs produits et services par rapport à ceux du marché, ce qui est essentiel pour rester compétitif.

L'ajout de la mesure de l'efficacité énergétique dans MLPerf illustre une prise de conscience croissante de l'importance de la durabilité dans le domaine de l'apprentissage automatique. Cela aide l'industrie à se diriger vers des solutions plus éco-énergétiques.

Ces outils peuvent être utiles pour surveiller et mesurer la consommation d'énergie des processeurs, en fonction de la marque et du modèle du processeur utilisé. Dans les applications d'IA, où les calculs sont intensifs et la consommation d'énergie peut être significative, ces outils sont extrêmement utiles pour la surveillance pendant l'entraînement de modèles. En comprenant comment les applications d'IA consomment de l'énergie, les développeurs peuvent optimiser les codes et choisir les paramètres matériels les plus efficaces.

Ces outils sont avérées a être essentiels pour évaluer l'efficacité énergétique des applications d'IA et des modèles de réseaux de neurones dans leur différentes phases, d'entraînement ou inférence.

3.5.2 Outils de mesure pour les applications

Outils de mesure en ligne

Green Algorithms

L'outil Green Algorithms [22] [LGI21], [LGBI21], avec la méthode qu'il propose, offrent aux utilisateurs un moyen pratique d'estimer l'empreinte carbone de leurs calculs. La méthode se concentre sur la production d'estimations raisonnables avec de faibles coûts pour les scientifiques souhaitant mesurer l'empreinte de leur travail. La calculatrice proposée

21. Ensemble de benchmarks pour l'apprentissage automatique qui inclut une mesure de l'efficacité énergétique.
22. Green Algorithms

en ligne est simple à utiliser et généralisable à presque tout type de tâches de calcul, y compris des simulations de physique des particules ou des conditions météorologiques.

ML CO2 Impact

ML CO2 Impact [23] est un outil facilitateur pour mesurer l'équivalent de CO_2 généré lors de l'entraînement de modèles d'apprentissage automatique. Disponible en ligne, les utilisateurs reçoivent une estimation de l'impact carbone de leurs solutions en saisissant des informations sur le matériel, le runtime et le fournisseur de cloud.

A la fin des différents calculs, le calculateur produit deux chiffres : les émissions de carbone brutes et les émissions de carbone compensées estimées, qui peuvent être sujettes à changement en fonction du type d'hébergement utilisé pour les applications.

Un des inconvénients potentiels de cet outil est qu'il peut ne pas fournir une estimation précise pour tous les scénarios potentiellement envisageables. Le calculateur s'appuie sur des hypothèses et des estimations pour certains facteurs, tels que l'efficacité énergétique du matériel et l'intensité carbone du réseau électrique utilisé par le fournisseur des services Cloud. Il est important de tenir compte de cette limitation et d'utiliser l'estimation fournie par la calculatrice comme un repère approximatif plutôt qu'une mesure précise.

Librairies Python

CodeCarbon

CodeCarbon [24] est une librairie Python qui peut être utilisée pour mesurer les émissions de CO_2 de la phase d'entraînement et de la phase d'inférence des modèles IA. Il peut être utilisé dans une variété de contextes, y compris les environnements de développement locaux, les instances de Cloud Computing et les clusters de calcul haute performance.

Un des inconvénients potentiels de l'utilisation de la librairie CodeCarbon est qu'elle nécessite une certaine expertise technique et une connaissance aisée du développement traditionnel dans le domaine de l'apprentissage automatique, en particulier avec Python. Cela peut constituer un obstacle pour les organisations qui ne disposent pas d'une expertise interne ou qui débutent dans ce domaine de l'IA.

Carbon Tracker

Carbon Tracker [25] est une organisation à but non lucratif basée à Londres, fondée en 2009. Elle se concentre sur la recherche financière et l'analyse des risques liés aux

23. MLCO2
24. CodeCarbon
25. Carbon Tracker

investissements dans les combustibles fossiles, en mettant l'accent sur le concept de "bulle carbone". L'idée principale derrière cette notion est que si le monde prend au sérieux les engagements climatiques, une grande partie des réserves de combustibles fossiles des entreprises ne pourront pas être brûlées, ce qui pourrait conduire à une dépréciation massive de leurs actifs et à des pertes financières importantes. Carbon Tracker produit des rapports et des analyses pour évaluer l'exposition des investisseurs aux risques liés au carbone, et promeut des stratégies d'investissement compatibles avec les objectifs climatiques internationaux, notamment ceux de l'Accord de Paris.

Leur travail est largement reconnu pour son influence sur les marchés financiers et son rôle dans la sensibilisation aux risques financiers liés aux émissions de carbone. Ils ont également contribué à encourager les investisseurs institutionnels à prendre en compte les risques climatiques dans leurs décisions d'investissement.

Experiment Impact Tracker

Experiment Impact Tracker[26] est censé être une méthode simple pour suivre la consommation d'énergie, les émissions de carbone et l'utilisation des calculs d'un système. Il est plutôt destiné aux systèmes Linux équipés de processeurs Intel (avec prise en charge des interfaces RAPL ou powergadget) et de cartes GPU NVIDIA, et il est en mesure d'enregistrer la consommation d'énergie du CPU et du GPU, les informations sur le matériel, les versions des packages python, les informations concernant les émissions de carbone estimées.

Logiciels

Scaphandre

Scaphandre[27] est un logiciel open-source de mesure de la consommation d'énergie d'un serveur informatique, mais aussi des services et applications qu'il exécute. Il est à la fois un outil utilisable en ligne de commande ou en mode service, pour un usage industrialisé. Son usage ne s'arrête pas seulement aux aspects d'infrastructure, il peut également être utilisé dans la phase de développement, de manière à savoir, par exemple, si d'une release à une autre, une application est plus ou moins énergivore.

Energy Scope

EnergyScope[28] est un projet open source dédié à la mesure d'un système énergétique régional. Le modèle optimise la conception et le fonctionnement horaire sur une année pour une année cible. Il propose une optimisation des stratégies d'investissement de l'en-

26. Experiment Impact Tracker
27. Scaphandre
28. EnergyScope

semble d'un système énergétique sur une période de 30 ans ou plus, avec une optimisation de son fonctionnement horaire. Cette approche permet une évaluation complète de l'intégration effective des sources d'énergie renouvelables intermittentes. Le modèle a une formulation concise et efficace, permettant son exécution sur des ordinateurs portables personnels en 15 minutes environ.

PowerAPI

PowerAPI est un ensemble d'outils middleware conçu pour mesurer la consommation énergétique des codes et logiciels. Il s'agit d'un projet visant à promouvoir une informatique plus verte en fournissant des PowerMeters définis par logiciel pour mesurer la consommation d'énergie des programmes ou des applications. Il offre également une API ouverte qui facilite l'intégration et l'optimisation des processus pour les entreprises, avec une documentation complète pour les développeurs. En outre, PowerAPI propose des produits en marque blanche qui peuvent être intégrés facilement dans des applications et plateformes existantes, permettant ainsi d'augmenter les capacités et la valeur pour les clients et les réseaux.

Dans le cadre de nos travaux nous avons fait le choix d'utiliser PowerAPI, une explication plus détaillée de son fonctionnement est fournie dans une partie dédiée.

Dans ce chapitre, nous avons abordé la nécessité de quantifier la consommation énergétique des applications d'IA pour en réduire la consommation énergétique. Nous avons souligné l'importance des benchmarks dans l'évaluation de la performance énergétique des divers processeurs utilisés dans l'IA, tout en notant l'absence actuelle de standards universels pour mesurer cet impact. Nous avons mis en lumière les efforts de certaines organismes pour établir des normes dans la mesure de l'efficacité énergétique des technologies numériques, incluant l'IA. La présentation d'outils spécifiques de mesure de la consommation énergétique par des fabricants comme NVIDIA et Intel a illustré des moyens concrets pour surveiller et optimiser l'efficacité énergétique lors du déploiement d'applications d'IA.

Après avoir souligné le besoin de mesurer et d'optimiser des applications IA, nous nous tournons vers le prochain chapitre qui se concentre sur l'architecture matérielle sous-jacente. Ce chapitre détaillera le rôle des processeurs dans la consommation énergétique des systèmes d'IA, en mettant en lumière non seulement les CPUs et les GPUs traditionnelles mais aussi en examinant des processeurs spécialisés tels que les circuits intégrés spécifiques à une application (ASIC) et les matrices de portes programmables sur site (FPGA). Ces technologies avancées offrent des opportunités significatives pour réduire la consommation énergétique grâce à leur capacité à exécuter des tâches d'IA de manière plus efficace.

Nous aborderons également la éco-conception matérielle et des DNNs, qui visent à

optimiser conjointement le matériel et les algorithmes d'IA pour maximiser l'efficacité énergétique. Cette approche permet de surmonter certaines des limitations des architectures matérielles conventionnelles en adaptant spécifiquement le matériel aux exigences computationnelles des modèles d'IA.

3.6 IA, Architecture Matérielle

L'architecture matérielle pour les applications d'IA est un domaine en constante évolution, visant à fournir des solutions de calcul optimisées pour les charges de travail spécifiques aux IA, telles que le Deep Learning ou le traitement du langage naturel (NLP), périmètre couvert par nos recherches.

3.6.1 Processeurs standard

Les processeurs standard, les CPUs et les GPUs, avec leurs architectures associées, constituent les fondations de l'informatique appliquée à l'IA. Leur conception, basée sur un modèle de calcul centralisé, est optimale pour les opérations de multiplication-accumulation (MAC) qui sont importantes pour les DNNs. Cette section explore comment ces architectures sont utilisées pour maximiser l'efficacité des DNNs [ZDZW20], [NBC+20], [CSAK14], [Kri09], [Kri09], [HCAD+17].

Les opérations MAC multiplient les entrées du modèle par leurs poids respectifs et additionnent ensuite ces produits. L'exécution efficace de ces opérations en parallèle est essentielle pour accélérer le traitement sur les unités de traitement généralistes telles que les CPUs et les GPUs. Deux techniques couramment utilisées pour cette exécution parallèle sont SIMD (Single Instruction, Multiple Data) et SIMT (Single Instruction, Multiple Threads). En utilisant ces techniques de parallélisme, il est possible d'exploiter pleinement les capacités de calcul des processeurs modernes et d'accélérer l'entraînement et l'inférence des DNNs [RGA19], [ABS12], [MKS22].

Les CPUs sont des composants polyvalents qui alimentent la plupart des systèmes informatiques traditionnels. Ils peuvent être utilisés pour l'entraînement et l'inférence des DNNs, mais leur utilisation se fait principalement dans les scénarios d'inférence. Cette utilisation est courante dans les systèmes qui n'ont pas d'accélérateurs d'inférence spécialisés, où les CPUs assurent la prise de décision basée sur des modèles préalablement entraînés.

L'Influence des GPUs sur l'IA

Contrairement aux CPUs, les GPUs sont conçus pour effectuer des calculs intensifs en parallèle, ce qui les rend adaptés à la phase d'entraînement des DNNs. Ils sont devenus le choix prédominant pour l'entraînement de modèles d'apprentissage supervisé, en grande partie en raison de leur capacité à accélérer les calculs matriciels nécessaires pour ajuster les poids du modèle pendant l'entraînement.

L'apparition des GPUs a été déterminante pour le progrès de l'IA, [MV19], en raison de leur aptitude au calcul parallèle, qui rend l'entraînement de modèles plus rapide et plus efficient avec de grandes quantités de données.

L'entraînement et l'inférence des DNNs illustrent parfaitement la complémentarité des CPUs et GPUs. Tandis que les GPUs dominent la phase d'entraînement grâce à leur capacité de calcul parallèle, les CPUs et GPUs peuvent tous les deux être efficaces pour l'inférence, en fonction des besoins spécifiques en performance et en efficacité énergétique.

L'Inférence en Évolution

Bien que les GPUs dominent l'entraînement des DNNs, les accélérateurs d'inférence spécialisés commencent à émerger. Ces accélérateurs sont conçus pour gérer efficacement les tâches d'inférence tout en optimisant la consommation d'énergie. Cela signifie que les CPUs et les GPUs pourraient partager la charge de travail, avec les CPUs s'occupant des tâches générales et les GPUs gérant les opérations d'inférence intensives.

Optimisation des DNNs sur CPU et GPU

L'optimisation des DNNs sur ces plateformes implique l'exploitation de processeurs multicœurs et l'utilisation de techniques avancées pour réduire le nombre d'opérations nécessaires. Des méthodes comme les matrices de Toeplitz, la transformation de Fourier rapide, et l'algorithme de Winograd sont employées pour améliorer les performances avec l'utilisation des CPUs et GPUs standards [CXS+20], [LLN+18], [JZDL18].

3.6.2 Processeurs spécialisés

L'une des approches essentielles pour réduire la consommation énergétique est le développement de matériel moins énergivore et l'accélération matérielle dédiée à l'IA. Le besoin d'accélérer les calculs d'IA tout en minimisant la consommation d'énergie s'explique par la demande croissante de modèles de NLP plus puissants et plus efficaces.

Le développement de puces et de processeurs spécialement conçus pour l'IA vise à

augmenter l'efficacité énergétique des calculs. Contrairement aux GPUs et CPUs traditionnels, les processeurs spécialisés, tels que que les FPGAs (Field-Programmable Gate Arrays) et les ASICs (Application-Specific Integrated Circuits), sont conçus avec une architecture optimisée pour les calculs. Cela se traduit par une meilleure gestion des tâches parallèles et des charges de travail intensives typiques des réseaux de neurones, qui les rendent plus performantes par rapport aux CPUs et GPUs traditionnels et aussi plus efficaces sur le plan énergétique pour les calculs d'IA, réduisant ainsi la réduction de la consommation énergétique associée à l'entraînement et à l'utilisation de modèles d'IA [WGDL21], [AR21].

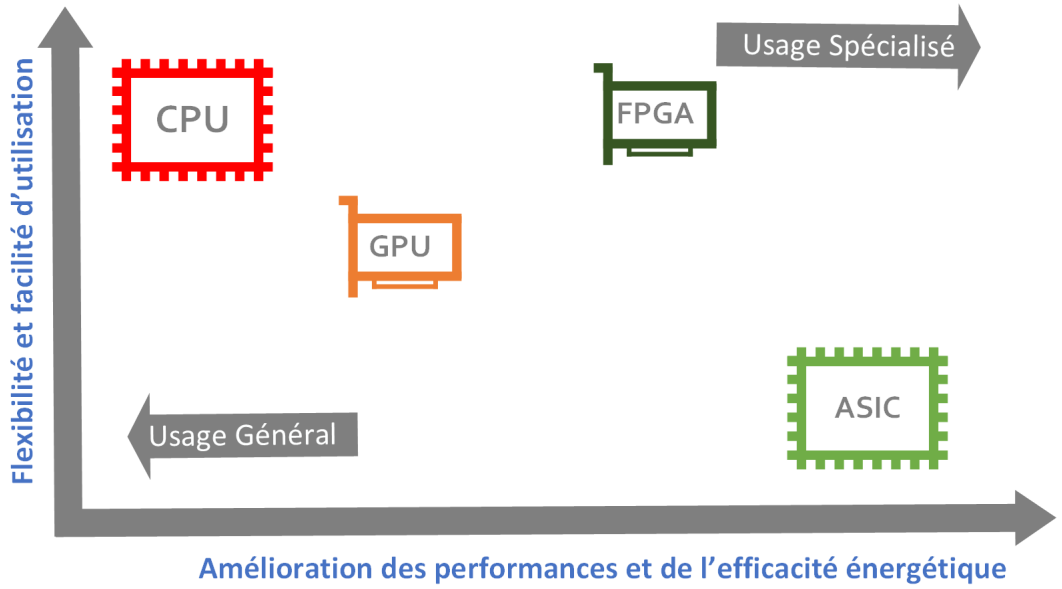

FIGURE 3.9 – Représentation des processeurs : Flexibilité versus Performance [MPJS+14]

La Figure 3.9 représente les différents types de processeurs avec une perspective liée à la facilité d'utilisation vers une meilleure performance possible [MPJS+14]. Ces technologies exploitent la sparsité des modèles et la réduction de la précision pour optimiser à la fois l'entraînement et l'inférence [JD18], [DPY18], [**?**], [VKB18], [PBBP+17], [CBS+17], [EMA+16], [CCZT14], [WLZ20], [JDSM17], [JYPP18], [ESYF+20], [DPY18], [JD18], [MAAI+14], [CLT+19], [SY21], [NDE+18], [GGK+19], [WPY+23].

Les FPGAs sont reconfigurables, ce qui signifie qu'ils peuvent être programmés après la fabrication pour effectuer des tâches spécifiques. Cette flexibilité les rend idéaux pour les applications d'IA en évolution rapide. Ils permettent une utilisation plus efficace des ressources pour des tâches spécifiques, réduisant ainsi la consommation d'énergie globale. Les FPGAs peuvent être optimisés pour des opérations telles que les convolutions et les multiplications matricielles, courantes dans les réseaux de neurones [LYZ+22], [JPRW14], [LLL23], [NYOdlT23], [ZYH+23].

Les ASICs sont conçus pour des applications spécifiques, comme les réseaux de neurones. Les TPUs de Google sont un exemple d'ASIC dédié à l'IA. En étant conçus spécifiquement pour certaines tâches d'IA, les ASICs peuvent offrir une meilleure efficacité énergétique par rapport aux processeurs plus généralistes. Ils offrent des performances élevées pour des tâches spécifiques, étant optimisés pour l'architecture exacte requise par l'application [JYP+17], [CH23].

Optimisation du Matériel L'optimisation du matériel pour mieux gérer les charges de travail est un domaine d'intérêt croissant pour les fabricants de processeurs tels que NVIDIA et Intel. Ces entreprises s'efforcent d'améliorer l'efficacité énergétique de leurs GPU et CPU afin de répondre aux exigences de calcul intensif des applications d'IA.

GPU Spécialisés pour l'IA : NVIDIA développe des GPUs qui sont spécifiquement optimisés pour les tâches d'IA, comme les séries Tesla et RTX. Ces GPU sont conçus pour offrir une puissance de calcul élevée.

Architecture Évolutive : L'architecture des NVIDIA est régulièrement mise à jour pour augmenter l'efficacité énergétique. Par exemple, des architectures telles que Turing et Ampere offrent des améliorations significatives en termes de performances par watt par rapport à leurs prédécesseurs.

Technologies d'Optimisation : NVIDIA intègre également des technologies comme CUDA (Compute Unified Device Architecture) qui permettent une utilisation plus efficace des ressources du GPU pour des tâches d'IA spécifiques.

CPU avec Accélération IA : Intel a intégré des fonctionnalités d'accélération d'IA dans ses CPU, comme les instructions AVX-512 et DL Boost, pour améliorer l'efficacité énergétique lors de l'exécution de calculs d'IA.

Innovation dans les Chipsets : Intel travaille également sur des chipsets spécialisés, tels que les processeurs Nervana et Movidius, qui sont conçus pour des tâches d'IA spécifiques avec une efficacité énergétique améliorée.
Accélérateurs Matériels Spécialisés .

Les **FPGA programmables et reconfigurables** sont essentiels dans l'accélération des réseaux de neurones [VKB19], grâce à leur capacité à opérer au niveau des portes. Ceux-ci sont adaptés aux architectures de réseaux de neurones à faible largeur de bits ou binaires [PBBP+17], car ils permettent des configurations directes des interconnexions entre les portes logiques, favorisant ainsi des calculs efficaces avec des données de faible résolution. Les tables de recherche facilitent l'accélération des fonctions d'activation trigonométriques [CBS+17] et permettent de générer directement des résultats pour des calculs à faible largeur de bits [EMA+16]. En outre, les produits partiels peuvent être conservés dans des registres spéciaux pour être réutilisés [CCZT14], et l'ordonnancement de l'accès à la mémoire, grâce à du matériel d'adressage spécialisé, réduit le nombre de cycles nécessaires pour calculer une sortie de réseau neuronal [JDSM17].

Les FPGAs se révèlent également efficaces pour l'inférence en temps réel dans des systèmes embarqués ou intégrés, offrant une faible latence et une consommation d'énergie réduite par rapport aux processeurs généralistes ou aux GPUs. Ainsi, ils représentent une solution polyvalente et efficace pour une variété d'applications d'IA, permettant une flexibilité de conception essentielle pour expérimenter différentes architectures de réseau [VKB19], [PBBP+17], [CBS+17], [EMA+16], [CCZT14], [JDSM17].

Des **circuits intégrés spécifiques à une application (ASIC)** spécialisés ont également été conçus pour l'accélération des réseaux de neurones. Les ASICs peuvent également cibler à la fois l'entraînement et l'inférence dans les centres de données. Un exemple d'un tel processeur est le Google TPU [JYPP18], [Kan17], qui est un ASIC initialement conçu pour l'inférence d'apprentissage automatique, tandis que les versions plus récentes prennent en charge à la fois l'inférence et l'entraînement [DPY18], [HLK+21], [SP23]. D'autres ASIC ont été conçus pour repousser les limites de l'inférence de réseaux de neurones à faible consommation d'énergie, notamment la calcul neuromorphique [MAAI+14] et l'architecture MIT Eyeriss [CLT+19], qui peuvent bien fonctionner dans des environnements à faible puissance ou à ressources limitées.

Une tendance intéressante à noter est que de nombreux fabricants de matériel, confrontés à des limitations dans les processus de fabrication, ont réussi à exploiter le fait que les réseaux de neurones peuvent bien fonctionner même en utilisant une précision limitée ou mixte [NDE+18] pour la représentation des fonctions d'activation, des poids et des biais. Ces plates-formes matérielles peuvent quantifier les poids et les biais en demi-précision (16 bits) ou même en représentations à un seul bit afin d'augmenter le nombre d'opérations par seconde sans impact significatif sur la prédiction du modèle, la précision ou l'utilisation de la puissance [GGK+19].

L'utilisation d'architectures matérielles spécialisées pour l'IA est motivée par la nécessité d'optimiser les performances des DNNs tout en économisant de l'énergie, argumentée par John Hennessy et Dave Patterson dans leur conférence Turing de 2018 [JD18]. Ils ont mis en avant plusieurs raisons pour lesquelles les architectures matérielles spécialisées sont importantes pour l'avenir de l'IA :

— **Efficacité énergétique** : Les réseaux de neurones profonds sont gourmands en énergie, spécialement lors de l'entraînement sur de grands ensembles de données. Les architectures matérielles spécialisées peuvent être conçues pour minimiser la consommation d'énergie tout en maximisant les performances, offrant ainsi des avantages significatifs en termes d'efficacité énergétique.

— **Performances spécifiques à l'application** : Les architectures spécialisées peuvent être optimisées pour les tâches spécifiques nécessaires à l'IA, telles que les opérations de calcul matricielles utilisées dans les DNNs . Cela permet d'accélérer l'exécution des modèles et d'améliorer les performances globales de l'application.

— **Fine-Tuning du modèle** : Affiner le modèle sur des données d'entraînement spécifiques à la tâche, en utilisant un algorithme d'apprentissage supervisé tel que la descente de gradient ou Adam. Cette étape adapte le modèle pré-entraîné aux spécificités de notre ensemble de données de classification.

— **Adaptabilité** : Les architectures matérielles spécialisées peuvent être conçues pour être hautement adaptatives aux besoins changeants des applications d'IA, ce qui permet une évolutivité et une flexibilité accrues par rapport aux architectures généralistes.

Globalement, ils ont souligné que les architectures matérielles spécialisées sont essentielles pour répondre aux exigences croissantes en termes de performances et d'efficacité énergétique dans le domaine de l'IA, et que leur développement continu est important pour l'avenir de la discipline.

Des accélérations matérielles spécialisées ont été développées pour améliorer les performances des DNNs, en exploitant notamment les zéros dans les modèles. Un bref aperçu de ces accélérateurs est présenté dans les travaux de [SCYE17], [RMJ+20], [JA21], [MSH+21] et [BDG+22], d'où ont peut retenir quelques-unes des principales façons dont ces accélérations matérielles peuvent contribuer à l'amélioration des performances des DNNs :

— **Calculs spécialisés** : Les accélérations matérielles peuvent être conçues pour exécuter efficacement les opérations couramment utilisées dans les DNNs, telles que les opérations de convolution, de pooling, et les calculs matriciels. En optimisant ces opérations, les accélérations matérielles peuvent accélérer considérablement l'exécution des modèles.
— **Parallélisme massif** : Les accélérations matérielles peuvent exploiter le parallélisme massif pour exécuter simultanément de nombreuses opérations en parallèle. Cela est efficace pour les tâches parallèles telles que celles rencontrées dans les DNNs, où de nombreuses opérations peuvent être effectuées en même temps.
— **Optimisations pour la mémoire** : Les accélérations matérielles peuvent inclure des optimisations spécifiques pour gérer efficacement les accès à la mémoire, tels que la mise en cache de données fréquemment utilisées et la minimisation des transferts de données entre la mémoire et le processeur. Cela permet d'améliorer l'efficacité et la vitesse des calculs.
— **Flexibilité et programmabilité** : Certaines accélérations matérielles sont conçues pour être flexibles et programmables, ce qui permet aux développeurs d'implémenter et de tester rapidement de nouveaux algorithmes et architectures de réseaux de neurones. Cela favorise l'innovation et la recherche dans le domaine de l'IA.

En combinant ces différentes techniques, les accélérations matérielles spécialisées peuvent considérablement accélérer l'entraînement et l'inférence des DNNs, ce qui permet de réaliser des avancées significatives dans de nombreux domaines d'application de l'IA.

La réduction de la précision des poids des DNNs peut entraîner des motifs de poids irréguliers, ce qui pose des défis lors de l'implémentation sur des GPUs conçus pour des calculs matriciels denses. Implémenter des opérations MAC avec de tels motifs de poids irréguliers sur des plateformes matérielles à conceptions régulières, comme les GPUs, peut entraîner une sous-utilisation des ressources matérielles. Par conséquent, les modèles épars ne peuvent pas obtenir d'accélération significative sur un matériel comme GPU conçu pour des calculs matriciels denses.

Vers l'Utilisation Potentielle des Architectures de Traitement en Mémoire (PIM)

Les architectures de type PIM (Processing-in-Memory) sont explorées comme une solution pour surmonter les limitations de la mémoire et améliorer l'efficacité du traitement des DNNs. Au cours des deux dernières décennies, les architectures de traitement en mémoire (PIM) ont été largement explorées par les travaux existants [JSRR21], [AHPF18], [CHMH20] comme un moyen potentiel de résoudre le défi du mur de la mémoire.

Co-conception de Modèles DNN Épars et d'Accélérateurs

La co-conception de modèles DNN et d'accélérateurs est une stratégie qui optimise simultanément la structure des DNNs et l'architecture des accélérateurs matériels, représentation dans la Figure 3.10.

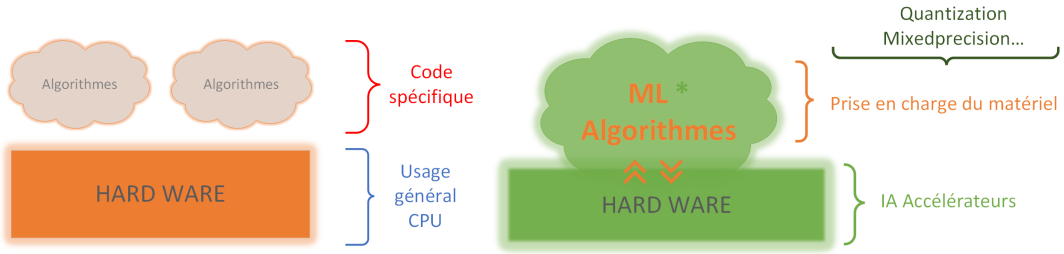

FIGURE 3.10 – Représentation de Co-design hardware aware

Cette approche vise à réduire la complexité et les coûts associés à l'exécution des modèles DNN, tout en préservant ou améliorant leur efficacité et leurs performances. En tenant compte des contraintes des accélérateurs matériels dès les premières étapes de conception des modèles DNN, il est possible d'atteindre une harmonisation optimale

entre l'architecture logicielle et matérielle, conduisant à des gains significatifs en termes d'efficience énergétique et de vitesse de calcul.

La co-conception des modèles DNN épars présente plusieurs avantages :

- permet d'adapter les modèles DNN aux besoins spécifiques des applications et des plateformes,
- réduit les risques de surapprentissage et
- améliore la robustesse des modèles DNN,
- facilite le déploiement et l'exécution des modèles DNN sur des dispositifs à faible puissance ou à mémoire limitée et
- favorise l'innovation et la créativité dans la conception des modèles DNN [RQD+23], [HMD16], [SCYE17], [KJP21], [PK23].

Les DNNs épars sont adaptés à l'utilisation d'accélérateurs matériels spécialisés. En co-concevant la conception des DNNs avec les caractéristiques uniques de ces accélérateurs, on peut tirer parti des caractéristiques spécifiques de l'accélérateur, comme par exemple les unités de traitement tensoriel (TPU), pour maximiser les performances et cela grâce à la puissance de calcul et à la parallélisation des accélérateurs matériels, mais aussi d'accroître la précision des modèles DNN. L'optimisation peut inclure des techniques de quantification et de compression de modèle pour réduire les exigences en termes de mémoire et de bande passante. Grâce à la compacité et à la simplification des modèles DNN épars il est possible de réduire la consommation d'énergie et de la mémoire des systèmes embarqués par exemple.

À ce jour, de nombreux architectes ont proposé diverses conceptions d'accélérateurs spécialisés [HLM+17],[GKK22] pour exploiter les zéros dans le modèle afin d'accélérer la latence ou la modification du matériel existant [ZZGX19]. [GHL+20] proposent une méthode de réduction co-conçue algorithme-logiciel qui permet d'obtenir des accélérateurs de latence sur des architectures denses existantes. Ils proposent un nouvel algorithme capable d'accélérer les modèles DNN épars sur des accélérateurs DNN courants sans modification matérielle.

Ils mettent en œuvre et évaluent le modèle de sparisté sur le cœur de tenseurs GPU, obtenant un gain de vitesse de 1,95 X par rapport aux unités de calcul en virgule flottante qui consomment plus d'énergie et prennent plus de temps pour effectuer des calculs par rapport aux unités de calcul en entiers. Par conséquent, les architectures à faible largeur de bits sont conçues pour accélérer les calculs [MGH+20]. La quantification uniforme est actuellement la méthode utilisée en raison de sa simplicité et de son efficacité de mappage vers le matériel. Les réseaux quantifiés avec moins de 8 bits de précision sont généralement mis en œuvre dans les FPGAs mais peuvent également être exécutés sur des processeurs généraux.

Nous avons abordé les fondations de l'architecture matérielle nécessaire à l'IA : des processeurs CPU et GPU. Ces unités de traitement permettent d'effectuer les opérations de multiplication-accumulation critiques pour le fonctionnement des DNNs. Nous avons également exploré comment ces architectures maximisent l'efficacité des DNNs à travers des opérations parallèles et des techniques avancées d'optimisation. Ensuite, nous nous sommes penchés sur le développement de matériel moins énergivore et les accélérateurs matériels dédiés à l'IA, révélant les stratégies et les avancées dans ce domaine. Nous avons discuté de l'importance des processeurs spécialisés, tels que les FPGAs et les ASICs, conçus spécifiquement pour les calculs d'IA, et comment ces technologies contribuent à réduire la consommation énergétique et l'empreinte carbone des modèles de NLP. Nous avons vu comment l'optimisation du matériel, menée par des géants de l'industrie comme NVIDIA et Intel, conduit à des avancées significatives en termes de performances par watt, favorisant une utilisation plus responsable de l'énergie dans le domaine de l'IA.

La prochaine section se concentre sur la parallélisation des calculs qui permettent d'améliorer davantage l'efficacité et la vitesse de traitement des tâches d'IA. Nous examinerons les stratégies et technologies actuelles qui facilitent la parallélisation, y compris les approches logicielles et matérielles qui rendent possible l'exécution simultanée de multiples opérations.

3.7 Parallélisation des Calculs, les avancées

La parallélisation des calculs est une approche essentielle pour accélérer les tâches d'IA, en spécial celles impliquant des DNNs qui exigent d'effectuer un grand nombre de calculs simultanément. Elle permette une exécution plus rapide et plus économe en énergie des tâches d'IA.

Nous allons citer quelques-unes des avancées récentes dans ce domaine.

Multithreading sur CPU

Le multithreading, une forme de parallélisation qui permet à un CPU de traiter plusieurs threads d'exécution simultanément, est essentiel pour accroître l'efficacité des calculs sur des architectures CPU. Cette technique peut réduire le temps nécessaire pour exécuter des tâches d'IA en distribuant la charge de travail sur plusieurs cœurs du processeur. En plus du temps de calcul il peut augmenter l'utilisation des ressources, et améliorer la performance des modèles de DL. Il peut aussi être plus accessible et moins coûteux que l'utilisation de GPU, qui sont souvent considérées comme le standard pour les applications IA et spécialement dans le domaine de DL.

Des bibliothèques comme OpenMP (Open Multi-Processing) facilitent l'implémenta-

tion du multithreading en permettant aux développeurs de paralléliser facilement leurs applications avec des directives préprocesseur simples [DM98], [Cha02], [BBB+22], [SPWL23].

Un point d'attention par rapport à cette technique est le fait que le multithreading sur CPU peut introduire de la non-déterminisme, c'est-à-dire que les résultats peuvent varier selon les conditions d'exécution. Cela peut affecter la reproductibilité, la fiabilité et la robustesse des modèles de Deep Learning, surtout pour les applications critiques. Il peut aussi être limité par la puissance, la mémoire et le nombre de cœurs du CPU.

Calcul Parallèle avec GPU

Les processeurs de type GPU, avec leur architecture parallèle, sont adaptés au calcul massivement parallèle nécessaire dans les DNNs. L'utilisation des processeurs GPU permettent d'exécuter des milliers de threads simultanément, ce qui les rend idéaux pour les opérations de matrice et de vecteur dominantes dans les DNNs. Dans le domaine de l'IA le framework CUDA (Compute Unified Device Architecture) de NVIDIA est le framework le plus utilisé et permet une parallélisation efficace et une utilisation optimale des ressources GPU pour l'accélération des calculs d'IA.

Traitement Distribué

Le traitement distribué, avec, par exemple le framework open source Apache Spark, permet l'exécution de tâches de calcul intensives sur des clusters de serveurs, optimisant ainsi les ressources et réduisant les délais. Apache Spark est reconnu pour sa capacité à traiter de grands ensembles de données en parallèle sur un nombre conséquent de nœuds, en utilisant des abstractions de haut niveau comme les RDDs (Resilient Distributed Datasets). Cette approche est avantageuse pour les tâches d'IA qui nécessitent l'analyse de grandes quantités de données, car elle minimise la latence et maximise le débit.

Nous avons exploré les méthodologies qui sous-tendent la parallélisation des calculs. Du multithreading sur CPU, qui optimise l'utilisation des ressources processeur, au calcul parallèle avec GPU, qui exploite les capacités de traitement massivement parallèle pour accélérer les opérations de réseau de neurones, jusqu'au traitement distribué qui permet une gestion et une analyse efficaces de vastes ensembles de données sur des clusters de serveurs.

3.8 Indicateurs de l'Efficacité Énergétique

Il est important de mesurer des paramètres que l'on devrait considérer lors de la comparaison et de l'évaluation des forces et des faiblesses des différentes conceptions

et propositions techniques et qui devraient être incorporées dans les considérations de conception. Cette démarche vise à enrichir les réflexions en matière de conception et à favoriser une prise de décision éclairée. Dans ce sens, comme l'explique [LHFT20], "l'efficacité d'un modèle de DNN doit être évaluée à travers un prisme multidimensionnel qui intègre des aspects allant au-delà des FLOPS/W ou TOPS/W, afin de prendre en compte les besoins spécifiques de chaque application et de chaque scénario d'utilisation."

3.8.1 Les Unités de Mesures de Consommation Énergétique

Watts et Wattheures

La consommation énergétique s est souvent mesurée en watts (W), représentant la puissance instantanée, et en wattheures (Wh), reflétant l'énergie consommée sur une période. Ces mesures sont nécessaires pour comprendre l'impact énergétique des opérations de calcul, notamment lors de phases intensives telles que l'entraînement des modèles.

Joules

Le joule, J, est une autre unité pour mesurer l'efficacité énergétique, utile pour évaluer l'énergie requise pour effectuer une opération spécifique ou traiter une donnée. L'efficacité énergétique, exprimée en opérations par joule, permet de comparer directement l'impact énergétique de différentes architectures ou optimisations.

3.8.2 Indicateurs de Performance et Efficacité

Précision (Accuracy)

La précision du modèle, appelée accuracy en anglais, doit être évaluée par rapport à l'efficacité énergétique. Un modèle très précis mais énergivore pourrait ne pas être viable pour certaines applications, surtout les applications embarquées fonctionnant sur des appareils alimentés par batterie [SGM20]

Elle mesure la proportion des prédictions qui sont correctes par rapport à l'ensemble des prédictions. La formule pour calculer l'accuracy est la suivante :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}} \tag{3.12}$$

Cela peut être exprimée de la manière suivante :

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.13}$$

où :

— TP (True Positives) est le nombre de vrais positifs, c'est-à-dire le nombre d'instances positives correctement identifiées par le modèle.

— TN (True Negatives) est le nombre de vrais négatifs, c'est-à-dire le nombre d'instances négatives correctement identifiées par le modèle.

— FP (False Positives) est le nombre de faux positifs, c'est-à-dire le nombre d'instances négatives incorrectement identifiées comme positives par le modèle.

— F (False Negatives) est le nombre de faux négatifs, c'est-à-dire le nombre d'instances positives incorrectement identifiées comme négatives par le modèle.

La précision est une mesure utile pour les ensembles de données où les classes sont à peu près équilibrées. Cependant, dans des situations où il y a un déséquilibre significatif entre les classes (quand une classe est beaucoup plus fréquente que l'autre), d'autres mesures comme la précision, le rappel (recall) et le score F1 peuvent fournir des informations plus utiles sur la performance du modèle.

Précision (Precision) : La précision est le nombre de vrais positifs (TP) divisé par le nombre de toutes les prédictions positives (vrais positifs TP plus faux positifs FP).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.14}$$

Rappel (Recall) ou Sensibilité : Le rappel est le nombre de vrais positifs divisé par le nombre de toutes les instances positives réelles (vrais positifs TP plus faux négatifs FN).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.15}$$

F1-score : Le F1-score est la moyenne harmonique de la précision et du rappel. Contrairement à une moyenne arithmétique simple, la moyenne harmonique pénalise les écarts extrêmes, rendant le F1-score plus robuste dans les cas où il y a un grand déséquilibre entre la précision et le rappel.

$$\text{F1} = 2 \times \frac{\textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}} \tag{3.16}$$

Le F1-score atteint sa meilleure valeur à 1 (performance parfaite) et sa pire à 0 en etant utile en cas de besoin d'un équilibre entre la précision et le rappel dans le cadre des classes déséquilibrées (par exemple, détection de fraude).

Performance du Système

La performance d'un système dédié au traitement de DNN peut être évaluée selon plusieurs critères, notamment le débit, la latence, l'efficacité énergétique, et le coût du matériel. Chacun de ces aspects est important dans la détermination de la viabilité et de l'efficacité globale du système pour des applications spécifiques, notamment dans des contextes exigeant des réponses en temps réel et une gestion optimale de la consommation d'énergie.

Débit et Latence

Le *débit* d'un système est défini comme le nombre d'opérations ou de tâches qu'il peut accomplir par unité de temps. Il est généralement mesuré en opérations par seconde. La formule pour calculer le débit est la suivante :

$$\text{Débit} = \frac{\text{Nombre total d'opérations}}{\text{Temps total}} \tag{3.17}$$

La *latence*, d'autre part, représente le temps nécessaire pour qu'une tâche unique soit traitée par le système, depuis son lancement jusqu'à l'obtention du résultat. Elle est importante pour les applications nécessitant des réponses immédiates. La latence peut être exprimée comme :

$$\text{Latence} = \text{Temps de réponse du système} \tag{3.18}$$

L'amélioration du débit peut parfois se faire au détriment de la latence, et vice-versa, selon la conception du système et les optimisations appliquées.

Efficacité Énergétique et Consommation d'Énergie

L'*efficacité énergétique* d'un système se mesure en opérations par joule (op/J), indiquant le nombre d'opérations effectuées pour chaque unité d'énergie consommée. Une efficacité élevée signifie une meilleure performance par unité d'énergie, ce qui est essentiel pour réduire les coûts opérationnels et l'impact environnemental.

$$\text{Efficacité Énergétique} = \frac{\text{Nombre d'opérations}}{\text{Énergie consommée (en Joules)}} \tag{3.19}$$

Coût du Matériel

Le coût du matériel nécessaire pour exécuter des DNNs inclut non seulement l'achat initial des composants, mais aussi les coûts liés à l'exploitation et à la maintenance. Les accélérateurs matériels spécialisés, tels que les ASICs et les FPGAs, peuvent offrir d'excellentes performances et efficacité énergétique mais peuvent entraîner des coûts initiaux et de maintenance élevés.

Le coût total du matériel peut être considéré comme :

$$\text{Coût Total du Matériel} = \text{Coût d'Achat} + \text{Coût d'Exploitation} + \text{Coût de Maintenance} \tag{3.20}$$

La sélection et l'optimisation du matériel doivent donc tenir compte de tous ces facteurs pour atteindre un équilibre optimal entre performance, efficacité énergétique et coût.

Compromis entre Consommation Énergétique et Performance

Le compromis entre consommation d'énergie et performance nécessite de trouver un équilibre optimal entre la diminution de la performance d'un modèle et l'amélioration de son efficacité énergétique. Cette balance revêt une importance particulière dans les environnements où l'énergie est une ressource limitée, comme dans le cas des appareils mobiles ou embarqués. Bien qu'un modèle d'IA très précis puisse fournir des résultats supérieurs, sa consommation élevée d'énergie peut le rendre peu pratique pour les dispositifs alimentés par batterie, où la conservation de l'énergie est primordiale. Inversement, un modèle moins exigeant en termes de précision, mais plus économe en énergie, peut s'avérer plus adapté dans de telles circonstances.

Nous avons abordé plusieurs indicateurs pour évaluer l'efficacité des DNNs, mettant l'accent sur l'équilibre entre la consommation énergétique et la performance. Ces indicateurs comprennent la précision (accuracy), qui mesure la justesse des prédictions du modèle, la précision et le rappel pour évaluer la qualité des classifications. Nous avons également discuté de l'importance du débit et de la latence pour la performance du système, l'efficacité énergétique exprimée en opérations par joule, et le coût du matériel. Ces mesures fournissent une vue d'ensemble pour optimiser les DNNs, permettant de faire des choix informés pour développer des modèles efficaces tout en étant économes en énergie, adaptés aux exigences des applications modernes d'IA.

Dans la section suivante nous explorerons comment l'Edge Computing offre une solution prometteuse face aux défis de latence, consommation d'énergie, et sécurité des données rencontrés par le Cloud Computing traditionnel.

3.9 Edge versus Cloud

Alors que le traitement centralisé dans le Cloud a longtemps été la norme, cette approche rencontre des limites en termes de latence, de consommation d'énergie et de sécurité des données. En réponse à ces défis, l'Edge Computing émerge comme une solution prometteuse, proposant une nouvelle manière de traiter les données directement là où elles sont générées. Cette tendance vers le calcul en périphérie, ou "on the Edge", illustre

une évolution dans la gestion des données et le traitement informatique, répondant aux besoins croissants de réactivité et d'efficience dans un monde connecté.

L'Edge Computing représente une évolution importante dans la façon dont les données sont traitées et gérées. Il s'agit de déplacer la puissance de calcul nécessaire au traitement des données vers la périphérie du réseau, éloignant ainsi la dépendance du traitement centralisé dans le cloud. Cette approche révolutionnaire permet de réduire les délais de traitement en traitant les données localement, au plus près de leur source et de l'utilisateur final. Par conséquent, les applications et les services peuvent fonctionner plus rapidement et de manière plus efficace, améliorant ainsi l'expérience utilisateur et la réactivité des systèmes.

En adoptant l'Edge Computing, les entreprises peuvent également améliorer la confidentialité et la sécurité des données en réduisant les risques associés à la transmission de données sensibles vers des serveurs distants. De plus, cette approche offre une grande flexibilité en permettant le déploiement de serveurs ou même de mini-centres de données dans des environnements divers tels que des usines, des véhicules autonomes, des stations météorologiques ou des bâtiments intelligents. Cette proximité physique avec les données générées par les capteurs connectés permet une analyse en temps réel et des prises de décision plus rapides.

Pourquoi le Edge prend une place importante ?

- *Communication* : il y des lieux sur la planete où la communication n'est pas très développée [MYZ+17].
- Les *données collectées qui sont sensibles* : la sécurité et les données privées sont critiques [CWZZ17].
- Le *temps de réponse* nécessaire est très important : pour la robotique, voitures autonommes [HZW15].
- Une autre raison pour travailler on Edge est la *puissance électrique* [SCZ+16].

L'Edge Computing offre des solutions innovantes pour répondre aux défis croissants de la connectivité, de la vitesse et de la sécurité des données dans un monde de plus en plus interconnecté. Il gère également de manière plus frugale les ressources énergétiques [ESSP+17].

Nous avons exploré le contraste entre le traitement centralisé dans le Cloud et l'approche émergente de l'Edge Computing, mettant en lumière les avantages et les défis de chaque paradigme. L'Edge Computing, avec son principe de traitement des données à la source, se présente comme une réponse efficace aux limitations du Cloud en termes de latence, de consommation énergétique et de sécurité des données. Cette approche décentralisée promet une amélioration de la réactivité et de l'efficience énergétique.

3.10 Conclusion

Dans un premier temps, nous avons délimité le périmètre de l'IA. Nous avons ensuite analysé les raisons sous-jacentes à la consommation élevée des réseaux de neurones actuels, notamment en raison de leur taille importante et du fait que le transfert de données de la DRAM vers le processeur engendre une consommation significative. Nous avons également exploré les méthodes de compression pour les algorithmes de NLP ces derniers étant très énergivores de nos jours. Poursuivant notre exploration, nous avons présenté les processeurs d'usage général tels que les CPUs et GPUs, ainsi que les processeurs spécialisés comme les FPGAs et les ASICs. Il a été souligné que les processeurs spécialisés tendent à consommer moins d'énergie en raison de leur capacité à être optimisés pour des tâches spécifiques, offrant ainsi une efficacité énergétique supérieure par rapport aux processeurs d'usage général. Cette distinction a mis en évidence l'importance de la co-conception des DNNs et des accélérateurs matériels. Nous avons ensuite abordé la question de la mesure et de l'évaluation de l'impact énergétique des applications d'IA. Cette section a mis en avant les différentes méthodologies pour quantifier la consommation énergétique, soulignant l'importance de développer des métriques précises pour évaluer l'efficience énergétique des systèmes d'IA. Enfin, les indicateurs de performance d'un système d'IA ont été examinés. Ces indicateurs, allant de la précision et la rapidité d'inférence à l'efficacité énergétique fournissent une connaissance approfondie de la performance d'un système d'IA.

Synthèse

→ La croissance exponentielle du nombre des paramètres dans les modèles IA reste un défi.

→ L'utilisation des méthodes de compression et des approches pour leur mise en œuvre sont essentielles.

→ Le développement de matériel moins énergivore et l'accélération matérielle dédiée à l'IA reste une alternative.

→ La co-conception de DNN et de processeurs peut contribuer à diminuer cette consommation.

→ La parallélisation des calculs représente une vrai stratégie pour réduire l'empreinte énergétique.

→ La mesure de la consommation énergétique s'avère être un prérequis indispensable à l'amélioration de l'efficience des systèmes d'IA.

→ Trouver le meilleur compromis entre performance et consommation énergétique.

Chapitre 4

FlauBERT et Techniques de Compression

Sommaire

Ce chapithe se tourne vers le traitement du langage naturel avec FlauBERT, un modèle conçu spécifiquement pour la langue française. Cette analyse approfondie de FlauBERT, depuis son architecture jusqu'aux techniques de compression, vise à optimiser son efficacité tout en maintenant des performances élevées, préparant le terrain pour une exploration pratique dans le chapitre suivant.

4.1 Introduction

FlauBERT [LVF+20] représente une avancée significative dans le domaine du traitement automatique du langage naturel (NLP) pour la langue française, tirant parti des architectures de modèle de langage pré-entraînées pour améliorer les performances sur une gamme de tâches NLP. Inspiré par le succès de BERT [1] et d'autres modèles basés sur les Transformers, FlauBERT a été spécifiquement formé sur un large corpus de texte français hétérogène, utilisant le superordinateur Jean Zay du CNRS. Ce modèle se distingue par sa capacité à comprendre et traiter les nuances complexes de la langue française, démontrant ainsi une amélioration notable par rapport aux approches de pré-entraînement antérieures dans plusieurs tâches NLP, notamment la classification de texte, la paraphrase, l'inférence de langue naturelle, l'analyse syntaxique, la désambiguïsation du sens des mots [DCLT18].

L'architecture de FlauBERT s'appuie sur des couches d'encodeurs Transformer empilées, intégrant des mécanismes d'auto-attention et des réseaux neuronaux feed-forward (FFNs) pour traiter les données d'entrée. Ces éléments permettent au modèle d'apprendre des représentations contextuelles des mots au niveau de la phrase, offrant une base solide pour le fine-tuning sur des tâches spécifiques en aval. Avec des versions de différentes tailles adaptées aux besoins variés de la recherche et de l'application pratique, FlauBERT est mis à disposition de la communauté scientifique, accompagné d'un protocole d'évaluation unifié pour les tâches en aval, nommé FLUE (French Language Understanding Evaluation). Ce cadre vise à faciliter des expériences reproductibles dans le domaine du NLP en français, encourageant ainsi l'avancement et l'innovation continue dans le traitement des langues moins ressources comme le français.

FlaubBERT, comme tous les modèles de NLP, connaît une tendance à l'augmentation de la taille car il est conçu pour mieux saisir et reproduire les nuances linguistiques spécifiques au français. La recherche incessante de capacités linguistiques améliorées a stimulé la croissance des modèles en NLP [BMR+20], [RWC+19]. Dans ce contexte, le déploiement et la maintenance de modèles comme FlauBERT, gourmands en ressources, présentent des défis importants comme : le coûts de calcul, la latence et l'impact environnemental.

Les techniques de compression de modèle, telles que la factorisation à rang réduit, l'intégration de Kronecker et le regroupement des matrices de poids, se distinguent comme des méthodes pour réduire la taille des modèles, sans compromettre significativement leur précision, ce qui est essentiel pour une utilisation efficace des ressources.

1. Acronyme anglais de Bidirectional Encoder Representations from Transformers, est un modèle de langage développé par Google en 2018. Cette méthode a permis d'améliorer significativement les performances en traitement automatique des langues et il est connu pour son aptitude à capturer les contextes bidirectionnels des mots dans une phrase, ce qui lui permet de mieux comprendre le sens et la structure des textes.

4.2 FlauBERT pour la classification de texte

4.2.1 Architecture de FlauBERT

FlauBERT tire parti de l'architecture des transformers [VSP+17], qui a révolutionnée les approches de modélisation dans le NLP, représentation dans la Figure 4.1.

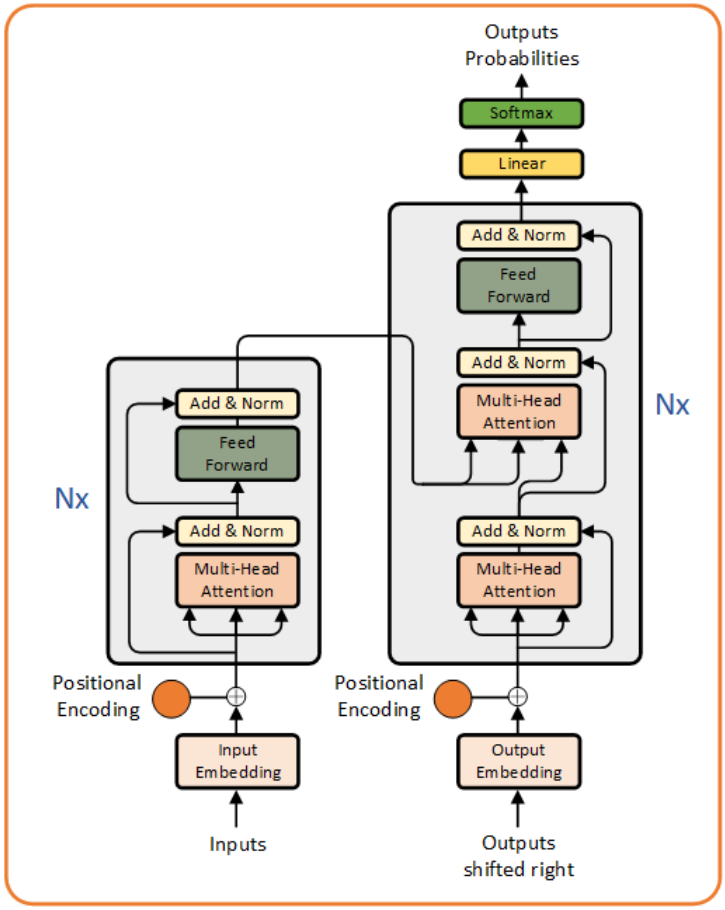

FIGURE 4.1 – La structure encodeur-décodeur de l'architecture Transformer [VSP+17]

Cette architecture se distingue principalement par son utilisation des mécanismes d'attention, permettant au modèle de se concentrer sur différentes parties d'une séquence de texte pour en extraire les informations pertinentes sans dépendre d'une structure séquentielle rigide.

Les composants principales de cette architecture incluent :

— **Mécanismes d'auto-attention** : Permettent à chaque mot dans la séquence d'entrée d'interagir directement avec tous les autres mots, aidant le modèle à comprendre le contexte et les relations entre les mots indépendamment de leur distance dans la séquence.

— **Réseaux neuronaux feed-forward** : Chaque couche d'encodeur dans l'architecture Transformer comprend un réseau de neurones feed-forward qui applique des transformations supplémentaires aux données traitées par l'auto-attention. Ces FFNs sont essentiels pour permettre à chaque couche d'encoder des informations plus complexes et abstraites à partir des données d'entrée.

Au cœur de l'architecture de FlauBERT se trouvent les **couches d'encodeurs**, qui sont empilées les unes sur les autres pour former le modèle. Chaque couche d'encodeur est composée d'un mécanisme d'auto-attention suivi d'un réseau de neurones feed-forward, avec des connexions résiduelles et une normalisation de couche pour faciliter l'apprentissage profond.

Couches d'Encodeurs

Les couches d'encodeurs dans FlauBERT sont organisées de manière séquentielle, avec chaque couche recevant l'entrée de la couche précédente et passant sa sortie à la couche suivante. Mathématiquement, l'opération d'une couche d'encodeur peut être représentée comme suit :

$$\text{Enc}_i(\text{input}_i) = \text{FFN}(\text{MultiHeadAttention}(\text{input}_i)) \tag{4.1}$$

où

— Enc_i est la i-ème couche d'encodeur,

— input_i est l'entrée pour cette couche,

— MultiHeadAttention est la fonction d'attention multi-tête, et FFN est le réseau de neurones feed-forward.

Mécanismes d'Auto-Attention

Les couches d'attention [VSP+17], ont révolutionné la manière dont les modèles de NLP appréhendent les séquences de données. Leur capacité à modéliser des dépendances à longue distance sans se limiter à une structure séquentielle fixe permet une flexibilité et une efficacité améliorées dans la modélisation des relations contextuelles.

Les caractéristiques principales sont :

— ***Têtes d'Attention*** : BERT, et par extension FlauBERT, utilise plusieurs têtes d'attention dans chaque couche de son réseau. Ces têtes d'attention permettent au modèle de se concentrer simultanément sur différentes parties de la séquence d'entrée, lui permettant ainsi de capter divers types de relations et de dépendances dans les données. Le nombre exact de têtes d'attention peut varier en fonction de la variante spécifique de FlauBERT. Chaque couche d'attention à têtes multiples comprend des transformations linéaires, représentées par "q lin"(requête), "k lin"(clé), "v lin"(valeur) et "out lin" en contribuant au calcul des scores d'attention et de la sortie finale [VSP+17].

$$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{4.2}$$

où W_i^Q, W_i^K, et W_i^V sont des matrices de poids spécifiques à chaque tête d'attention pour les requêtes, les clés et les valeurs, respectivement. L'attention combinée de toutes les têtes est ensuite calculée par :

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{Head}_1, \text{Head}_2, \ldots, \text{Head}_h)W^O \tag{4.3}$$

où h est le nombre de têtes d'attention, et W^O est une matrice de poids qui combine les sorties de toutes les têtes d'attention.

— ***Mécanisme d'Attention Détail*** : Le mécanisme d'attention repose sur la formule suivante, qui calcule un poids d'attention pour chaque paire de positions dans la séquence d'entrée :

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{4.4}$$

où $e_{ij} = \frac{q_i k_j^T}{\sqrt{d_k}}$ est le score d'attention non normalisé entre la requête q_i et la clé k_j, et n est le nombre d'éléments dans la séquence. Les scores d'attention α_{ij} sont utilisés pour créer une combinaison linéaire pondérée des vecteurs de valeur, produisant ainsi la sortie de la couche d'attention pour chaque position i :

$$\text{AttentionOutput}i = \sum j = 1^n \alpha_{ij} v_j \tag{4.5}$$

La mécanique de l'attention peut être décrite de la manière suivante :

— Score d'Attention Le score d'attention, qui mesure la pertinence de chaque mot de la séquence par rapport à un mot donné, est calculé par la formule :

$$e_{ij} = \frac{q_i k_j^T}{\sqrt{d_k}} \tag{4.6}$$

où e_{ij} est le score d'attention entre le i-ème mot et le j-ème mot de la séquence. q_i représente le vecteur de requête pour le i-ème mot, k_j le vecteur de clé pour

le j-ème mot, et d_k la dimension de ces vecteurs. Ce score est ensuite normalisé pour garantir que l'attention totale attribuée à travers tous les mots est égale à 1.

— Normalisation et Poids d'Attention Les poids d'attention normalisés sont obtenus en appliquant la fonction softmax aux scores d'attention :

$$\alpha_{ij} = \text{softmax}\,(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{4.7}$$

Ces poids déterminent la proportion d'attention que le modèle attribue à chaque mot lors de la construction de la représentation du mot cible.

— Sortie de l'Attention La sortie de l'opération d'attention pour chaque mot est une somme pondérée des vecteurs de valeur, pondérée par les poids d'attention :

$$\text{AttentionOutput}i = \sum j = 1^{n} \alpha_{ij} v_j \tag{4.8}$$

où v_j est le vecteur de valeur associé au j-ème mot. Cette somme pondérée permet au modèle de synthétiser une représentation qui intègre l'information contextuelle pertinente de toute la séquence.

Formule de Base de l'Attention : La formule d'attention est essentielle dans le calcul de l'attention et permet aux modèles de NLP de traiter dynamiquement les séquences de texte en se concentrant sur les informations pertinentes (illustration dans la Figure 4.2).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{4.9}$$

où :

— Q représente la matrice de requête, qui encode les informations de la requête ou de la question posée :

$$Q = (q_1, q_2, \ldots, q_m) \in \mathbb{R}^{d \times m}$$

avec chaque $q_i \in \mathbb{R}^d$

— K représente la matrice de clé, qui encode l'information dans la séquence d'entrée :

$$K = (k_1, k_2, \ldots, k_n) \in \mathbb{R}^{d \times n}$$

avec chaque $k_i \in \mathbb{R}^d$.

— V représente la matrice de valeur, qui contient les valeurs associées à chaque élément de la séquence d'entrée :

$$V = (v_1, v_2, \ldots, v_n) \in \mathbb{R}^{d \times n}$$

avec chaque $v_i \in \mathbb{R}^d$

— d_k est la dimension de la clé. Il s'agit d'un paramètre de dimensionnalité utilisé pour normaliser le produit scalaire entre les matrices Q et K.

L'opération d'attention calcule les réponses d'un vecteur de requête q_i en lui accordant de l'attention à tous les vecteurs de clé dans K et utilise les résultats pour effectuer une somme pondérée sur les vecteurs de valeur dans V.

La multiplication QK^T calcule la compatibilité entre chaque paire de requête et de clé, produisant un score qui mesure à quel point les éléments de la séquence d'entrée sont pertinents les uns par rapport aux autres. La division par $\sqrt{d_k}$ normalise ces scores pour éviter des gradients extrêmement grands ou petits pendant l'entraînement. La fonction softmax est ensuite appliquée pour transformer ces scores en probabilités d'attention, indiquant à quel point chaque élément de la séquence devrait contribuer à la représentation de sortie. Enfin, ces poids d'attention sont utilisés pour créer une combinaison linéaire des vecteurs de valeur, générant ainsi la sortie pondérée par l'attention.

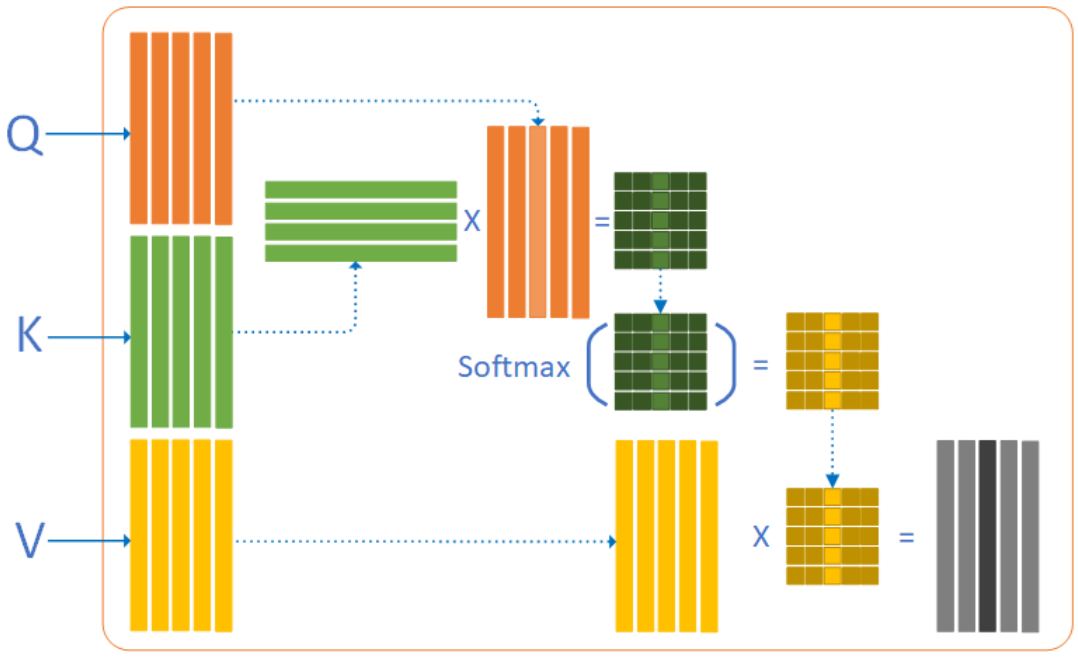

FIGURE 4.2 – Illustration de l'opérateur d'attention. Softmax est l'opérateur softmax par colonne. Q, K et V sont des matrices d'entrée.

Dans le calcul de l'attention, représentation dans la Figure 4.2, la **fonction softmax** est utilisée pour normaliser les scores de similarité calculés entre les vecteurs de requête et les vecteurs de clé.

Plus précisément, supposons que nous ayons calculé les scores de similarité s entre les vecteurs de requête q_i et les vecteurs de clé k_j. Ces scores de similarité sont souvent représentés comme une matrice S où S_ij représente le score de similarité entre le i-ème vecteur de requête et le j-ème vecteur de clé.

La fonction softmax est appliquée à chaque colonne de la matrice S comme suit :

$$\text{softmax}(\mathbf{S})_{ij} = \frac{e^{S_{ij}}}{\sum_{k=1}^{n} e^{S_{ik}}}$$

où softmax(S_ij) est le score softmax normalisé correspondant au i-ème vecteur de requête et au j-ème vecteur de clé.

Le fait d'être appliquée à chaque colonne de la matrice S, ce qui normalise ces scores de manière à ce qu'ils soient tous compris entre 0 et 1, et que leur somme soit égale à 1. Cela permet de transformer les scores de similarité en une distribution de probabilité sur les vecteurs de clé, où les scores les plus élevés ont une probabilité plus élevée d'être sélectionnés lors du calcul de la somme pondérée des vecteurs de valeur.

La fonction softmax est utilisée dans le calcul de l'attention pour transformer les scores de similarité en une distribution de probabilité qui est ensuite utilisée pour pondérer les vecteurs de valeur lors du calcul de la sortie de l'opération d'attention.

Réseaux Neuronaux Feed-Forward

Après la phase d'attention multi-têtes, chaque couche d'encodeur dans le modèle FlauBERT procède à une transformation feed-forward sur les données traitées.
Cette sous-couche, désignée par FFN (Feed-Forward Network), se compose de deux transformations linéaires séparées par une fonction d'activation non-linéaire :

— deux transformations linéaires[2] : Elle est généralement représentée par une matrice de poids W et un vecteur de biais b.Pour une entrée x, la sortie y est calculée comme suit : y=Wx+b. Cette transformation linéaire est effectuée deux fois consécutivement pour chaque couche FFN.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{4.10}$$

où W_1, W_2, b_1, et b_2 sont des paramètres appris par le modèle, et $\max(0, x)$ est la fonction d'activation ReLU appliquée élément par élément.

2. Opération mathématique qui applique une transformation affine aux données d'entrée

Première Couche Linéaire : Cette couche est une couche linéaire entièrement connectée qui prend la sortie du mécanisme d'attention multi-têtes comme entrée. Elle applique une transformation linéaire aux données d'entrée, incluant généralement une multiplication de matrice de poids et l'ajout d'un biais. La sortie de cette couche a une représentation potentiellement de dimension supérieure.

Deuxième Couche Linéaire : Après la fonction d'activation, une autre couche linéaire entièrement connectée est appliquée. Cette couche transforme davantage la sortie de la fonction d'activation, réduisant potentiellement la dimensionnalité ou la mappant à un espace de caractéristiques différent.

Les transformations linéaires sont séparées par :

— une fonction d'activation non linéaire, généralement une fonction ReLU.

Ces FFNs aident à capturer des motifs et des relations complexes dans les données. Ils ajoutent une couche de non-linéarité et permettent au modèle d'apprendre des représentations complexes des données d'entrée, ce qui peut être bénéfique pour diverses tâches de NLP telles que la modélisation du langage, la traduction et la classification de séquences.

La normalisation de couche (Layer Normalization) [3] est ensuite appliquée après les couches d'attention à têtes multiples et après les FFNs pour améliorer la stabilité et l'efficacité de l'entraînement du modèle, réduisant ainsi le décalage de covariance interne et permettant une convergence plus rapide dans les réseaux profonds [BKH16]. Ces couches de normalisation appliquées après les couches d'attention à têtes multiples capturent les dépendances entre les jetons, en les normalisant avant d'être transmises aux couches suivantes (telles que les réseaux feedforward) dans l'architecture du réseau.

Pour l'entraînement de FlauBERT Large, l'attention pré-norme et les profondeurs stochastiques ont été appliqués. Dans une mise à jour de l'implémentation du Transformer [VSP+17], la normalisation des couches est appliquée avant chaque couche d'attention par défaut. Il a été observé par [VSP+17], que la pré-normalisation aide à stabiliser l'entraînement.

Variantes de FlauBERT

Il peut y avoir différentes variantes de FlauBERT, comme Base ou Large (illustration

3. La normalisation de couche ajuste et met à l'échelle les activations de manière à ce qu'elles aient une moyenne et une variance spécifiques (généralement, une moyenne de 0 et une variance de 1), mais pour chaque exemple individuellement et à travers les features plutôt qu'à travers le batch entier (comme dans la normalisation par batch) et permet de stabiliser les activations à travers le réseau, ce qui peut conduire à une convergence plus rapide pendant l'entraînement et améliorer les performances du modèle.

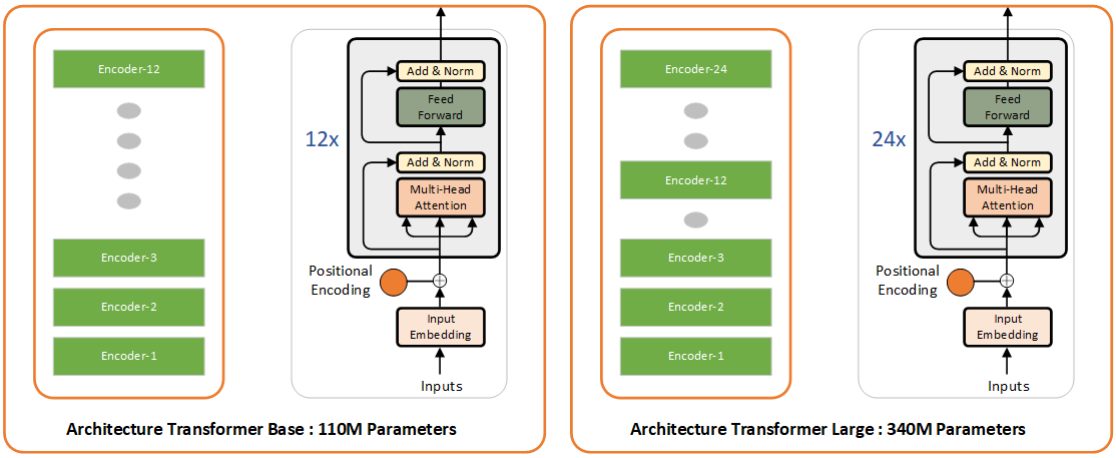

FIGURE 4.3 – Représentation de FLAUBERT - taille et architecture

dans 4.3, chacune avec des dimensions différentes en termes de couches, de têtes d'attention, et donc de nombre de paramètres. Les détails des paramètres pour ces deux tailles de modèle :

La taille d'un modèle de langage comme FlauBERT détermine sa capacité à capturer et à représenter les informations complexes dans un corpus de texte. Dans cette section, nous explorons en détail le nombre de paramètres associés aux différentes variantes de FlauBERT et leur impact sur les performances du modèle.

Nombre de Paramètres : La taille du modèle, exprimée en termes de paramètres, dépend de la variante spécifique de FlauBERT. Généralement, la version "Base" de FlauBERT compte environ 138 millions de paramètres, tandis que la version "Large" peut en comporter jusqu'à 373 millions comme illustré dans le tableau 4.1.

TABLEAU 4.1 – Comparaison entre FlauBERT Base et FlauBERT Large

Caractéristique	**FlauBERT Base**	**FlauBERT Large**
Nombre de couches (transformateurs)	12	24
Taille de plongement (E)	768	1024
Nombre de têtes d'attention (H)	12	16
Taille du vocabulaire (V)	68 731	68 731
Longueur maximale de séquence (L)	512	512
Taille intermédiaire du FFN (F)	3072 (4x plongements)	4096 (4x plongements)
Nombre total de paramètres	138 millions	373 millions

Plongements (Embeddings)

FlauBERT utilise des plongements pour représenter des mots ou des jetons de sous-mots [VSP+17]. Les plongements de sous-mots sont utilisés pour gérer efficacement les mots hors vocabulaire (OOV), soulignant ainsi le rôle essentiel de cette couche dans le modèle. Elle convertit les jetons (tokens) d'entrée en représentations vectorielles continues, permettant au modèle de travailler avec des données textuelles de manière sémantiquement informative. Pour représenter les données d'entrée textuelles, FlauBERT s'appuie sur 3 types distincts de plongements : les plongements de jetons, les plongements de position et les plongements de type de jeton.

Les plongements de jetons (Token Embeddings)

Avant qu'une chaîne de texte ne soit transmise au modèle BERT, le Tokenizer BERT est utilisé pour convertir l'entrée d'une chaîne en une liste d'identifiants de jetons entiers, où chaque identifiant est directement associé à un mot ou à une partie d'un mot dans la chaîne d'origine. Par exemple, la phrase "vers une ia numérique responsable" est convertie par le Tokenizer en les identifiants de jetons suivants :[[0, 197, 30, 5121, 1915, 1148, 1]], comme le montre la Figure 4.4.

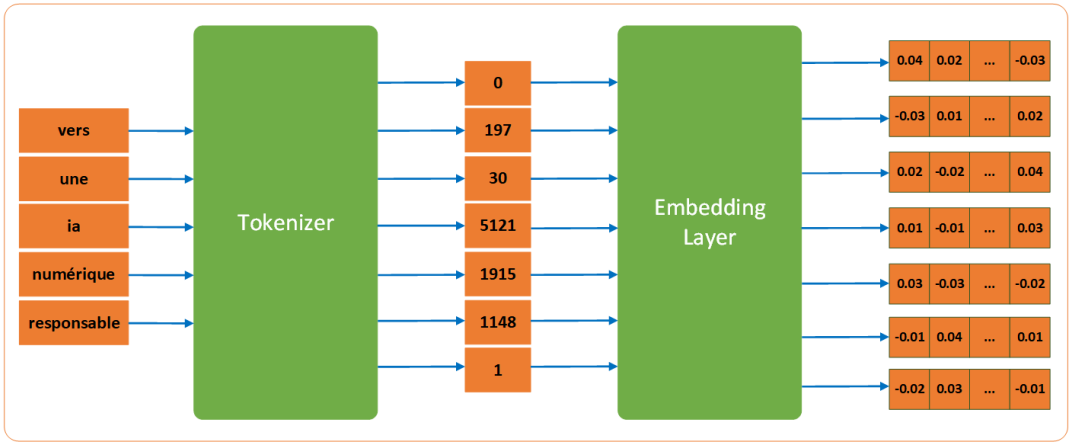

FIGURE 4.4 – Representation de Token Embedding

Pour chaque identifiant de jeton unique (c'est-à-dire pour chacun des 68 731 mots et sous-mots dans le vocabulaire du Tokenizer FlauBERT [4]), le modèle FlauBERT contient un plongement qui est entraîné pour représenter ce jeton spécifique. La couche de plongement à l'intérieur du modèle est responsable de la mise en correspondance des jetons avec leurs plongements. correspondants.

4. FlauBERT possède un vocabulaire étendu géré par des techniques de tokenisation de sous-mots, avec une taille de vocabulaire de 68 731, permettant de gérer un large ensemble de jetons (tokens) uniques et associant chaque jeton à une représentation vectorielle unique.

La **couche de plongement(Embedding layer)** a une taille de vocabulaire de 68 731 et une dimension de plongement de 768, convertissant les jetons d'entrée en représentations vectorielles continues. Ces plongements capturent des informations sémantiques sur les jetons, facilitant ainsi la compréhension et le traitement des données textuelles par le modèle. La dimension de plongement est également appelée **taille cachée (hidden size)**. Il s'agit du nombre de poids entraînables pour chaque jeton du vocabulaire. Le modèle BERT original a une taille cachée de 768, mais d'autres variations de BERT ont été entraînées avec des valeurs de taille cachée plus petites et plus grandes.

Encodages Positionnels (Position Embeddings)

FlauBERT utilise des encodages positionnels pour capturer l'information positionnelle des jetons dans une séquence, ajoutés aux plongements de mots. Ces encodages permettent au modèle de comprendre les positions relatives des jetons et de capturer les dépendances séquentielles dans les données, représentation dans la Figure 4.5.

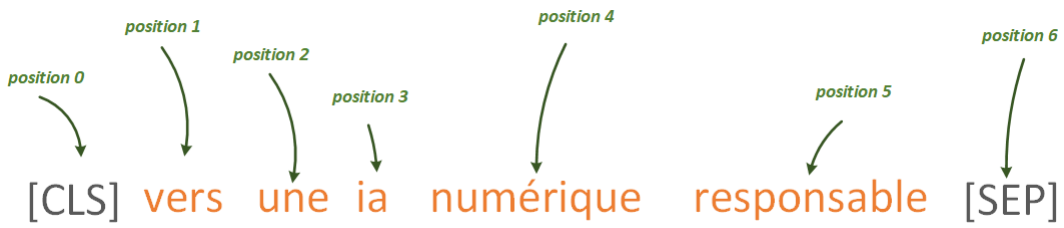

FIGURE 4.5 – Représentation de Plongements Positionnels (Position Embeddings)

Structure de la Couche des Plongements Positionnels :

— La couche des plongements positionnels dans FlauBERT comporte une couche de plongement dédiée à l'encodage des informations positionnelles.

— *Taille du Vocabulaire* : 512, permettant de représenter des positions de 0 à 511, adapté à différentes longueurs de séquences.

— *Dimension de Plongement* : 768, codant efficacement des informations positionnelles complexes en un espace vectoriel de haute dimension.

Les encodages positionnels enrichissent les plongements de mots avec des informations positionnelles, facilitant la compréhension de l'ordre des jetons et la capture des motifs séquentiels, important pour le traitement de séquences.

Plongements de type de jeton (Token Type Embeddings) Le dernier type de plongement utilisé par BERT est le plongement de type de jeton. Une des tâches pour lesquelles BERT a été initialement entraîné consistait à prédire la phrase suivante. Cela signifie que, étant donné deux phrases A et B, FlauBERT, comme BERT, a été entraîné pour déterminer si B suit logiquement A.

Couche de Plongement (Embedding Layer) Étant donné une liste d'identifiants de jetons (tokens IDs), la couche de plongement, représentée dans la Figure 4.6, est responsable de calculer un plongement final pour chaque jeton d'entrée en combinant les trois types de plongements : les plongements de jetons, les plongements de position et les plongements de type de jeton.

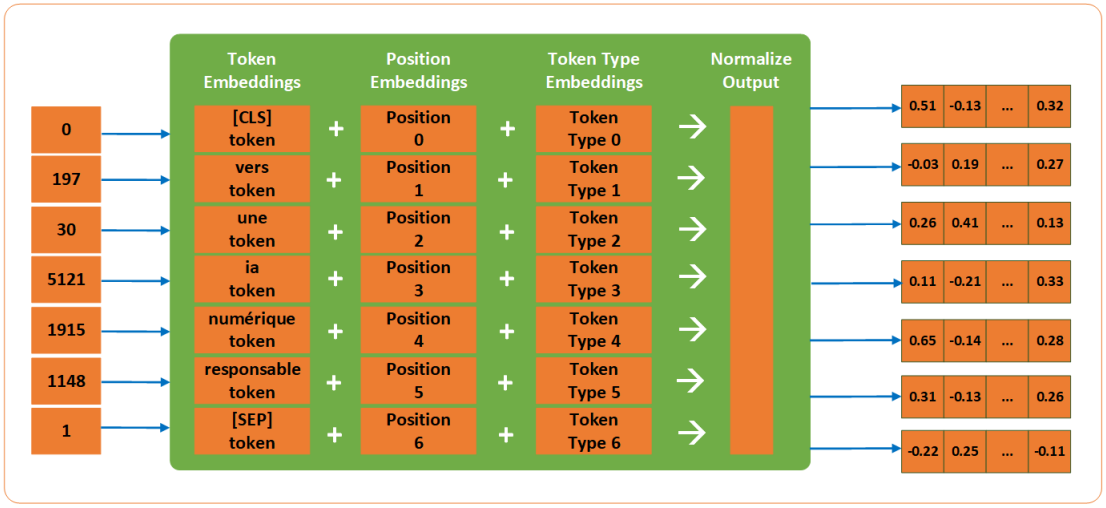

FIGURE 4.6 – La Couche de Plongement (Embedding Layer) pour FlauBERT

Le plongement finale pour chaque jeton est calculée en additionnant les plongements pour son identifiant de jeton spécifique, sa position et son type de jeton, puis en appliquant une normalisation sur les sommes. L'illustration dans la Figure 4.6 montre comment la couche de plongement calcule le plongement pour la chaîne "vers une ia numérique responsable".

Le tableau 4.2 détaille la structure et les paramètres du modèle FlauBERT. Il met en évidence les composants spécifiques tels que les plongements de position, les plongements de jetons, les couches d'attention, la normalisation par couche (Layer Norm), les réseaux feedforward (FFNs), et un module de résumé de séquence (SequenceSummary), chacun avec leurs paramètres distincts et leur rôle dans l'architecture du modèle.

Tableau 4.2 – Paramètres et facteurs influençant la taille de FlauBERT

Composant	**Paramètres**	**Détails**
Position Embedding	393 216 (512 x 768)	Permet au modèle de prendre en compte l'ordre des mots dans la séquence. Dimension de l'embedding : 768.
Plongements	52 783 872 (68 729 x 768)	Transforme les jetons d'entrée en vecteurs de 768 dimensions. Nombre de jetons dans le vocabulaire : 68 729.
Couche d'Attention	Pour les 12 têtes : 28 348 416	Comprend 12 blocs d'attention multi-têtes, chaque tête d'attention contient des couches linéaires avec Input/Output features : 768.
Layer Norm	18 432 (768 x 2 x 12)	Normalisations de couche, avec 12 modules, pour chaque dimension d'embedding : 768.
FFNs (Feed Forward Networks)	56 623 104 [(768 x 3072 + 3072 x 768) x 12]	Comprend 12 réseaux feedforward. Chaque réseau a une première couche linéaire de 768 à 3072 dimensions, suivi d'une activation.
SequenceSummary	1 538	Résume la séquence en une représentation fixe.

FlauBERT : Pré-entraînement et Affinage

Dans cette section, nous explorons les étapes de pré-entraînement et d'affinage qui font de FlauBERT un modèle adaptable à différentes tâches de traitement du langage naturel.

Pré-entraînement

FlauBERT est pré-entraîné sur un large corpus de textes en français, en suivant une approche similaire à celle employée par d'autres modèles linguistiques issus de la famille BERT. Au cours de cette phase de pré-entraînement, le modèle se concentre principalement sur deux tâches : la Modélisation de Langage Masqué (MLM) et la Prédiction de la Phrase Suivante (NSP), ce qui lui permet d'acquérir une compréhension profonde des motifs statistiques et des subtilités linguistiques présentes dans le corpus d'entraînement, avec la représentation du diagramme dans la Figure 4.7 :

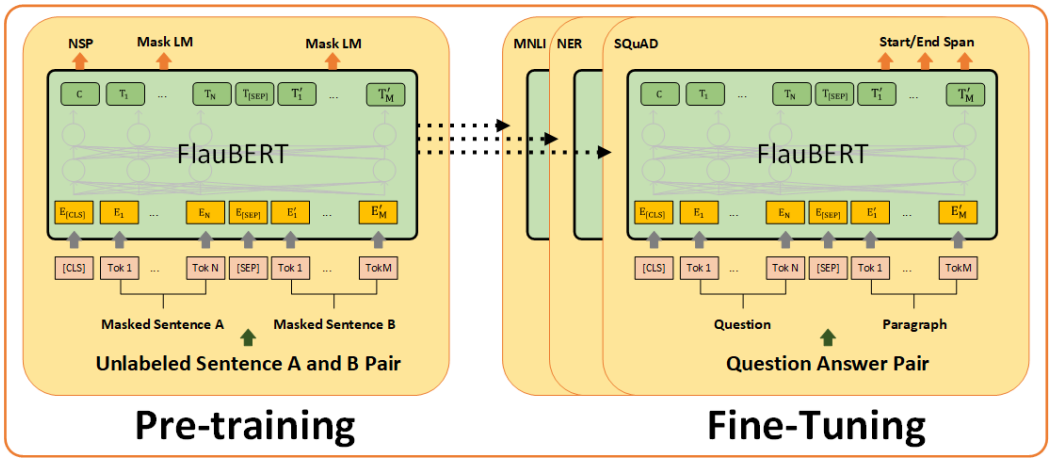

FIGURE 4.7 – Principe général de pré-entraînement et d'ajustement fin de FlauBERT [DCLT18]

Modèle de Langage Masqué (MLM)

La tâche de MLM consiste à masquer aléatoirement certains jetons dans une séquence et à entraîner le modèle à prédire ces jetons masqués en se basant sur leur contexte. La fonction de perte associée à cette tâche est la log-vraisemblance négative, qui est calculée comme suit :

$$\mathcal{L}_{\text{MLM}} = -\sum_{i=1}^{N} \log P(w_{\text{masked}} \mid w_1, w_2, \ldots, w_{i-1}, w_i, \ldots, w_N) \tag{4.11}$$

où N est la longueur de la séquence, masked w_{masked_i} est le token masqué à la position i et 1,2,...,w_1 ,w_2,...,w_N représentent les jetons adjacents dans la séquence.

La tâche MLM est réalisée en masquant environ 15% des mots dans une séquence et en les remplaçant par un jeton spécial [MASK]. Le modèle est ensuite entraîné pour prédire ces mots masqués.

Prédiction de la Prochaine Phrase (NSP)

Le modèle apprend à prédire si la phrase B suit logiquement la phrase A. Habituellement, une paire de phrases A et B est donnée en entrée, et le modèle doit déterminer si B est la phrase qui suit réellement A. La fonction de perte associée à cette tâche peut être la perte de la cross-entropie pour la classification binaire, où le modèle doit prédire la probabilité que B suive A.

Dans le contexte du pré-entraînement des modèles de langage comme BERT ou FlauBERT, la perte de la cross-entropie est souvent calculée pour évaluer la performance du modèle sur les deux tâches : la prédiction de mots masqués (MLM) ou la prédiction de la phrase suivante (NSP). La perte de la cross-entropie est souvent utilisée comme fonction objectif lors de l'entraînement du modèle. Elle est calculée à chaque étape de l'entraînement et utilisée pour mettre à jour les poids du modèle via des techniques d'optimisation telles que la rétropropagation du gradient.

La formule de la perte de la cross-entropie peut être exprimée comme suit pour une prédiction P et une vérité de terrain Q :

$$\text{CrossEntropyLoss} = -\sum_{i} Q(i) \cdot \log(P(i)) \tag{4.12}$$

La formule mesure la distance entre la distribution de probabilité prédite par le modèle (P) et la distribution de probabilité cible (Q). Un résultat de perte plus faible indique que les distributions sont similaires, tandis qu'une perte plus élevée indique des différences importantes entre les distributions.

Ces deux tâches : MLM et NSP, sont utilisées conjointement lors du pré-entraînement de FlauBERT pour permettre au modèle d'apprendre des représentations linguistiques profondes et générales à partir de grandes quantités de données textuelles non annotées.

Affinage (Fine-Tuning)

Une fois le pré-entraînement terminé, FlauBERT peut être affiné sur des differentes tâches, telles que la classification de texte, la reconnaissance d'entités nommées ou la traduction automatique. L'affinage consiste à adapter le modèle aux caractéristiques particulières de la tâche cible en ajustant ses paramètres à l'aide de données annotées supplémentaires. Cette étape permet au modèle de capitaliser sur les représentations apprises lors du pré-entraînement tout en se spécialisant dans la résolution de la tâche spécifique en question.

L'affinage est un processus d'optimisation durant lequel les configurations internes du modèle sont modifiées pour améliorer les performances sur des tâches cibles. Ce processus vise à ajuster le modèle pour qu'il réponde mieux aux caractéristiques des données annotées, en affinant sa capacité à produire des prédictions qui correspondent étroitement aux résultats attendus. Les modifications des configurations sont réalisées à l'aide d'algorithmes d'optimisation, tels que la descente de gradient stochastique, permettant une amélioration progressive et une pertinence accrue du modèle pour la tâche spécifique.

En combinant ces deux phases, FlauBERT parvient à atteindre des bonnes performances dans une gamme diversifiée de tâches de traitement du langage naturel.

Dans le cadre de cette thèse, l'étude de cas est centrée sur la classification de texte. Pour aborder ce sujet, nous approfondirons notre exploration en utilisant **FlaubertForSequenceClassification**, un modèle de langage pré-entraîné spécialement conçu pour le français.

4.2.2 Applications courantes de FlauBERT en classification de texte.

La classe **FlaubertForSequenceClassification** est un composant du modèle FlauBERT utilisé spécifiquement pour les tâches de classification de séquences. Elle est conçue pour traiter des séquences d'entrée et produire des sorties adaptées à la classification.

FlauBERT trouve des applications diverses dans la classification de texte comme :

— Analyse Sentimentale : FlauBERT est utilisé pour déterminer le sentiment d'un texte, par exemple, en classant les critiques de produits ou de films comme positives ou négatives [ZWL18].

— Classification de Catégorie : Il peut classifier des articles ou des documents dans des catégories prédéfinies, telles que des genres littéraires ou des types de documents juridiques [Seb02].

— Détection de Spam et de Contenu Inapproprié : FlauBERT peut être appliqué pour identifier et filtrer les spams ou les contenus inappropriés dans les textes, en utilisant sa capacité à comprendre le contexte et la nuance du langage [DBC+19].

— Classification de Questions : Utilisé dans les systèmes de questions-réponses, FlauBERT aide à catégoriser les questions en fonction de leur nature pour améliorer la précision des réponses [LR02], [LR06].

Pour les tâches de classification, FlauBERT tire parti de son apprentissage bidirectionnel et de sa compréhension approfondie du français. Cependant, en raison de la complexité et de la taille des modèles tels que FlauBERT utilisés dans ces tâches, il est impératif de développer des techniques de compression adaptées pour réduire leur empreinte mémoire et améliorer leur efficacité, tout en préservant leurs performances dans des applications pratiques.

Face à la complexité et à la grande taille de FlauBERT, il est nécessaire d'examiner plus en détail les méthodes de compression visant à réduire la taille du modèle et à alléger la charge computationnelle, lors de la phase d'inférence, tout en conservant ses performances. Pour nos travaux nous fait le choix des trois techniques suivantes :

— La factorisation de rang faible (low-rank factorization) [SKS+13],

— Le clustering de poids [GCHZ17] et

— Le Plongement Inspiré par Kronecker [PSS+20].

Ces approches visent à réduire le nombre de paramètres, ce qui peut accélérer l'inférence et réduire la consommation énergétique. Nous analyserons l'effet de ces méthodes

sur FlauBERT, cherchant à simplifier le modèle tout en préservant sa capacité à effectuer des tâches de NLP avec précision. Notre objectif est de développer des modèles de NLP plus compacts et éco-responsables, qui répondent aux limites matérielles sans altérer la performance linguistique.

4.3 Techniques de Compression

4.3.1 Factorisation de rang faible (Low-Rank Factorization)

Cette technique de compression consiste à réduire la taille des modèles en décomposant leurs matrices de poids en matrices de rang inférieur. L'utilisation de cette technique permet de diminuer la complexité des modèles tout en préservant leur performance.

Le Concept de Factorisation de rang faible

La factorisation de rang faible exploite le concept de sur-paramétrisation des réseaux neuronaux, où les matrices de poids contiennent souvent des redondances et des dépendances inutiles. Cette technique utilise la factorisation de matrices pour remplacer les matrices de poids originales par des approximations de rang inférieur [SKS+13], réduisant ainsi le nombre de paramètres. En décomposant ces matrices en composantes plus simples, il est possible de préserver la capacité du modèle à réaliser des prédictions précises, tout en diminuant la taille du modèle et les ressources de calcul nécessaires.

Le **rang** d'une matrice, concept fondamental en algèbre linéaire, représente le nombre maximum de colonnes (ou de lignes) linéairement indépendantes dans la matrice. Pour une matrice de dimensions mxn, le rang ne peut pas excéder le minimum de m et n, en devenant une matrice de rang faible lorsque son rang est nettement inférieur à ses dimensions d'origine.

Processus de Factorisation de Rang Faible

La factorisation de rang faible consiste à décomposer une matrice de poids de haute dimension en un produit de deux matrices de dimensions inférieures ou plus, comme le représente la Figure 4.8.

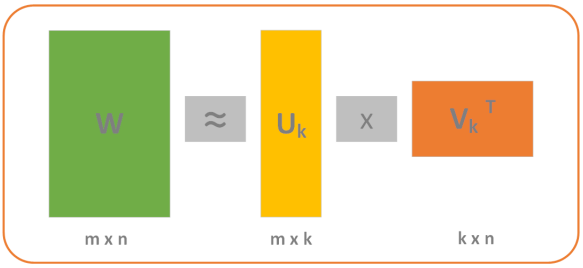

FIGURE 4.8 – Factorisation de rang faible

Une matrice de poids W de forme (M, N) est représentée comme le produit de deux matrices de rang inférieur, U et V, accompagnées d'une matrice diagonale Σ :

$$W \approx U\Sigma V^T.$$

Ici :

— U est une matrice (M, K), où K « M.

— V est une matrice (K, N), où K « N.

Σ est une matrice diagonale (K, K) contenant des valeurs singulières.

Cette factorisation génère une représentation compressée de la matrice de poids d'origine, car les dimensions de U et V sont plus petites que celles de W. Les valeurs singulières dans Σ déterminent la quantité d'informations conservée à partir de la matrice d'origine.

Définition du Rang d'une Matrice

Le rang d'une matrice A, noté rank(A), est défini comme le nombre maximum de vecteurs colonnes ou de vecteurs lignes linéairement indépendants dans la matrice. Pour une matrice $A \in \mathbb{R}^{m \times n}$, son rang est limité par :

$$\text{rank}(A) \leq \min(m, n)$$

Formulation et Optimisation

Le processus de recherche de la factorisation de rang faible implique de résoudre un problème d'optimisation, où l'objectif est de minimiser la différence entre la matrice originale et son approximation, sous contrainte que la matrice d'approximation ait un rang r. Formellement, le problème d'optimisation peut être exprimé comme suit :

$$\min_{\hat{A}} ||A - \hat{A}||_F^2 \quad \text{sous contrainte de} \quad \text{rank}(\hat{A}) = K$$

où $|| \cdot ||_F$ désigne la norme de Frobenius [5].

Qualité de l'approximation et l'impact du rang sur la compression

La qualité de l'approximation dépend du choix du rang K. Un rang K plus bas entraîne une compression plus agressive, car réduit le nombre de paramètres dans un modèle, et peut entraîner une perte de précision du modèle. À l'inverse, un rang K plus élevé préserve plus d'informations mais offre moins de compression. Le défi réside dans la recherche d'un

5. La norme de Frobenius ou la norme matricielle de Frobenius, est une manière de mesurer la "taille" d'une matrice. Elle est définie comme la racine carrée de la somme des carrés de tous les éléments de la matrice.

équilibre optimal entre la réduction du rang et la préservation de l'information. Pour y parvenir on peut envisager de commencer avec un rang relativement élevé et à l'abaisser progressivement tout en évaluant l'impact sur la précision du modèle. Cette méthode permet d'identifier le point où la réduction supplémentaire du rang commence à avoir un effet négatif disproportionné sur la précision.

Les modèles compressés avec la factorisation de rang faible peuvent généraliser mieux et se transférer plus efficacement à différentes tâches, car ils capturent des caractéristiques plus robustes et essentielles [SKS+13].

4.3.2 Clustering de Poids (Regroupement)

Cette technique vise à compresser le modèle en réduisant le nombre de poids uniques, ce qui peut réduire la taille du modèle tout en maintenant ses performances. Le principe de la technique est représente dans la Figure 4.9.

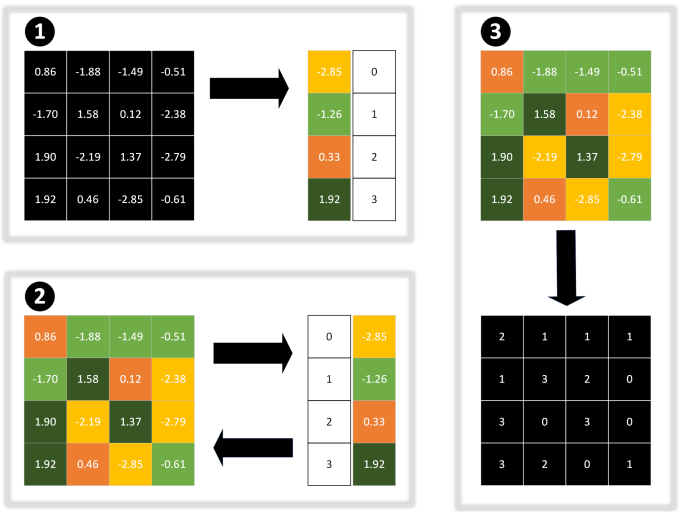

FIGURE 4.9 – Regroupement des poids [HMD16]

Concept du Regroupement

Le regroupement, dans le contexte du NLP et de l'apprentissage automatique, désigne une technique de compression où des paramètres de modèle similaires sont regroupés ou clusterisés. Au lieu de stocker et de traiter chaque poids ou paramètre individuellement, des clusters sont formés, et seuls les centroïdes ou représentants de ces clusters sont conservés. Ces centroïdes capturent les informations essentielles du modèle d'origine tout

en réduisant considérablement la taille du modèle et sa complexité computationnelle [HPTD15].

Processus du regroupement de modèle

Le regroupement de modèle fonctionne en suivant ces étapes :

- **Regroupement des paramètres :** La première étape consiste à regrouper des paramètres de modèle similaires en fonction de certains critères. Dans les modèles NLP, ces paramètres incluent généralement les plongements de mots, les poids dans les couches de réseaux neuronaux et les mécanismes d'attention.
- **Centroïdes de cluster :** Au lieu de stocker chaque paramètre individuellement, l'algorithme calcule les centroïdes des clusters. Ces centroïdes représentent le comportement collectif des paramètres au sein de chaque cluster.
- **Représentation éparse (sparsity) :** Le modèle est ensuite représenté de manière éparse en ne stockant que ces centroïdes de cluster. Lors de l'inférence, lorsque des prédictions sont nécessaires, le modèle se réfère à ces centroïdes pour effectuer les calculs. Cela introduit une forme de "partage de paramètres" où plusieurs poids partagent effectivement la même valeur [HPTD15].
- **Optimisation de l'inférence :** Lors de l'inférence, les données d'entrée sont traitées en se référant aux centroïdes de cluster. Cela se traduit par des prédictions plus rapides et plus efficaces en termes de mémoire.

Algorithmes de Regroupement

Dans le contexte des modèles NLP, plusieurs algorithmes et stratégies de regroupement ont été adaptés pour comprimer efficacement ces modèles. Ces algorithmes visent à regrouper des paramètres de modèle similaires afin de réduire la redondance et d'améliorer l'efficacité. Voici quelques-uns des algorithmes de regroupement couramment appliqués :

- **Regroupement K-Means** : Le K-Means est l'un des algorithmes de regroupement largement utilisés qui partitionne les données en K clusters en fonction de la similarité des caractéristiques. Comme l'a souligné le chercheur de renom, Yann LeCun, "K-Means est un choix naturel pour la compression des modèles de langage, car il est simple, efficace et bien compris." Le K-Means peut être appliqué pour regrouper les plongements de mots ou les poids des couches de réseaux neuronaux afin de réduire la taille du modèle.
- **Méthode de regroupement hiérarchique** : La méthode de regroupement hiérarchique est une autre approche populaire pour la compression des modèles NLP.
- **PCA (Analyse en composantes principales)** : L'utilisation du PCA dans le regroupement des paramètres de modèle NLP extrait les dimensions les plus importantes d'un modèle de langage, ce qui permet une compression significative sans perte majeure de performance [BDLR$^+$04].

— **Autoencodeurs** : Les autoencodeurs sont des réseaux de neurones particuliers qui ont été employés avec succès pour comprimer les modèles NLP. Selon Ilya Sutskever, co-fondateur d'OpenAI, "Les autoencodeurs sont très flexibles et peuvent être adaptés pour compresser efficacement des modèles NLP tout en maintenant leur capacité de génération de texte."

— **K-Means quantifié** : Une variante du K-Means, connue sous le nom de "K-Means quantifié" [Sch92].

Le choix de l'algorithme dépendra souvent de la tâche spécifique et des besoins en termes de performances.

Dans le cadre de la thèse, nous avons employé la méthode de regroupement K-Means pour regrouper les poids du modèle FlauBERT. Cette approche visait à compresser le modèle en réduisant le nombre de poids uniques, afin de diminuer la taille globale du modèle tout en préservant au maximum ses performances en termes de compréhension et de traitement du langage naturel.

Le regroupement K-Means est une méthode de clustering qui partitionne n observations en k clusters dans lesquels chaque observation appartient au cluster avec la moyenne la plus proche, servant de prototype du cluster. Cette méthode est utile pour la compression des modèles NLP car elle permet de regrouper des poids ou des caractéristiques similaires, réduisant ainsi la complexité et la taille du modèle tout en conservant une performance acceptable.

La procédure de l'algorithme K-Means peut être décrite par les étapes mathématiques suivantes :

Initialisation : Choisissez k points initiaux comme centroïdes des clusters. Ces points peuvent être sélectionnés aléatoirement ou selon une certaine logique qui vise à améliorer la convergence de l'algorithme.

Affectation : Affectez chaque observation au cluster dont le centroïde est le plus proche. Formellement, soit x_i une observation (dans ce contexte, un poids ou un vecteur de caractéristiques d'un modèle NLP), l'affectation C_i de x_i est définie par :

$$C_i = \arg\min_j \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \tag{4.13}$$

où $\boldsymbol{\mu}_j$ est le centroïde du cluster j, et $\|\cdot\|$ désigne la norme euclidienne, mesurant la distance entre l'observation x_i et le centroïde $\boldsymbol{\mu}_j$.

Mise à jour des centroïdes : Après avoir affecté toutes les observations à des clusters, les nouveaux centroïdes sont calculés comme le centre moyen de toutes les observations dans chaque cluster. Le nouveau centroïde $\boldsymbol{\mu}'_j$ pour le cluster j est calculé comme :

$$\boldsymbol{\mu}'_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

où $|C_j|$ est le nombre d'observations dans le cluster j.

Convergence : Répéter les étapes plusieurs fois jusqu'à ce que les centroïdes ne changent plus significativement entre les itérations, ou jusqu'à ce qu'un critère d'arrêt prédéfini soit satisfait, indiquant que l'algorithme a convergé.

4.3.3 Plongement Inspiré par Kronecker

La Plongement Inspiré par Kronecker est une méthode qui gère les plongements (embeddings), pour les tâches de NLP. Cette méthode tire son nom du « produit de kronecker », une opération mathématique qui étend le concept de produit matriciel, représentation dans la Figure 4.10. L'objectif principal de cette méthode est de fournir une représentation vectorielle efficace pour les éléments d'un vocabulaire (comme des mots ou des jetons) dans un espace de dimension réduite, tout en conservant une représentation riche et informative.

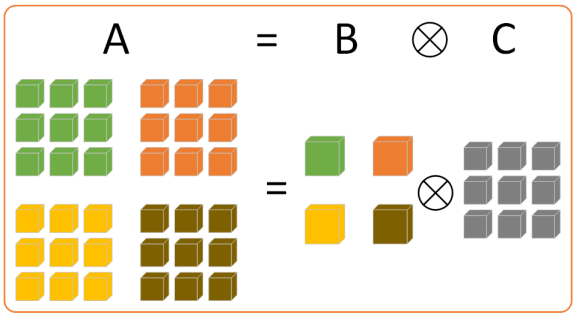

FIGURE 4.10 – Kronecker Produit

Concept du Plongement de Kronecker

Le produit de Kronecker, noté $\otimes$, est une opération algébrique qui généralise le concept de multiplication externe à des matrices de dimensions arbitraires. Il est utilisé dans la méthode Plongement Inspiré par Kronecker, permettant une expansion structurée de l'espace des caractéristiques.

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \tag{4.14}$$

L'objectif principal de la méthode Plongement Inspiré par Kronecker est de fournir une représentation vectorielle efficace pour les éléments d'un vocabulaire, tels que des mots ou des jetons, tout en conservant une représentation riche et informative. Contrairement aux approches traditionnelles qui se concentrent sur la réduction de la dimensionnalité des plongements par des méthodes telles que la SVD (Décomposition en Valeurs Singulières) ou le PCA (Analyse en Composantes Principales), la méthode Plongement Inspiré par Kronecker explore la structure intrinsèque du vocabulaire pour créer des plongements compacts mais informatifs.

Processus du Plongement Inspiré par Kronecker

Soit :

— V le vocabulaire,
— $|V|$ la taille du vocabulaire,
— D la dimension des plongements originaux, et
— K la dimension souhaitée des plongements réduits.

La méthode peut être formalisée de la manière suivante :

1. *Création de la Matrice de Projection P :*

La matrice de projection P de taille $|V| \times K$ est générée en utilisant le produit de Kronecker entre une matrice sélectionnée U de dimension $D \times K$ et le vecteur ou la matrice représentant les plongements originaux V.

$$P = V \otimes U \tag{4.15}$$

2. *Projection des plongements :*

Les nouveaux plongements E' sont obtenus en projetant les plongements originaux E de dimension D dans l'espace de dimension réduite K à l'aide de la matrice de projection P.

$$E' = E \times P \tag{4.16}$$

3. *Vecteurs de Reconstruction :*

Nous avons R vecteurs de reconstruction, chacun ayant une dimension D. Le processus de reconstruction consiste à ajouter les dimensions perdues lors de la compression initial :

$$E'' = E' + RV \tag{4.17}$$

où RV représente la matrice des vecteurs de reconstruction ajoutés aux plongements compressés E'. Cette étape présente l'intérêt de récupérer une partie de l'information qui pourrait avoir été perdue durant la réduction de dimensionnalité, permettant ainsi de maintenir la performance globale du modèle malgré la compression appliquée.

Cette intégration des vecteurs de reconstruction améliore significativement la fidélité des plongements après compression, en fournissant un moyen robuste de préserver les nuances et la richesse des données originales.

4. *Indice de Compression :*

L'indice de compression, C, mesure l'efficacité de la réduction de dimensionnalité, défini comme le rapport entre la dimension originale et la dimension réduite des plongements.

$$C = \frac{D}{K} \tag{4.18}$$

Cet indice de compression indiquerait combien de fois la dimensionnalité des plongements a été réduite lors du processus de projection à l'aide de la méthode Plongement Inspiré par Kronecker. Un C plus élevé indiquerait une compression plus importante, ce qui signifie que les plongements réduits occupent moins d'espace tout en essayant de conserver autant que possible les informations pertinentes des plongements initiaux.

4. *Conservation de l'Information* :

Malgré la réduction de dimension, la méthode vise à préserver autant que possible les informations essentielles contenues dans les plongements originaux, permettant ainsi de maintenir une performance optimale des modèles NLP utilisant ces plongements réduits. La méthode Plongement de Kronecker a été appliquée dans diverses tâches de NLP, y compris la classification de texte, la génération de langage naturel, et la recherche d'information [GWJ20].

Clarification sur l'Utilisation du Produit de Kronecker :

Bien que le nom « Plongement Inspiré par Kronecker » suggère une utilisation directe du produit de Kronecker mathématique, il est important de préciser que la méthode mise en œuvre dans notre approche n'emploie pas littéralement cette opération pour la réduction des plongements. Au lieu de cela, l'inspiration derrière le nom découle de la façon dont nous combinons des informations provenant de deux sources distinctes — les plongements réduits des jetons et un vecteur de compression global — dans le but de préserver une quantité maximale d'informations dans un espace à dimension

réduite. Cette stratégie nous permet d'atteindre l'efficacité en termes de compression et de maintenir la richesse sémantique des plongements, s'alignant sur l'esprit du produit de Kronecker par la manière dont elle structure et enrichit l'espace des caractéristiques.

Cette clarification souligne notre objectif d'exploiter des principes inspirés par le produit de Kronecker, plutôt que d'appliquer directement l'opération algébrique, pour créer des plongements efficaces et informatifs adaptés aux tâches de NLP. Notre méthode vise ainsi à équilibrer la préservation de l'information, facilitant l'usage des modèles NLP dans des contextes où les ressources de calcul ou de stockage sont limitées.

L'intégration des méthodes de compression dans FlauBERT post-entraînement offre une stratégie efficace pour réduire la complexité du modèle tout en maintenant des performances élevées. Ces méthodes sont utiles pour les applications où les ressources de calcul et la mémoire sont limitées, mais où une précision élevée est toujours requise.

Nous avons exploré des techniques avancées de compression de modèles, telles que la factorisation de rang faible, le clustering de poids et le Plongement Inspiré par Kronecker, qui offrent des moyens efficaces de réduire la complexité et l'empreinte mémoire de modèles de traitement du langage naturel tels que FlauBERT. Ces méthodes promettent de rendre les modèles de NLP plus accessibles et performants, même dans des environnements avec des ressources limitées, en préservant la qualité des prédictions tout en optimisant l'utilisation de la mémoire et la vitesse d'inférence.

4.4 Conclusion

Dans un premier temps, nous avons introduit FlauBERT et son architecture, mettant en lumière les mécanismes d'auto-attention et d'encodage qui sous-tendent ce modèle de traitement automatique du langage naturel pour la langue française. Nous avons ensuite poursuivi avec la présentation de trois techniques de compression - la factorisation de rang faible, le Plongement Inspiré par Kronecker, et le clustering des poids.

La factorisation de rang faible agit sur la réduction des dimensions des matrices de poids, en les décomposant en produits de matrices de plus petites dimensions, ce qui allège le modèle sans sacrifier significativement l'exactitude des prédictions. Le clustering de poids regroupe les poids similaires, permettant de partager les mêmes valeurs pour plusieurs connexions dans le réseau, réduisant ainsi le nombre de poids uniques à stocker. Enfin, le Plongement Inspiré par Kronecker propose une méthode pour paramétrer les plongements en exploitant la structure mathématique du produit de Kronecker, réalisant de la compression sans perdre en qualité de représentation sémantique.

Chacune de ces méthodes sera examinée sous le prisme de son application à Flau-

BERT, en évaluant leur impact sur la simplification du modèle tout en veillant à maintenir une performance robuste dans des scénarios d'application réels.

Dans le chapitre suivant, nous présentons un cas d'usage de FlauBERT appliqué à la classification de texte, en nous concentrant spécifiquement sur une tâche de classification binaire utilisant une base de données en open-source. Nous aborderons la mise en application pratique des différentes mesures et examinerons la consommation énergétique. Un comparatif entre le modèle de base et les modèles compressés sera réalisé, mettant en évidence l'impact de la compression sur la consommation énergétique sans compromettre les performances des modèles.

Synthèse

→ Augmentation de la taille et des besoins en ressources de FlauBERT.

→ Introduction des techniques de compression pour contrer cette augmentation.

→ Réduction de l'empreinte mémoire par le regroupement des poids similaires et l'utilisation de représentations compactes.

→ Application de la régularisation par limitation du nombre de valeurs distinctes que peuvent prendre les poids, aidant à prévenir le surajustement.

→ Diminution de la dimensionnalité des plongements pour réduire l'espace de stockage nécessaire.

→ Réduction du nombre de paramètres à stocker pour une meilleure utilisation de la mémoire.

→ Diminution du coût computationnel des multiplications matricielles pour accélérer les calculs.

→ Amélioration de l'adaptabilité aux diverses tâches de NLP grâce à une structure simplifiée et à une meilleure généralisation des caractéristiques apprises.

Chapitre 5

Résultats expérimentaux

Sommaire

Dans ce chapitre, nous présentons nos contributions qui consistent dans l'utilisation des méthodes de compression spécifiques visant à réduire la consommation énergétique des modèles de traitement automatique de langage naturel, précisément le modèle FlauBERT, sans altérer leurs performances post-entraînement. La validation des travaux est effectuée à l'aide de PowerAPI, outil permettant de réaliser les mesures nécessaires et liées à la consommation d'énergie des modèles. Les mesures de consommation d'énergie ont été effectuées dans un environnement expérimental spécifiquement construit pour ce besoin.

5.1 Introduction

Dans le chapitre précédent, nous avons exploré trois méthodes de compression applicables à l'optimisation des modèles IA utilisés dans la branche NLP, visant à réduire leur consommation énergétique sans compromettre leurs performances, et cela, après la phase d'entraînement.

Dans ce chapitre nous allons décliner spécifiquement l'utilisation de ces méthodes de compression dans un contexte d'utilisation du modèle FlauBERT.

Pour valider la performance de ces méthodes de compression, nous avons mis en place une procédure de validation visant à mesurer l'efficience énergétique des modèles dans la phase d'inférence. Pour la réalisation de ces mesures nous nous sommes fait accompagné de l'outil Open Source PowerAPI, développé par l'équipe ADAM de l'INRIA [1], les détails de son fonctionnement étant décrites dans [BNRS13] et sur le site Internet [2] dédié.

5.2 Environnement expérimental

Cette section explique les outils et les technologies utilisés dans la mise en œuvre et l'évaluation des méthodes d'optimisation proposées dans la phase d'inférence.

5.2.1 Description de la méthodologie d'expérimentation

Pour être en mesure de tirer les bonnes conclusions par rapport à la performance énergétique des modèles optimisés la démarche suivante a été appliquée à chacune des itérations d'expérimentation :

— A partir du modèle FlauBERT de base, appliquer les méthodes de compression afin de construire une nouvelle version du modèle.
— Déployer le nouveau modèle dans l'environnement d'expérimentation.
— Executer le programme de test, en charge de la simulation de la phase d'inférence et du démarrage des mesures de la consommation énergétique.
— Récupérer les mesures en vue des interprétations.
— Validation du modèle sous deux angles : précision (accuracy) et consommation énergétique.

L'ensemble de ces étapes sont illustrées dans la Figure 5.1.

1. INRIA : Institut National de Recherche en Informatique et en Automatique, établissement public à caractère scientifique et technologique, institution de formation. Inria
2. PowerAPI

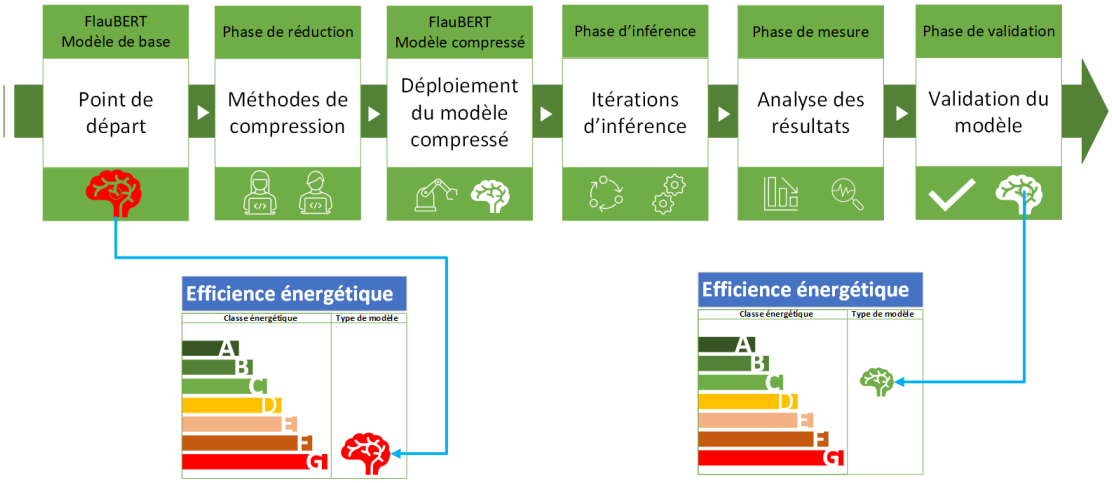

FIGURE 5.1 – Les étapes de validation du modèle compressé

5.2.2 Caractéristiques techniques de l'environnement

Pour la mise en place de l'environnement d'expérimentation nous avons utilisé comme base un serveur LENEVO avec les caractéristiques suivantes :

- **Modèle de serveur :** LENOVO X3650 M5 R028
- **Microprocesseur :** deux microprocesseurs multicoeurs Intel Xeon E5-2600 v3 avec deux liens QuickPath Interconnect (QPI) jusqu'à 9,6 GT par seconde
- **Mémoire :** 384 Go avec barrettes LRDIMM
- **Stockage :** 4 Unités de disque dur SAS/SATA 2,5 remplaçables à chaud de 1 To
- **Alimentation :** deux blocs d'alimentation 750 W

La distribution d'Ubuntu en version 20.04 LTS a été installée sur ce serveur pour permettre ensuite la mise en place de l'ensemble des autres applications nécessaires (Python, Docker, Mongo DB, Grafana).

5.2.3 Outil de mesure de la consommation énergétique

Pour réaliser les mesures nécessaires à l'estimation du coût énergétique nous avons fait le choix de l'outil Open Source PowerAPI. Il a été créé dans le but de permettre aux utilisateurs de réaliser et personnaliser des solutions modulables qui fournissent des estimations d'énergie à différents niveaux et fréquences.

Il fait également partie des outils intégrés dans le catalogue de référence des logiciels

libres recommandés par la MiNumEco [3] à travers le programme TECH.GOUV [4].

Après l'installation sur le serveur d'expérimentation, nous avons intégré PowerAPI dans la pipeline de test, chose possible du fait de sa conception modulaire. Un diagramme représentant le principe de fonctionnement est représenté dans la Figure 5.2 [5].

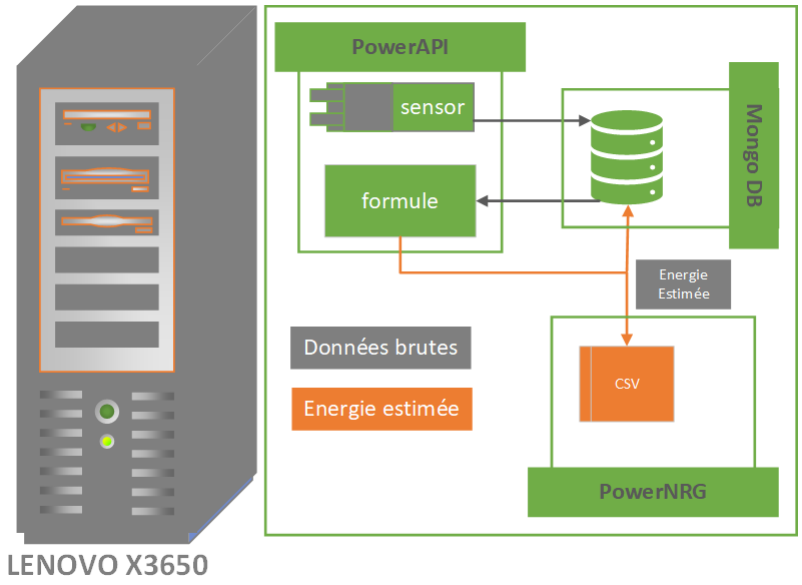

FIGURE 5.2 – Schéma générique de fonctionnement de PowerAPI

.

Une étape de configuration spécifique a été aussi nécessaire afin d'adapter le paramétrage des capteurs pour surveiller la consommation par nos inférences, et en utilisant les formules fournies par PowerAPI pour calculer la consommation énergétique, nous avons pu obtenir des mesures des consommations.

Description du fonctionnement de PowerAPI

PowerAPI embarque plusieurs composants nécessaires pour son fonctionnement [CRS15] :

— **Clock** représente l'horloge interne permettant le fonctionnement à différentes fréquences. Des messages sont publiés sur le bus d'événements interne en fonction des fréquences paramétrées.

3. MiNumEco, la mission interministérielle numérique écoresponsable.
4. tech.gouv
5. PowerAPI

— **Monitor** représente le suivi énergétique d'un ou plusieurs processus. Il réagit aux messages publiés par la Clock en fonction de la fréquence paramétrée pour le suivi. Des messages sont publiés sur le bus d'événements interne pour chaque processus suivi. Monitor est également en charge d'agréger les estimations en utilisant la fonction paramétrée par l'utilisateur.

— **Sensor** composant permettant de se connecter au système pour récolter les métriques utiles à la représentation des activités des processus. Ces métriques peuvent représenter la consommation globale d'une machine reportée par des sondes internes (RAPL [KHN+18]) ou des statistiques d'utilisation du processeur (PROCFS [6]). Ce composant réagit aux messages publiés par les Monitor et publie les données récoltées à destination de la formule adéquate.

— **Formula** représente le modèle utilisé pour produire une estimation énergétique. Il réagit aux messages publiés par le Sensor et convertit les métriques envoyées en consommations.

— **Reporter** produit des rapports formatés permettant d'afficher les estimations énergétiques selon un format choisi. Ces rapports peuvent être affichés, par exemple, au sein d'une interface Web, d'un fichier ou directement en console.

Focus sur le fonctionnement du capteur et de la formule dans notre contexte

Le capteur collecte les données brutes provenant des ressources matérielles pendant l'exécution de l'application visée par les mesures et les stocke dans une base de données de type Mongo DB. Cette collecte de données permet d'avoir une vision précise et en temps réel de l'utilisation de ces ressources matérielles, ce qui est nécessaire pour l'analyse de la consommation d'énergie.

Les types de capteurs présents dans PowerAPI permet de mesurer divers paramètres tels que l'utilisation du processeur, la mémoire, les opérations d'entrée/sortie, et d'autres indicateurs de performance qui sont essentiels pour évaluer la consommation d'énergie d'une application .

La Formule accède aux informations stockées dans cette base de données, transforme les données brutes en une consommation d'énergie Watt. Pour le faire, PowerAPI utilise un modèle inféré par régression pour estimer la consommation énergétique avec taux

6. Sur les systèmes du type Unix, procfs (process file system, système de fichiers processus en anglais) est un pseudo-système de fichiers (pseudo car dynamiquement généré au démarrage1) utilisé pour accéder aux informations du noyau sur les processus. Le système de fichiers est souvent monté sur le répertoire /proc. Puisque /proc n'est pas une arborescence réelle, il ne consomme aucun espace disque mais seulement une quantité limitée de mémoire vive. Cela aboutit à un paradoxe apparent : un fichier non vide a une taille affichée de 0 (avec ls). procfs est supporté sur les systèmes suivants : Solaris, BSD, Linux, qui étend le concept au-delà des processus (entre autres, description détaillée des composants matériels, du réseau...etc), IBM AIX (qui utilise l'implémentation de Linux pour une meilleure compatibilité), QNX, (Unix temps réel).

d'erreur maximal observé inférieur à 1%. Elle est capable de transformer ces données en informations compréhensibles et utiles, telles que la quantité d'énergie consommée par une fonction spécifique du logiciel ou par un composant matériel.

La formule dans PowerAPI convertit les données complexes en estimations de consommation d'énergie, ce qui permet d'identifier les parties du logiciel qui sont les plus énergivores. La formule est capable de stocker les résultats dans une base de données ou de les exporter dans un fichier au format CSV, offrant ainsi une flexibilité pour l'analyse et le partage des données a posteriori.

Nous avons configuré ces deux composants clés de PowerAPI, le capteur et la formule, le capteur permettant de surveiller en temps réel l'utilisation de CPU par notre application NLP dans la phase d'inférence, tandis que la formule a été définie pour calculer la consommation énergétique à partir des données recueillies par le capteur.

Dans notre contexte nous avons fait le choix d'utiliser la formule SmartWatts [FRS20] de PowerAPI pour analyser la consommation d'énergie de Python (langage utilisé pour le développement de notre application de test PowerNRGIA). SmartWatts utilise l'interface RAPL pour collecter des mesures de base sur la consommation d'énergie du CPU et de la DRAM, ainsi que des événements de compteurs de performance matérielle pour estimer la consommation d'énergie des conteneurs applicatifs. Il gère en temps réel des modèles de puissance.

Ce logiciel de mesure d'énergie offre une solution économe en ressources, sans nécessiter d'investissements matériels spécifiques et parfois conséquents, et peut être déployé dans une variété de typologies d'environnements (serveurs physiques, plate-formes virtualisées, serveurs hébergés dans un cloud privé ou publique avec clusters HPC distribués) fournissant des estimations de puissance en temps réel avec un taux d'erreur inférieur à 3,5% en moyenne.

Procédure d'exécution des tests de simulation de la phase d'Inférence

Pour chaque séance de test, nous avons lancé 100 appels d'inférence, chacun étant répété sur 1000 itérations. Le choix de 1000 itérations est motivé par la nécessité d'obtenir un échantillon représentatif des performances du modèle. En effet, en répétant chaque inférence un grand nombre de fois, nous réduisons l'influence des variations aléatoires et du bruit qui pourraient affecter les résultats. Cela permet de mieux lisser les fluctuations et d'obtenir une estimation plus précise et fiable des performances moyennes du modèle.

Quant au nombre de 100 inférences, il a été choisi comme un compromis pour gérer les contraintes liées à la bande passante mémoire et aux ressources disponibles. Effectivement, augmenter ce nombre à 1000 inférences aurait alourdi de manière significative la charge sur le système, ce qui pourrait entraîner des goulots d'étranglement ou des perturbations non souhaitées dans le flux de traitement. À l'inverse, une seule inférence

aurait été insuffisante pour capturer la variabilité inhérente à l'inférence en temps réel, risquant de donner une vue trop simplifiée et potentiellement biaisée de la performance du modèle. Ainsi, en choisissant 100 inférences, nous nous assurons d'avoir un échantillon suffisamment large pour obtenir une mesure fiable, tout en restant dans des limites raisonnables en termes de consommation des ressources.

Pour analyser la consommation énergétique nécessaire pour l'ensemble des inférences effectuées l'outil mesure la puissance en watts en temps réel en utilisant les compteurs de performance et ensuite en se basant sur les données recueillies par ces compteurs il calcule la puissance consommée par le processus d'inférence a chaque instant.

5.3 FlauBERT : méthodes de compression

Pour notre étude, parmi les différentes déclinaisons de FlauBERT la classe FlaubertForSequenceClassification [7]a été sélectionnée pour effectuer une série d'expérimentations visant à évaluer sa capacité à classifier des séquences de texte. Nous avons adopté une approche méthodique en appliquant des techniques de compression post-entraînement telles que la factorisation de rang faible, le clustering de poids et le Plongement Inspiré par Kronecker. Ces méthodes ont été choisies pour leur potentiel à réduire la charge computationnelle tout en maintenant la performance du modèle.

Au début de nos travaux, nous avons pris comme hypothèse l'utilisation de FlauBERT pour la classification multicritère, avec un cas d'usage concret et lié au traitement des mails issus de France Travail (anciennement le Pôle Emploi), visant à automatiser la classification des demandes utilisateurs.

Par la suite, pour des raisons de confidentialité, nous avons fait le choix d'une application plus générique et accessible, en utilisant une base de données Open Source, la base "Allociné" allocine.fr. Cette base est une collection de données recueillies à partir du site web Allociné, qui propose des informations sur les films, les séries télévisées, et autres contenus liés au cinéma et à la télévision. Ce choix m'a permis de tester l'efficacité des techniques de compression sur un cas d'usage largement utilisé et reconnu dans la communauté de recherche autour de NLP.

En mettant l'accent sur la configuration détaillée du modèle FlauBERT et l'analyse de son nombre de paramètres, mon objectif est de parvenir à trouver un équilibre le plus optimal possible entre l'efficacité énergétique et un maintien de performances acceptables.

Nous avons minutieusement examiné l'impact de différentes stratégies de compression

7. FlaubertForSequenceClassification

sur la précision (accuracy) du modèle, en veillant à ne retenir que celles qui ne compromettent pas la qualité des prédictions par rapport au modèle original. Pour chaque méthode de compression, plusieurs réglages ont été testés, générant ainsi divers modèles à degrés de compression variés. Nos démarches visaient à obtenir un modèle FlauBERT optimisé en termes de consommation énergétique et de performance computationnelle pour la tâche de classification de texte.

5.3.1 Plongement Inspiré par Kronecker

En exploitant la structure mathématique du produit de Kronecker, solution proposée à travers la méthode de Plongement Inspiré par Kronecker, inovante pour le paramétrage des plongements, nous visons une compression efficace sans perdre en qualité de représentation sémantique.

Application et Résultats

Notre objectif est de comprimer efficacement les plongements des jetons (tokens), qui sont essentiels pour la compréhension du texte par le modèle, en utilisant l'approximation des plongements par Plongement Inspiré par Kronecker. Factuellement, nous avons réduit la dimensionnalité des matrices de plongements E en les remplaçant par le produit de deux matrices plus petites, A et B, grâce à la méthode de Kronecker. Cette approche a été conçue pour maintenir l'intégrité des informations essentielles portées par les plongements, tout en allégeant la charge computationnelle.

Processus de Réduction Dimensionnelle

— *Transformation des Identifiants de Jetons* : Les identifiants de jetons (input_ids) sont convertis en plongements de jetons grâce à la couche de plongements spécifique. En ajustant le facteur de compression, nous avons pu réduire la dimensionnalité initiale de ces plongements de 768 à une dimension N plus petite, optimisant ainsi l'utilisation de l'espace.

— *Fusion avec les Vecteurs de plongement* : Les plongements de jetons ainsi réduits ont été combinés avec les plongements vectoriels issus de la couche vector_embedding. Cette fusion crée une représentation enrichie et condensée des jetons, facilitant leur traitement tout en minimisant la perte d'informations sémantiques.

— *Synthèse du Plongement Final* : En additionnant les deux séries de plongements comprimés, nous obtenons le plongement final. Cette étape permet de maintenir une richesse sémantique des données traitées tout en bénéficiant d'une réduction considérable du volume de données nécessaires au fonctionnement du modèle.

Calcul de la taille du plongement après compression

L'utilisation et l'application de la méthode de compression par Plongement Inspiré par Kronecker implique de réduire chaque plongement de 768 dimensions à une dimension plus petite, contrôlée par le paramètre N.

Le plongement initial dans le modèle Flaubert est de dimension 768. Par exemple, pour une dimension réduite N de 100, nous ramenons chaque plongement de 768 dimensions à une dimension de 100. De cette manière, chaque plongement est réduit à environ 13% de sa taille initiale. Puisque chaque plongement correspond à une colonne dans la matrice de plongement, et que la matrice est réduite en utilisant Plongement Inspiré par Kronecker, on obtient une réduction massive du nombre total de paramètres. Après compression, le nombre total de paramètres est de 6,872,900, ce qui représente une réduction significative par rapport aux 52,785,792 paramètres initiaux du plongement, d'un facteur d'environ 7,7 fois pour un choix du rang de compression de 100.

Lors de nos expériences, nous avons pu mesurer l'effet de cette compression sur la capacité du modèle à classer des séquences de texte. L'objectif étant de vérifier que, malgré une réduction notable du nombre de paramètres, le modèle conservait une précision de prédiction élevée. Nos résultats démontrent très bien le fait que le Plongement Inspiré par Kronecker permet une gestion plus efficiente des ressources computationnelles par le modèle, sans avoir à faire un compromis significatif sur ses performances.

Processus itératif pour trouver le bon équilibre

Nous avons procédé à de multiples réglages pour identifier le bon équilibre entre le degré de compression et la précision de la prédiction, ajustant le rang de compression. Cette démarche a nécessité une série d'expérimentations pour déterminer le niveau de compression idéal qui préserve la l'efficacité du modèle sans compromettre sa précision.

La Figure 5.3, à travers le **graphique**[8], représente une analyse visuelle comparative de la consommation énergétique de FlauBERT, soumis à différentes configurations de Plongement Inspiré par Kronecker, spécifiquement pour des degrés de compression r=50, 100, 200, et 300. Ces configurations sont désignées respectivement par KE 50, KE 100, KE 200, et KE 300 dans l'analyse. Cette visualisation met en évidence les variations de la consommation énergétique associées à chaque configuration, révélant l'impact direct de l'ajustement de Plongement Inspiré par Kronecker sur l'efficacité énergétique du modèle.

8. Graphique qui permet de visualiser la distribution complète des données

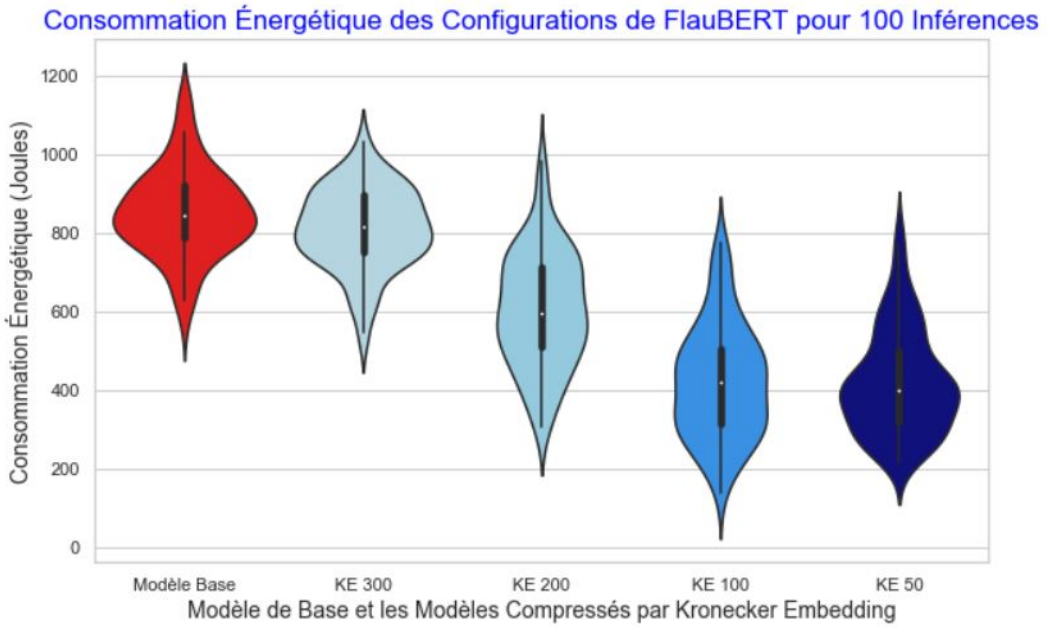

FIGURE 5.3 – Résultats de Compression par le Plongement de Kronecker

Contrairement à ce que l'on pourrait attendre, une forte compression de la couche de plongement, comme dans le cas de KE300, peut entraîner une perte d'information importante. Pour compenser cette perte et reconstruire le vecteur original, un effort computationnel supplémentaire est nécessaire, ce qui augmente la consommation énergétique. Ainsi, une compression excessive peut paradoxalement augmenter l'empreinte énergétique.

En revanche, une compression modérée, telle que KE100, équilibre mieux la réduction de la taille du modèle et la préservation de l'information essentielle. Elle permet une consommation énergétique plus raisonnable tout en maintenant la performance du modèle. Cela suggère que pour la méthode de Plongement Inspirée par Kronecker, il est crucial de choisir un rang de compression qui minimise les pertes d'information sans alourdir inutilement la charge computationnelle.

Lorsque nous appliquons la technique de Plongement Inspiré par Kronecker, nous modifions la structure des plongements pour réduire le nombre total de paramètres tout en cherchant à préserver les informations essentielles à la performance du modèle. Cette réduction des paramètres impacte directement la consommation énergétique lors de l'inférence, en réduisant la complexité calculatoire et en optimisant l'utilisation des caches de données.

En affinant le rang de compression, on peut potentiellement réduire davantage la

consommation énergétique sans affecter significativement les performances du modèle.

5.3.2 Factorisation à Rang Faible

La deuxième méthode choisie dans le cadre de notre expérimentation visant à optimiser la consommation énergétique du modèle FlauBERT, nous avons adopté une approche pour mesurer l'efficacité de la factorisation à rang faible sur différentes configurations des couches d'attention.

La configuration initiale du modèle comprenait 12 couches d'attention, chacune avec un nombre substantiel de paramètres contribuant à la charge computationnelle et à la consommation énergétique. Nos résultats ont montré qu'une application ciblée de la factorisation à rang faible sur une sélection de 6 couches d'attention offrait un meilleur compromis entre la consommation énergétique et le maintien de la performance. Cette configuration particulière a permis de réduire la consommation énergétique sans compromettre la précision des prédictions du modèle.

Application et Résultats

Réduction Dimensionnelle des Matrices de Poids

Nous avons appliqué la factorisation de rang faible pour compresser efficacement les matrices de poids dans l'objectif de minimiser la taille du modèle.

Processus de Compression et Amélioration de la Performance

— *Compression des Matrices de Poids* : Nous avons reconstruit les matrices de poids en versions de rang inférieur.

— *Optimisation de la Performance* : Grâce à la compression, la taille du modèle a été réduite.

Bilan de la Compression

Pour calculer le nombre de paramètres réduits dans ce modèle FlauBERT après l'application de la factorisation de rang faible avec un rang r, nous utilisons la formule générale de la factorisation de rang faible appliquée à une matrice de poids W de dimension mn. Lorsque nous factorisons cette matrice en deux matrices de rang inférieur U et V, où U est de dimension mr et V est de dimension rn, le nombre total de paramètres après la factorisation devient :

$$\text{Nombre de paramètres après factorisation} = (m \times r) + (r \times n) \tag{5.1}$$

En sachant que matrice de poids originale W a une dimension 768×768 (ce qui est typique pour un modèle comme FlauBERT dans ses couches linéaires). Pour le rang r=20, le calcul serait :

$$\text{Nombre de paramètres après factorisation} = (768\times20)+(20\times768) = 15360+15360 = 30720 \tag{5.2}$$

Ceci est à comparer aux 768×768=589 824 paramètres de la matrice originale.

La réduction totale des paramètres pour le modèle FlauBERT, après l'application de la factorisation de rang faible, est de 13 574 248 paramètres. Cela représente une réduction d'environ 9.81% par rapport au modèle original qui compte 138 371 946 paramètres. Cette réduction des paramètres traduit l'efficacité de la factorisation de rang faible dans la simplification du modèle tout en maintenant ses performances.

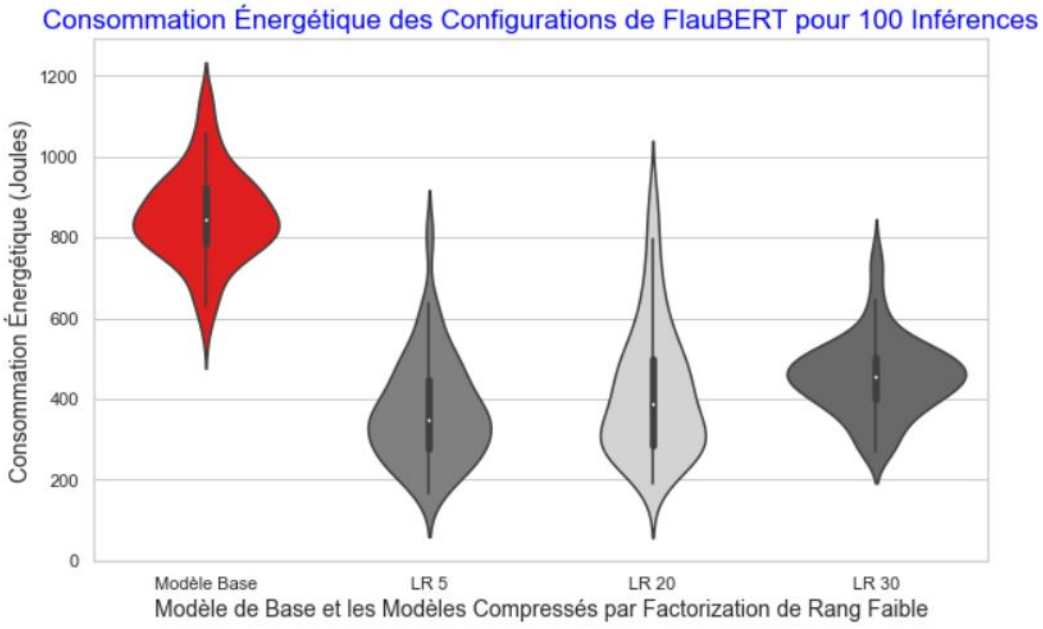

FIGURE 5.4 – Résultats de Compression par Factorization de Rang Faible

Dans la figure 5.4, nous constatons une tendance selon laquelle la consommation d'énergie diminue à mesure que le rang de compression diminue. Le choix du niveau de compression doit être adapté au modèle spécifique et aux données utilisées.

Réduire la taille d'un modèle peut améliorer la consommation énergétique jusqu'à un certain point, mais une compression excessive peut entraîner une perte de efficience énergétique comme montré dans la figure 5.5.

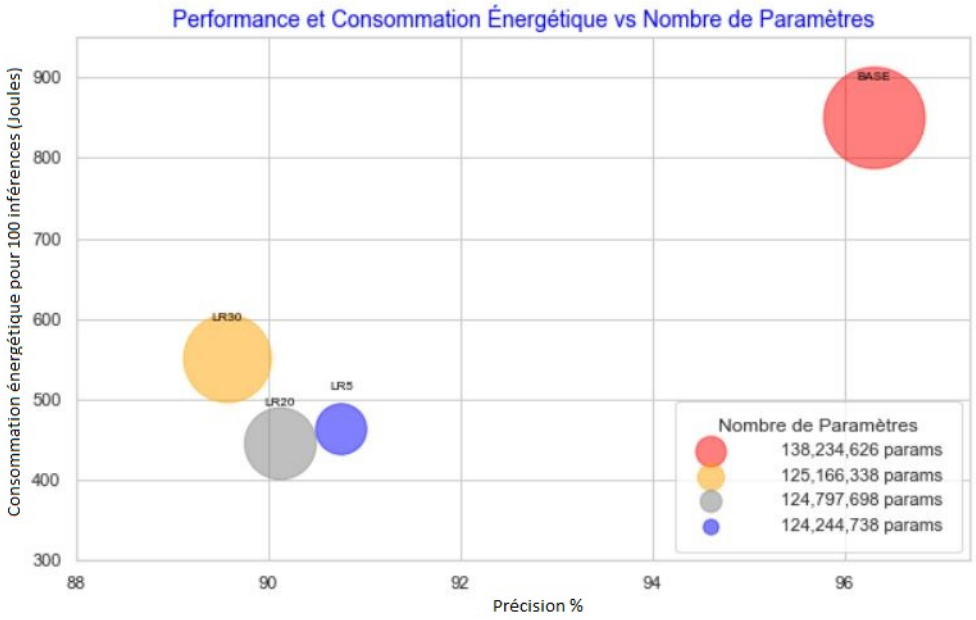

FIGURE 5.5 – Compression : Précision (Accuracy) vs Consommation Énergétique vs Nombre de Paramètres

Cela démontre qu'on ne peut pas établir systématiquement un lien entre les MACs et la consommation énergétique. Il peut y avoir plusieurs raisons : un modèle plus compressé peut utiliser moins efficacement la capacité computationnelle du matériel que le même modèle moins compressé. Ou encore, la stratégie de mouvements des données entre différents niveaux de mémoire (par exemple, de la RAM au cache) et entre les processeurs peut éclipser les coûts de calcul [SCYE17].

L'objectif est d'identifier le seuil optimal où minimiser la taille du modèle améliore son efficacité sans nuire la performance du modèle. Atteindre cet équilibre entre taille, précision, et performance opérationnelle nécessite une approche itérative dans la compression des modèles.

Concernant l'application de la factorisation de rang faible, cela dépend de plusieurs facteurs, notamment la nature spécifique de la tâche et des données, ainsi que des exigences de performance du modèle. En général, il est recommandé de réaliser le fine-tuning après la compression, car le processus de fine-tuning peut aider à préserver ou à rétablir les performances du modèle tout en bénéficiant des avantages de la compression.

5.3.3 Regroupement des Poids

L'application du regroupement des poids a été effectuée dans le but de réduire la complexité de FlauBERT en minimisant le nombre de poids uniques. Le processus déployé est décrit par la suite.

Application et Résultats du Clustering K-means des Poids

Réduction de la Complexité du Modèle

En regroupant les poids similaires et en les remplaçant par un centroid commun, a permis de réduire la diversité des poids tout en préservant les fonctionnalités essentielles du modèle.

Mécanisme de Compression et d'Optimisation

— *Compression des Poids* : Nous avons appliqué le clustering K-means pour réduire la précision des poids dans le modèle FlauBERT sur les couches d'attention et les couches feed-forward networks. Cela a permis de simplifier la représentation des poids et réduiser la taille globale du modèle

— *Optimisation de la Performance* : Nous avons évalué la performance du modèle clusterisé sur un ensemble de validation afin d'évaluer l'impact du clustering sur la précision et la capacité de généralisation. Cette évaluation a permis de comparer les résultats avec ceux du modèle original non-clusterisé.

Bilan de la Compression

Le clustering, telle qu'il est implémenté, ne réduit pas directement le nombre total de paramètres du modèle. Il regroupe les poids existants des couches linéaires spécifiques (dans les modules d'attention) en un nombre fixe de clusters et remplace chaque poids par le centre du cluster le plus proche, Figure 5.6.

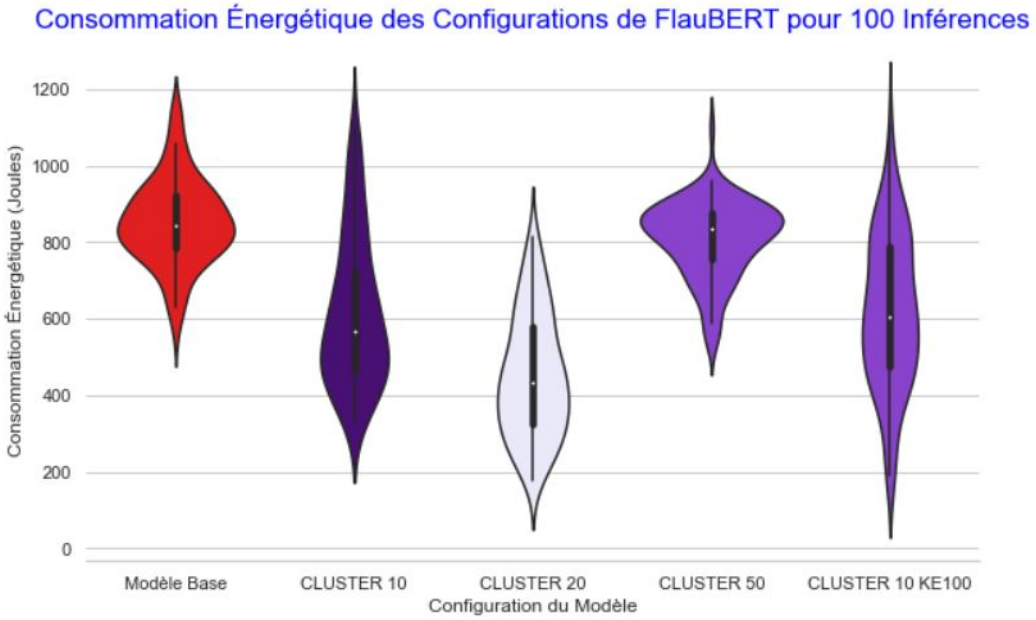

FIGURE 5.6 – Clustering Compression

Cela signifie que, bien que le nombre de valeurs uniques de poids dans ces couches soit réduit, la structure et la dimensionnalité globales des matrices de poids restent inchangées. En regroupant les poids de FlauBERT via la méthode K-means, nous avons pu observer une réduction de la taille du modèle sans compromettre sa performance. Une variabilité plus importante est observée dans les mesures énergétiques pour la compression avec un nombre de 10 et 20 clusters, ainsi que pour le modèle où nous avons combiné deux techniques de compression, à savoir le clustering avec 10 clusters suivi de l'application de Plongement Inspiré par Kronecker pour le plongement.

L'utilisation de différents nombres de clusters pour le clustering des poids dans le modèle FlauBERT peut avoir des impacts variés sur la consommation énergétique.

10 Clusters :

— Compression Forte : Avec 10 clusters, la compression est forte. Chaque cluster représente une large gamme de poids, réduisant la diversité des poids.

— Accès aux Centroïdes : Lors de l'inférence, le modèle utilise les centroïdes des clusters pour les calculs à la place des poids d'origine. Avec moins de clusters, chaque cluster représente une plus grande variété de poids, ce qui peut entraîner des requêtes plus fréquentes pour les centroïdes appropriés.

— Accès Mémoire : Les accès mémoire sont moins prévisibles car les poids sont répartis sur un plus petit nombre de clusters, chacun couvrant une gamme plus large de valeurs. Cela signifie que les accès aux centroïdes sont moins réguliers et plus dispersés en termes de localisation en mémoire. Les accès mémoire irréguliers sont moins optimisés par les mécanismes de cache, ce qui peut conduire à des cache

misses (lorsque les données requises ne sont pas trouvées dans le cache et doivent être récupérées de la mémoire principale). Les cache misses sont plus énergivores car la récupération de données depuis la mémoire principale est plus lente et consomme plus d'énergie.

20 Clusters :

— Équilibre entre Compression et Précision : Avec 20 clusters, il y a un bon équilibre entre compression et précision. Chaque cluster couvre une gamme moyenne de poids, ce qui maintient une bonne précision.

— Optimisation de la Consommation Énergétique : La consommation énergétique est optimisée, car les accès mémoire sont plus réguliers et les calculs restent efficaces sans nécessiter de compensation significative pour la précision.

50 Clusters :

— Compression moindre : Chaque cluster couvre une gamme très étroite de poids, se rapprochant des valeurs originales.

— Consommation énergétique élevée : La consommation d'énergie augmente presque au niveau du modèle original. Plusieurs explications possibles incluent : Les accès mémoire deviennent plus fréquents et détaillés, ce qui augmente l'overhead (surcharge) des accès mémoire et peut mener à une utilisation inefficace du cache. Avec un nombre élevé de clusters, l'overhead de la gestion des clusters (comme l'assignation et la recherche des clusters) peut devenir significatif. Cela ajoute une complexité computationnelle supplémentaire qui augmente la consommation énergétique. À 50 clusters, les bénéfices de la compression en termes de réduction de la taille du modèle sont diminués. La réduction de la mémoire nécessaire n'est plus significative par rapport aux clusters de 20, et les coûts énergétiques de la gestion des clusters supplémentaires contrebalancent les gains.

Il est essentiel de tester plusieurs configurations de clustering pour trouver le nombre optimal de clusters qui offre le meilleur gain en termes de réduction de la consommation énergétique. Le nombre optimal de clusters atteint un équilibre où les bénéfices de la compression sont maximisés sans introduire une surcharge de gestion ou de calcul significative.

5.4 Discussions

Les configurations des modèles ont été représentées sur un graphique en nuage de points, Figure 5.7 où l'axe des abscisses (axe X) illustre l'accuracy des modèles, exprimée en pourcentage, tandis que l'axe des ordonnées (axe Y) indique la consommation énergétique, mesurée en joules.

Parmi les modèles évalués, quatre configurations ont été mises en avant, encadrées en vert dans notre graphique pour leur équilibre entre efficacité énergétique et précision. Pour ce type de modèle, spécifiquement dans le cadre d'une tâche de classification

FIGURE 5.7 – Consommation versus Précision par configuration de modèle

binaire, deux méthodes basées sur le Plongement Inspiré par Kronecker et deux sur le regroupement de poids se sont distinguées, démontrant les meilleurs compromis entre consommation énergétique et la précision (accuracy) pour ce cas d'usage spécifique.

Dans nos expérimentations, nous avons tenté d'associer deux techniques de compression, à savoir le Plongement Inspiré par Kronecker avec la factorisation de rang faible, ainsi que le Plongement Inspiré par Kronecker avec le regroupement de poids, dans l'objectif de réduire la consommation énergétique de notre modèle FlauBERT lors de l'inférence. Toutes les combinaisons de techniques n'apportent pas nécessairement des gains significatifs et que certaines peuvent même se révéler inefficaces.

Lors de l'association de deux techniques de compression, comme le Plongement Inspiré par Kronecker avec la factorisation de rang faible et le Plongement Inspiré par Kronecker avec le regroupement de poids, l'objectif était de maximiser la réduction de la consommation énergétique pour le modèle FlauBERT durant l'inférence. Cependant l'efficacité de ces combinaisons peut varier considérablement en fonction de plusieurs facteurs, tels que la nature spécifique des données traitées et la configuration du modèle.

La combinaison de techniques de compression implique une augmentation de la complexité dans la gestion des paramètres du modèle. Le Plongement Inspiré par Kronecker modifie la structure des plongements (embeddings), tandis que la factorisation de rang

faible se concentre sur la réduction du nombre de paramètres dans les couches de poids. Cette complexité peut entraîner des difficultés dans l'optimisation du modèle, rendant parfois certaines combinaisons moins efficaces que prévu. De plus, chaque technique de compression a ses propres avantages et inconvénients qui peuvent se complémenter ou, dans certains cas, entrer en conflit, menant à des résultats sous-optimaux.

5.5 Conclusion

Nous avons pu démontrer que, grâce à l'application des techniques de compression et l'utilisation d'un outil comme PowerAPI pour la mesure énergétique, il est tout à fait possible d'optimiser la consommation d'énergie des modèles NLP .

L'application des techniques telles que la factorisation de rang faible, le clustering de poids, et Plongement Inspiré par Kronecker ont été évaluées dans un cadre expérimental, permettant d'observer leur impact sur la consommation énergétique dans la phase d'inférence. Les travaux réalisés démontre qu'il est possible d'optimiser l'efficacité énergétique des modèles NLP en appliquant ces techniques de compression, ouvrant ainsi la voie à une utilisation plus durable et économe en ressources des technologies de NLP.

Notre approche, avec l'application de méthodes de compression sur des modèles de NLP, dans notre cas FlauBERT, vise principalement à explorer des nouvelles stratégies d'optimisation adaptatives plutôt qu'à établir une solution universelle et applicable dans toute circonstances ou de quantifier de manière absolue la consommation énergétique de ce type de modèles. La sélection des techniques de compression et l'ajustement de leurs paramètres doivent tenir compte d'une multitude de variables spécifiques à chaque scénario d'utilisation. Ces facteurs incluent non seulement la complexité du modèle et le type de tâche envisagée mais aussi la nature des données traitées et les spécificités de l'environnement d'exécution telles que la configuration du processeur et les caractéristiques du dataset.

L'optimisation d'un modèle d'IA pour un cas d'usage spécifique exige une personnalisation approfondie. Cela implique une évaluation et l'ajustement des niveaux de compression pour trouver le juste équilibre entre le maintien de la précision des prédictions et la réduction de la consommation énergétique. Ce processus itératif de test et de mesure vise à identifier la configuration optimale qui, dans un contexte donné, assure un équilibre idéal entre performance et efficacité énergétique et il est difficilement généralisable. Chacune de méthodes de compression présentée précédemment a des avantages uniques, et leur efficacité dépend fortement du contexte d'application qui peut être des fois très spécifique.

Le Plongement Inspiré par Kronecker, quant à lui, exploite le produit de Kronecker pour réduire la dimensionnalité des plongements tout en préservant les informations essentielles, offrant une manière efficace de comprimer les données sans perte importante

de performance.

La factorisation de rang faible vise à décomposer les matrices de poids volumineuses en produits de matrices de dimensions inférieures, réduisant ainsi le nombre de paramètres nécessaires sans sacrifier significativement la qualité des prédictions.

Le regroupement de poids consiste à partitionner les poids du modèle en clusters et à les remplacer par la valeur moyenne de chaque cluster, simplifiant le modèle tout en minimisant la perte d'information.

Synthèse

→ Présentation de la démarche d'expérimentation

→ Application des techniques de compression post-entraînement.

→ Réduction de la consommation d'énergie des nouveaux modèles par rapport au modèle de base.

Chapitre 6

Conclusions et perspectives

Sommaire

6.1 Conclusions

Les réseaux neuronaux profonds ont révolutionné une multitude des domaines d'applications en intelligence artificielle et leur développement croissant continu de surprendre à la fois les scientifiques mais également l'ensemble de notre société. Leurs architectures, par leur complexité croissante et la profusion de leurs paramètres, bénéficie aussi des avancées technologiques, liées à l'évolution des processeurs graphiques GPU et des plateformes spécialisées, ce qui facilite leur entraînement dans des délais raisonnables. Confrontés à la complexité croissante de ces modèles, notamment dans le domaine du traitement automatique du langage naturel (NLP), nous avons compris l'importance de guider nos actions de manière efficace les actions pour maximiser l'efficience énergétique.

Dans ce mémoire, nous avons combiné l'analyse de deux éléments structurants concernant l'IA, le calcul et la consommation d'énergie qui nous permettent d'avoir une perspective nouvelle et différente et bien plus complète sur l'impact énergétique de l'IA.

L'élément distinctif de notre analyse est que nous nous concentrons sur le coût de l'inférence, qui est généralement plus faible que le coût d'entraînement lorsqu'ils sont tous deux rapportés à la recherche scientifique, mais en raison de facteurs multiplicatifs, il est beaucoup plus élevé dans l'ensemble. La plus part des modèles de réseaux neuronaux profonds, dont les modèles NLP, sont entraînés très peux de fois et utilisés des millions de fois après leur mise en service.

Nos travaux se concentre sur la réduction de la consommation énergétique selon trois aspects : décomposition des matrices de poids, regroupement des poids et la compression des couches de plongement. L'ensemble des trois méthodes partagent un principe commun qui consiste en la réduction de la complexité des modèles et leur poids avec l'objectif de réduire leur empreinte énergétique et faire ainsi un premier pas vers une IA Numérique Responsable.

Décomposition en Facteurs de Rang Faible Cette technique permet de réduire le nombre de paramètres nécessaires pour représenter le modèle, tout en conservant ses propriétés essentielles. Elle exploite la structure de corrélation entre les poids pour identifier les régions où une décomposition en facteurs de rang faible est efficace.

Clustering de Poids Cette méthode regroupe les poids similaires en clusters, puis quantifie les poids de chaque cluster pour les représenter de manière plus compacte. En identifiant les poids qui ont des valeurs proches, le clustering de poids permet de réduire la précision nécessaire pour les représenter, ce qui entraîne une compression du modèle.

Plongement Inspiré par Kronecker En exploitant la structure régulière des couches de plongement, le Plongement Inspiré par Kronecker (Kronecker Inspired Embedding) permet de réduire le nombre de paramètres nécessaires pour représenter ces couches.

Dans l'ensemble, ces techniques visent à exploiter différentes propriétés et structures des réseaux neuronaux pour réduire leur taille tout en minimisant les pertes de performance. Il est important de souligner qu'il existe un *compromis à faire entre le choix du facteur de compression, la performance du modèle et la réduction de la consommation énergétique*. Chaque facteur doit être testé et ajusté en fonction de la spécificité du modèle, des données utilisées et de la tâche à accomplir. Cette démarche permet d'assurer que la compression n'altère pas de manière significative la capacité du modèle à réaliser ses fonctions de base tout en optimisant l'efficacité énergétique. L'objectif est de choisir judicieusement ces différents paramètres pour trouver l'équilibre optimal qui soutient à la fois l'innovation en IA et notre responsabilité environnementale.

6.2 Perspectives

Poursuivant l'idée visant à diminuer la consommation énergétique des modèles de réseaux neuronaux profonds, sans compromettre leurs performances, une nouvelle approche consiste dans l'évaluation et ensuite la classification de l'impact énergétique par rapport à l'utilisation des techniques de compression spécifiques. Cette démarche ouvre la perspective de développer des classifications énergétiques adaptées pour chaque type de modèle et pour chaque type d'usage. Une telle catégorisation, fondée sur l'efficience éner-

gétique, pourrait transformer nos critères de sélection et d'évaluation des technologies d'intelligence artificielle, en plaçant la consommation énergétique au centre de préoccupations dans le développement des services numériques, avec l'ambition de créer des vrais services numérique responsable.

Une vrai adoption de techniques de quantification, qui consiste à diminuer la précision des poids du modèle en passant de 32 bits à 16 bits en virgule flottante, et potentiellement d'aller jusqu'à 2-4 bits en entier, en fonction des capacités du processeur, devient impérative. Cette idée ne se limite pas seulement à une réduction de la consommation énergétique, mais peut également faciliter le déploiement des modèles sur des dispositifs à faible puissance. L'exploration des accélérateurs matériels, conçus spécifiquement pour optimiser les calculs liés à l'IA, représente une autre piste d'amélioration tant en termes de performances que d'efficience énergétique.

L'avènement de l'informatique quantique représente une nouvelle voie pour le traitement automatique du langage naturel. L'informatique quantique, avec sa capacité à effectuer des calculs complexes à grande vitesse grâce à la superposition et l'intrication quantique, va permettre d'aller encore plus loin dans le domaine du NLP. Les plongements de mots, qui sont au cœur des modèles de NLP d'aujourd'hui, pourraient être calculés à l'aide de techniques quantiques. L'utilisation de l'informatique quantique pour calculer des distances ou des similarités dans des espaces de plongement de grande dimension pourrait rendre ces opérations plus rapides, permettant des analyses sémantiques plus complexes sur de vastes corpus de texte. À long terme, cette technologie pourrait aboutir à la création de nouveaux modèles NLP quantiques, surpassant largement les performances des approches classiques en termes d'efficacité et de précision.

Enfin, il devient aussi nécessaire d'apprendre et de se familiariser avec les méthodes de compression adaptées aux modèles de type LLM (Large Language Models) qui, malgré leur popularité croissante, se heurtent à des contraintes significatives en termes de consommation énergétique due à leur gigantesque nombre de paramètres et de données nécessaires pour leur développement.

Ces idées soulignent l'importance de poursuivre les recherches dans le domaine de l'IA afin de la rendre plus efficiente de point de vue énergétique, et de ce fait plus responsable. En adoptant cette voie de l'innovation nous pouvons aspirer à créer une intelligence artificielle non seulement performante et au service de tous mais également durable et respectueuse de notre environnement.

Publications issues de ces travaux de recherche

2023 Angela Ciocan and Vincent Courboulay
Sustainability of neural network applications in training and inference
- Some approaches and practices
2023 International Conference on ICT for Sustainability (ICT4S)

2024 Angela Ciocan et Vincent Courboulay
Vers une IA numérique responsable : Les actions concrètes des institutions publiques
Colloque Intelligence artificielle et institutions publiques : enjeux, controverses et perspectives
Publication validée. Date de sortie début 2025.

2024 Angela Ciocan et Vincent Courboulay
Vers une IA numérique responsable : Les actions concrètes dans la gestion des territoires
Ouvrage L'avènement de la société numérique dans les territoires :
Des pratiques citoyennes et des politiques publiques en mutation
Publication validée. Date de sortie début 2025.

Tableau 6.1 – Liste des publications récentes et à venir d'Angela Ciocan et Vincent Courboulay

Bibliographie

[AAH+23] Shams Forruque Ahmed, Md Sakib Bin Alam, Maruf Hassan, Mahtabin Rodela Rozbu, Taoseef Ishtiak, Nazifa Rafa, M. Mofijur, A. B.M. Shawkat Ali, and Amir H. Gandomi. Deep learning modelling techniques : current progress, applications, advantages, and challenges. *Artificial Intelligence Review*, 56(11), 2023.
Voir page 50.

[ABS12] Nikolaos Alachiotis, Simon A. Berger, and Alexandros Stamatakis. Coupling SIMD and SIMT architectures to boost performance of a phylogeny-aware alignment kernel. *BMC Bioinformatics*, 13(1), 2012.
Voir page 71.

[AHPF18] Shaahin Angizi, Zhezhi He, Farhana Parveen, and Deliang Fan. IMCE : Energy-efficient bit-wise in-memory convolution engine for deep neural network. In *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, pages 111–116, 2018.
Voir page 77.

[AKS20] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker : Tracking and Predicting the Carbon Footprint of Training Deep Learning Models, 2020.
Voir page 20.

[AR21] Adrián Alcolea and Javier Resano. FPGA accelerator for gradient boosting decision trees. *Electronics (Switzerland)*, 10(3), 2021.
Voir page 73.

[AS23] Sulaiman Aftan and Habib Shah. A Survey on BERT and Its Applications. In *20th International Learning and Technology Conference*, 2023.
Voir page 12.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, 2019.
Voir page 53.

[BBB+22] Seonmyeong Bak, Colleen Bertoni, Swen Boehm, Reuben Budiardja, Barbara M. Chapman, Johannes Doerfert, Markus Eisenbach, Hal Finkel, Oscar Hernandez, Joseph Huber, Shintaro Iwasaki, Vivek Kale, Paul R.C. Kent, Jae Hyuk Kwack, Meifeng Lin, Piotr Luszczek, Ye Luo, Buu Pham, Swaroop Pophale, Kiran Ravikumar, Vivek Sarkar, Thomas Scogland, Shilei Tian, and P. K. Yeung. OpenMP application experiences : Porting to accelerated nodes. *Parallel Computing*, 109, 2022.
Voir page 80.

[BCB15] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
Voir page 49.

[BCNM06] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model Compression Cristian. *Kdd*, 54(1) :1–9, 2006.
Voir page 62.

[BDG+22] Sathwika Bavikadi, Abhijitt Dhavlle, Amlan Ganguly, Anand Haridass, Hagar Hendy, Cory Merkel, Vijay Janapa Reddi, Purab Ranjan Sutradhar, Arun Joseph, and Sai Manoj Pudukotai Dinakarrao. A Survey on Machine Learning Accelerators and Evolutionary Hardware Platforms. *IEEE Design and Test*, 39(3), 2022.
Voir page 76.

[BDLR+04] Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean François Paiement, Pascal Vincent, and Marie Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16 :2197–2219, 2004.
Voir page 107.

[BGMSS18] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. SGD learns over-parameterized networks that provably generalize on linearly separable data. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.
Voir page 53.

[BGNL21] Nesrine Bannour, Sahar Ghannay, Aurélie Névéol, and Anne Laure Ligozat. Evaluating the carbon footprint of NLP methods : a survey and analysis of existing tools. In *SustaiNLP 2021 - 2nd Workshop on Simple and Efficient Natural Language Processing, Proceedings of SustaiNLP*, 2021.
Voir page 50.

[BHV19] Bauke Brenninkmeijer, Youri Hille, and Arjen Vries. *On the Generation and Evaluation of Tabular Data using GANs.* PhD thesis, 2019.
Voir pages 3 et 41.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, 2016.
Voir page 95.

[BLZ+22] S. A. Budennyy, V. D. Lazarev, N. N. Zakharenko, A. N. Korovin, O. A. Plosskaya, D. V. Dimitrov, V. S. Akhripkin, I. V. Pavlov, I. V. Oseledets, I. S. Barsola, I. V. Egorov, A. A. Kosterina, and L. E. Zhukov. eco2AI : Carbon Emissions Tracking of Machine Learning Models as the First Step Towards Sustainable AI. *Doklady Mathematics*, 106, 2022.
Voir page 36.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 2020-December, 2020.
Voir pages 50, 56 et 88.

[BNRS13] Aurélien Bourdon, Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. PowerAPI : A Software Library to Monitor the Energy Consumed at the Process-Level. *ERCIM News*, 92, 2013.
Voir page 115.

[BOFG20] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the State of Neural Network Pruning ?, 2020.
Voir page 60.

[Bot10] Bottou Léon. *Proceedings of COMPSTAT'2010.* Physica-Verlag HD, 2010.
Voir page 47.

[BPC20] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer : The Long-Document Transformer, 2020.
Voir page 59.

[BRSH22] Dieter Balemans, Philippe Reiter, Jan Steckel, and Peter Hellinckx. Resource efficient AI : Exploring neural network pruning for task specialization. *Internet of Things (Netherlands)*, 20, 2022.
Voir page 60.

[CBS+17] Yeongjae Choi, Dongmyung Bae, Jaehyeong Sim, Seungkyu Choi, Minhye Kim, and Lee Sup Kim. Energy-Efficient Design of Processing Element for Convolutional Neural Network. *IEEE Transactions on Circuits and Systems II : Express Briefs*, 64(11), 2017.
Voir pages 73, 74 et 75.

[CCP17] Alfredo Canziani, Eugenio Culurciello, and Adam Paszke. Analysis of Deep Neural Network architectures for practical applications. *Iclr*, 2017.
Voir page 52.

[CCZT14] Yunji Chen, Tianshi Chen, Xu Zhiwei, and Olivier Temam. DianNao Family : Energy-Efficient Hardware Accelerators for Machine Learning. *Communications of the ACM*, 57(5), 2014.
Voir pages 73, 74 et 75.

[CGRS19] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers, 2019.
Voir page 49.

[CH23] Kung Fu Chen and Ding Yong Hong. Rewriting Deep Learning Models for Maximizing Edge TPU Utilization. In *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, volume 2023-January, 2023.
Voir page 74.

[Cha02] Barbara Chapman. Parallel application development with the hybrid MPI+OpenMP programming model. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2474, 2002.
Voir page 80.

[CHMH20] Gang Chen, Shengyu He, Haitao Meng, and Kai Huang. PhoneBit : Efficient GPU-Accelerated Binary Neural Network Inference Engine for Mobile Phones. In *Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition, DATE 2020*, 2020.
Voir page 77.

[CHSV17] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave Gaussian quantization. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
Voir page 36.

[CLT+19] Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that : Deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
Voir pages 73 et 75.

[CRS15] Maxime Colmant, Romain Rouvoy, and Lionel Seinturier. Estimation de la consommation des systèmes logiciels sur des architectures multi-coeurs. In *Compas*, Lille, France, 2015.
Voir page 117.

[CSAK14] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project ADAM : Building an efficient and scalable deep learning training system. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation*, 2014.
Voir page 71.

[CWZZ17] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A Survey of Model Compression and Acceleration for Deep Neural Networks, 2017.
Voir pages 56 et 85.

[CXS+20] Yiran Chen, Yuan Xie, Linghao Song, Fan Chen, and Tianqi Tang. A Survey of Accelerator Architectures for Deep Neural Networks, 2020.
Voir page 72.

[DBC+19] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, and Opeyemi Emmanuel Ajibuwa. Machine learning for email spam filtering : review, approaches and open research problems. *Heliyon*, 5(6), 2019.
Voir page 103.

[DCLT18] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : Pre-training of deep bidirectional transformers for language understanding. *arXiv*, pages 4171–4186, 2018.
Voir pages 4, 49, 56, 88 et 101.

[DD18] Dario Amodei and Danny Hernandez. AI and Compute, 2018.
Voir pages 50 et 52.

[Dha20] Payal Dhar. The carbon impact of artificial intelligence. *Nature Machine Intelligence*, 2(8), 2020.
Voir page 50.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2011.
Voir page 48.

[DM98] L. Dagum and R. Menon. OpenMP : an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1), 1998.
Voir page 80.

[DM17] Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.
Voir page 49.

[Dom08] Max Domeika. Getting Ready for Low Power Intel Architecture. In *Software Development for Embedded Multi-core Systems*, pages 347–383. Elsevier, 2008.
Voir page 35.

[DP19] Peter B. Denton and Stephen J. Parke. Simple and Precise Factorization of the Jarlskog Invariant for Neutrino Oscillations in Matter. *Physical Review D*, 100(5), 2019.
Voir page 61.

[DPY18] Jeff Dean, David Patterson, and Cliff Young. A New Golden Age in Computer Architecture : Empowering the Machine-Learning Revolution. *IEEE Micro*, 38(2), 2018.
Voir pages 73 et 75.

[ECF22] Lees Perasso Etienne, Vateau Caroline, and Domon Firmin. EVALUATION DE L'IMPACT ENVIRONNEMENTAL DU NUMERIQUE EN FRANCE ET ANALYSE PROSPECTIVE. Technical report, ADEME, Arcep, Paris, 2022.
Voir pages 6, 7 et 8.

[EMA+16] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo Di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences of the United States of America*, 113(41), 2016.
Voir pages 73, 74 et 75.

[EMH19] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search : A survey. *Journal of Machine Learning Research*, 2019.
Voir page 63.

[ESSP+17] Hesham El-Sayed, Sharmi Sankar, Mukesh Prasad, Deepak Puthal, Akshansh Gupta, Manoranjan Mohanty, and Chin Teng Lin. Edge of Things : The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment. *IEEE Access*, 2017.
Voir page 85.

[ESYF+20] Frank Emmert-Streib, Zhen Yang, Han Feng, Shailesh Tripathi, and Matthias Dehmer. An Introductory Review of Deep Learning for Prediction Models With Big Data, 2020.
Voir page 73.

[FAH+20] Jessica Fjeld, Nele Achten, Hannah Hilligoss, Adam Nagy, and Madhulika Srikumar. Principled Artificial Intelligence : Mapping Consensus in Ethical and Rights-Based Approaches to Principles for AI. *SSRN Electronic Journal*, 2020.
Voir page 19.

[FC22] Luciano Floridi and Josh Cowls. A unified framework of five principles for AI in society. In *Machine Learning and the City : Applications in Architecture and Urban Design*. 2022.
Voir page 19.

[FRS20] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. SmartWatts : Self-Calibrating Software-Defined Power Meter for Containers. In *Proceedings - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*, 2020.
Voir page 119.

[Gab23] Gabriela RAMOS. Readiness assessment methodology. A tool of the Recommendation on the Ethics of Artificial Intelligence. Technical report, UNESCO, 2023.
Voir page 19.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Journal of Machine Learning Research*, volume 15, 2011.
Voir page 59.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
Voir pages 39 et 46.

[GCHZ17] Congcong Gong, Haisong Chen, Weixiong He, and Zhanliang Zhang. Improved multi-objective clustering algorithm using particle swarm optimization. *PLoS ONE*, 12(12), 2017.
Voir page 103.

[GFF+22] Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. SCALING LAWS FOR NEURAL MACHINE TRANSLATION. In *ICLR 2022 - 10th International Conference on Learning Representations*, 2022.
Voir page 53.

[GGK+19] Vijay Gadepally, Justin Goodwin, Jeremy Kepner, Albert Reuther, Hayley Reynolds, Siddharth Samsi, Jonathan Su, and David Martinez. AI Enabling Technologies : A Survey, 2019.
Voir pages 73 et 75.

[GHL+20] Cong Guo, Bo Yang Hsueh, Jingwen Leng, Yuxian Qiu, Yue Guan, Zehuan Wang, Xiaoying Jia, Xipeng Li, Minyi Guo, and Yuhao Zhu. Accelerating sparse dnn models without hardware-support via tile-wise sparsity. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, volume 2020-November, 2020.
Voir page 78.

[GKD+21] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference, 2021.
Voir pages 62 et 63.

[GKK22] Deepak Ghimire, Dayoung Kil, and Seong Heum Kim. A Survey on Efficient Convolutional Neural Networks and Hardware Acceleration, 2022.
Voir page 78.

[GT95] Yue Seng Goh and Eng Chong Tan. Pruning neural networks during training by backpropagation. In *IEEE Region 10's Annual International Conference, Proceedings*, volume 2, 1995.
Voir page 60.

[GWJ20] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Kronecker Attention Networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 229–237. Association for Computing Machinery, 8 2020.
Voir page 111.

[GZS+21] Amir Gholami, Yao Zhewei, Kim Sehoon, W. Mahoney Michael, and Keutzer Kurt. AI and Memory Wall, 2021.
Voir page 56.

[HABN+21] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning : Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22, 2021.
Voir page 62.

[HCAD+17] Franz C Heinrich, Alexandra Carpen-Amarie, Augustin Degomme, Sascha Hunold, Arnaud Legrand, Anne-Cécile Orgerie, and Martin Quinson. Predicting the Performance and the Power Consumption of MPI Applications With SimGrid. 2017.
Voir page 71.

[HCS+18] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks : Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18, 2018.
Voir page 62.

[HE75] Fitzgerald Hiram E. Energy Use in the Telecommunications Industry, 1975.
Voir page 26.

[HHR+20] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21, 2020.
Voir page 36.

[HLK+21] Seyedehfaezeh Hosseininoorbin, Siamak Layeghy, Brano Kusy, Raja Jurdak, and Marius Portmann. Exploring Deep Neural Networks on Edge TPU. *Internet of Things*, 22, 2021.
Voir page 75.

[HLM+17] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark Horowitz, and Bill Dally. Deep compression and EIE : Efficient inference engine on compressed deep neural network. *IEEE Hot Chips 28 Symposium*, 16, 2017.
Voir page 78.

[HLX+23] Shaoyi Huang, Bowen Lei, Dongkuan Xu, Hongwu Peng, Yue Sun, Mimi Xie, and Caiwen Ding. Dynamic Sparse Training via Balancing the Exploration-Exploitation Trade-off. In *Proceedings - Design Automation Conference*, 2023.
Voir page 53.

[HMD16] Song Han, Huizi Mao, and William J. Dally. Deep compression : Compressing deep neural networks with pruning, trained quantization and Huffman coding. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–14, 2016.
Voir pages 4, 57, 59, 78 et 106.

[Hod15] Hodson James. AI for Good, 2015.
Voir page 36.

[Hor14] Mark Horowitz. 1.1 Computing's energy problem (and what we can do about it). In *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, volume 57, 2014.
Voir pages 3, 51, 60 et 61.

[HPTD15] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems*, 2015-Janua :1135–1143, 2015.
Voir pages 57, 60 et 107.

[HSS12] Geoffrey E. Hinton, Nitish Srivastava, and Kevin Swersky. Neural Networks for Machine Learning Lecture 6a Overview of mini-batch gradient descent., 2012.
Voir page 48.

[HTJW21] Trevor Hastie, Robert Tibshirani, Gareth James, and Daniela Witten. An introduction to statistical learning (2nd ed.). *Springer texts*, 102, 2021.
Voir page 54.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015.
Voir pages 58 et 62.

[HZW15] Tianqiang Hu, Dajun Zhang, and Jinliang Wang. A meta-analysis of the trait resilience and mental health. *Personality and Individual Differences*, 76 :18–27, 2015.
Voir page 85.

[JA21] Fatimah Jasem and Manar AlSaraf. A Survey of Neural Network Hardware Accelerators in Machine Learning. *Machine Learning and Applications : An International Journal*, 8(4), 2021.
Voir page 76.

[JAZ+18] Delona C Johny, Anju J S Assistant, Ruirui Zhang, Xin Xiao, Nasir Rashid, Javaid Iqbal, Fahad Mahmood, Anam Abid, Umar S. Khan, Mohsin I. Tiwana, Zhang Fan, Chen Wen, Li Tao, Cao Xiaochun, Peng Haipeng, Chao Yang, Lin Jia, Bing Qiu Chen, Hai Yang Wen, Matthew D. Zeiler, John E. Hunt, Denise E. Cooke, Stephanie Forrest, Alan S Perelson, Lawrence Allen, and Rajesh Cherukuri. ADADELTA : An Adaptive Learning Rate Method. *IEEE Access*, 7(November) :51886–51898, 2018.
Voir page 48.

[JD18] John L. Hennessy and David A. Patterson. A new golden age for computer architecture : Domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018.
Voir pages 73 et 75.

[JDSM17] Patrick Judd, Alberto Delmas, Sayeh Sharify, and Andreas Moshovos. Cnvlutin2 : Ineffectual-Activation-and-Weight-Free Deep Neural Network Computing, 2017.
Voir pages 73, 74 et 75.

[JKC+18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.
Voir page 58.

[JPRW14] Ernest Jamro, Tomasz Pabis, Pawel Russek, and Kazimierz Wiatr. The algorithms for FPGA implementation of sparse matrices multiplication. *Computing and Informatics*, 33(3), 2014.
Voir page 73.

[JSRR21] Shubham Jain, Abhronil Sengupta, Kaushik Roy, and Anand Raghunathan. RxNN : A Framework for Evaluating Deep Neural Networks on Resistive Crossbars. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(2), 2021.
Voir page 77.

[JYP+17] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir, Rajendra Gottipati, William Gulland, Robert Hagmann, C Richard Ho, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon Mackean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-Datacenter Performance Analysis of a Tensor Processing Unit, 2017.
Voir page 74.

[JYPP18] Norman P. Jouppi, Cliff Young, Nishant Patil, and David Patterson. A domain-specific architecture for deep neural networks. *Communications of the ACM*, 61(9), 2018.
Voir pages 73 et 75.

[JYS+20] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT : Distilling BERT for Natural Language Understanding, 2020.
Voir page 56.

[JZDL18] Zhen Jia, Aleksandar Zlateski, Fredo Durand, and Kai Li. Optimizing N-dimensional, winograd-based convolution for manycore CPUs. *ACM SIGPLAN Notices*, 53(1), 2018.
Voir page 72.

[Kan17] David Kanter. Google TPU boosts machine learning. *Microprocessor Report*, 2017.
Voir page 75.

[Kar90] Ehud D. Karnin. A Simple Procedure for Pruning Back-Propagation Trained Neural Networks, 1990.
Voir page 60.

[KB15] Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 2015.
Voir page 47.

[KCK21] Jangho Kim, Simyung Chang, and Nojun Kwak. PQK : Model compression via pruning, quantization, and knowledge distillation. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 1863–1867, 2021.
Voir pages 3, 58 et 59.

[Kha20] Paresh Kharya. TensorFloat-32 in the A100 GPU Accelerates AI Training, HPC up to 20x, 2020.
Voir pages 3 et 61.

[KHN+18] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. RAPL in Action. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 3(2) :1–26, 6 2018.
Voir page 118.

[KJP21] Suchang Kim, Jihyuck Jo, and In Cheol Park. Hybrid convolution architecture for energy-efficient deep neural network processing. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 68(5), 2021.
Voir page 78.

[KMH+20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, 2020.
Voir pages 53 et 60.

[Kri09] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*, 2009.
Voir page 71.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553) :436–444, 2015.
Voir page 41.

[LGBI21] Loïc Lannelongue, Jason Grealey, Alex Bateman, and Michael Inouye. Ten simple rules to make your computing more environmentally sustainable, 2021.
Voir page 67.

[LGI21] Loïc Lannelongue, Jason Grealey, and Michael Inouye. Green Algorithms : Quantifying the Carbon Footprint of Computation. *Advanced Science*, 8(12), 2021.
Voir page 67.

[LGR+15] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 2015.
Voir page 58.

[LGW+21] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration : A survey. *Neurocomputing*, 461, 2021.
Voir page 62.

[LHFT20] Jongseok Lee, Matthias Humt, Jianxiang Feng, and Rudolph Triebel. Estimating Model Uncertainty of Neural Networks in Sparse Information Form, 2020.
Voir page 81.

[Lin21] Hsiao Ying Lin. Colors of Artificial Intelligence. *Computer*, 54(11), 2021.
Voir page 36.

[LLL23] Shiqing Li, Di Liu, and Weichen Liu Di Liu. Efficient FPGA-Based Sparse Matrix-Vector Multiplication With Data Reuse-Aware Compression. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(12), 2023.
Voir page 73.

[LLM23] Zhuo Li, Hengyi Li, and Lin Meng. Model Compression for Deep Neural Networks : A Survey, 2023.
Voir page 50.

[LLN+18] Sheng Lin, Ning Liu, Mahdi Nazemi, Hongjia Li, Caiwen Ding, Yanzhi Wang, and Massoud Pedram. FFT-based deep learning deployment in embedded systems. In *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition*, 2018.
Voir page 72.

[LLSD19] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the Carbon Emissions of Machine Learning, 2019.
Voir page 36.

[LPM15] Minh Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Conference Proceedings - EMNLP 2015 : Conference on Empirical Methods in Natural Language Processing*, 2015.
Voir page 62.

[LR02] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002 : The 19th International Conference on Computational Linguistics*, pages 1–7. Association for Computational Linguistics (ACL), 2002.
Voir page 103.

[LR06] Xin Li and Dan Roth. Learning question classifiers : The role of semantic information. *Natural Language Engineering*, 12(3), 2006.
Voir page 103.

[LSL+15] Minh Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, volume 1, 2015.
Voir page 49.

[LVF+20] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. FlauBERT : Unsupervised language model pre-training for French. In *LREC 2020 - 12th International Conference on Language Resources and Evaluation, Conference Proceedings*, 2020.
Voir page 88.

[LWF+15] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Penksy. Sparse Convolutional Neural Networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.
Voir page 59.

[LWTL17] Hehe Li, Jingge Wang, Miaomiao Tang, and Xinzhong Li. Deep Sparse Rectifier Neural Networks Xavier. *Journal of the Optical Society of America A : Optics and Image Science, and Vision*, 34(7), 2017.
Voir page 59.

[LYZ+22] Xing Liu, Jianfeng Yang, Chengming Zou, Qimei Chen, Xin Yan, Yuao Chen, and Chenran Cai. Collaborative Edge Computing with FPGA-Based CNN Accelerators for Energy-Efficient and Time-Aware Face Tracking System. *IEEE Transactions on Computational Social Systems*, 9(1), 2022.
Voir page 73.

[MA21] Athar Hussein Mohammed and Ali H. Ali. Survey of BERT (Bidirectional Encoder Representation Transformer) types. In *Journal of Physics : Conference Series*, volume 1963, 2021.
Voir page 12.

[MAAI+14] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197), 2014.
Voir pages 73 et 75.

[MGD20] Rahul Mishra, Hari Prabhat Gupta, and Tanima Dutta. A Survey on Deep Neural Network Compression : Challenges, Overview, and Solutions, 2020.
Voir page 50.

[MGH+20] Mayan Moudgill, John Glossner, Wei Huang, Chaoyang Tian, Chunxia Xu, Nianliang Yang, Lei Wang, Tailin Liang, Shaobo Shi, Xiaodong Zhang, Daniel Iancu, Gary Nacer, and Kerry Li. Heterogeneous Edge CNN Hardware Accelerator. In *12th International Conference on Wireless Communications and Signal Processing*, 2020.
Voir page 78.

[Mit97] Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL : McGraw Hill*, 45, 1997.
Voir page 38.

[MKS22] Diksha Moolchandani, Anshul Kumar, and Smruti R. Sarangi. Performance and Power Prediction for Concurrent Execution on GPUs. *ACM Transactions on Architecture and Code Optimization*, 19(3), 2022.
Voir page 71.

[MMS+18] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1), 2018.
Voir page 59.

[MPJS+14] Phitchaya Mangpo Phothilimthana, Tikhon Jelvis, Rohin Shah, Nishant Totla, Sarah Chasins, and Rastislav Bodik. Chlorophyll : Synthesis-aided compiler for low-power spatial architectures. In *ACM SIGPLAN Notices*, volume 49, pages 396–407. Association for Computing Machinery, 6 2014.
Voir pages 3 et 73.

[MPMF23] Giosué Cataldo Marinó, Alessandro Petrini, Dario Malchiodi, and Marco Frasca. Deep neural networks compression : A comparative survey and choice recommendations. *Neurocomputing*, 520, 2023.
Voir page 50.

[MSH+21] Nitheesh Kumar Manjunath, Aidin Shiri, Morteza Hosseini, Bharat Prakash, Nicholas R. Waytowich, and Tinoosh Mohsenin. An Energy Efficient EdgeAI Autoencoder Accelerator for Reinforcement Learning. *IEEE Open Journal of Circuits and Systems*, 2, 2021.

Voir page 76.

[MV19] Sparsh Mittal and Shraiysh Vaishay. A survey of techniques for optimizing deep learning on GPUs, 2019.

Voir page 72.

[MXL+23] Yukta Mehta, Rui Xu, Benjamin Lim, Jane Wu, and Jerry Gao. A Review for Green Energy Machine Learning and AI Services, 8 2023.

Voir page 50.

[MYZ+17] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. A Survey on Mobile Edge Computing : The Communication Perspective, 2017.

Voir page 85.

[NBC+20] Seyed Morteza Nabavinejad, Mohammad Baharloo, Kun Chih Chen, Maurizio Palesi, Tim Kogel, and Masoumeh Ebrahimi. An Overview of Efficient Interconnection Networks for Deep Neural Network Accelerators. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(3), 2020.

Voir page 71.

[NDE+18] Sharan Narang, Gregory Diamos, Erich Elsen, Paulius Micikevicius, Jonah Alben, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2018.

Voir pages 73 et 75.

[NJPS21] Meike Nauta, Annemarie Jutte, Jesper Provoost, and Christin Seifert. This Looks Like That, Because.. Explaining Prototypes for Interpretable Image Recognition. In *Communications in Computer and Information Science*, volume 1524 CCIS, 2021.

Voir page 49.

[NLB+18] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards Understanding the Role of Over-Parametrization in Generalization of Neural Networks. *ICLR 2019*, (2017), 2018.

Voir page 53.

[NYOdlT23] Jose Nunez-Yanez, Andres Otero, and Eduardo de la Torre. Dynamically reconfigurable variable-precision sparse-dense matrix acceleration in Tensorflow Lite. *Microprocessors and Microsystems*, 98, 2023.

Voir page 73.

[PBBP+17] Adrien Prost-Boucle, Alban Bourge, Frédéric Pétrot, Hande Alemdar, Nicholas Caldwell, Vincent Leroy, and al Scal. Scalable High-Performance Architecture for Convolutional Ternary Neural Networks on FPGA. *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–7, 2017.

Voir pages 73, 74 et 75.

[PK23] Eva Patel and Dharmender Singh Kushwaha. An Integrated Deep Learning Prediction Approach for Efficient Modelling of Host Load Patterns in Cloud Computing. *Journal of Grid Computing*, 21(1), 2023.

Voir page 78.

[PNI+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies - Proceedings of the Conference*, volume 1, 2018.

Voir page 49.

[PPA18] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

Voir pages 50, 62 et 63.

[PSS+20] Anh Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavský, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable Low-Rank Tensor Decomposition for Compression of Convolutional Neural Network. In

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 12374 LNCS, pages 522–539. Springer Science and Business Media Deutschland GmbH, 2020.
Voir page 103.

[PY10] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22 :1345–1359, 2010.
Voir pages 42 et 62.

[RDK+19] David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, Alexandra Luccioni, Tegan Maharaj, Evan D. Sherwin, S. Karthik Mukkavilli, Konrad P. Kording, Carla Gomes, Andrew Y. Ng, Demis Hassabis, John C. Platt, Felix Creutzig, Jennifer Chayes, and Yoshua Bengio. Tackling climate change with machine learning, 2019.
Voir page 20.

[RDK+23] David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, Alexandra Sasha Luccioni, Tegan Maharaj, Evan D. Sherwin, S. Karthik Mukkavilli, Konrad P. Kording, Carla P. Gomes, Andrew Y. Ng, Demis Hassabis, John C. Platt, Felix Creutzig, Jennifer Chayes, and Yoshua Bengio. Tackling Climate Change with Machine Learning, 2023.
Voir page 20.

[RGA19] Md Aamir Raihan, Negar Goli, and Tor M. Aamodt. Modeling Deep Learning Accelerator Enabled GPUs. In *Proceedings - 2019 IEEE International Symposium on Performance Analysis of Systems and Software*, 2019.
Voir page 71.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. *MIT Press, Cambridge, MA*, 1986.
Voir page 39.

[RMJ+20] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. Survey of Machine Learning Accelerators. *2020 IEEE High Performance Extreme Computing Conference, HPEC 2020*, pages 1–11, 2020.
Voir page 76.

[RN10] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach Third Edition.* 2010.
Voir page 37.

[RN21] Stuart Russell and Peter Norivg. Artificial Intelligence : A Modern Approach (Global Edition). *Artificial Intelligence : A Modern Approach*, 2021.
Voir page 37.

[RQD+23] Wei Qing Ren, Yu Ben Qu, Chao Dong, Yu Qian Jing, Hao Sun, Qi Hui Wu, and Song Guo. A Survey on Collaborative DNN Inference for Edge Intelligence. *Machine Intelligence Research*, 2023.
Voir page 78.

[RWC+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019.
Voir pages 49 et 88.

[SAM+18] Taiji Suzuki, Hiroshi Abe, Tomoya Murata, Shingo Horiuchi, Kotaro Ito, Tokuma Wachi, So Hirai, Masatoshi Yukishima, and Tomoaki Nishimura. Spectral-Pruning : Compressing deep neural network via spectral analysis. *Revista Paulista de Pediatria (English Edition)*, 2018.
Voir page 60.

[SAM+20] Taiji Suzuki, Hiroshi Abe, Tomoya Murata, Shingo Horiuchi, Kotaro Ito, Tokuma Wachi, So Hirai, Masatoshi Yukishima, and Tomoaki Nishimura. Spectral pruning : Compressing deep neural networks via spectral analysis and its generalization error. In *IJCAI International Joint Conference on Artificial Intelligence*, 2020.

Voir page 60.

[SB17] R S Sutton and A G Barto. Reinforcement learning : an introduction 2nd. *Neural Networks IEEE Transactions on*, 2017.
Voir page 42.

[SBF+19] Andrew D. Selbst, Danah Boyd, Sorelle A. Friedler, Suresh Venkatasubramanian, and Janet Vertesi. Fairness and abstraction in sociotechnical systems. In *FAT* 2019 - Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, 2019.
Voir page 17.

[Sch92] Jürgen Schmidhuber. Learning Factorial Codes by Predictability Minimization. *Neural Computation*, pages 863–879, 1992.
Voir page 108.

[SCYE17] Vivienne Sze, Yu Hsin Chen, Tien Ju Yang, and Joel S. Emer. Efficient Processing of Deep Neural Networks : A Tutorial and Survey, 2017.
Voir pages 3, 38, 76, 78 et 126.

[SCZ+16] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge Computing : Vision and Challenges. *IEEE Internet of Things Journal*, 2016.
Voir page 85.

[SDCW19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT : smaller, faster, cheaper and lighter, 2019.
Voir pages 58 et 63.

[SDSE20] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, pages 54–63, 2020.
Voir pages 36 et 50.

[Seb02] Fabrizio Sebastiani. Machine Learning in Automated Text Categorization, 2002.
Voir page 103.

[SEKH23] Benoit Steiner, Mostafa Elhoushi, Jacob Kahn, and James Hegarty. MODeL : Memory Optimizations for Deep Learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 32618–32632. PMLR, 2023.
Voir page 51.

[SGM20] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. *57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2020.
Voir pages 3, 7, 19, 35, 50 et 81.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.
Voir page 55.

[SKS+13] Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 6655–6659, 2013.
Voir pages 103, 104 et 106.

[SLGKAE20] Abu Sebastian, Manuel Le Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. Memory devices and applications for in-memory computing, 2020.
Voir page 51.

[SLL19] David R. So, Chen Liang, and Quoc V. Le. The Evolved Transformer, 2019.
Voir page 49.

[SMDH13] Ilya Sutskever, James Martens, George Dahl, and Georey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on International Conference on Machine Learning*, Proceedings of Machine Learning Research, page 1139–1147, 2013.

Voir page 48.

[SP23] Diego Sanmartín and Vera Prohaska. Exploration of TPUs for AI Applications, 2023.
Voir page 75.

[SPWL23] Yuanyuan Shen, Manman Peng, Qiang Wu, and Renfa Li. A machine learning method to variable classification in OpenMP. *Future Generation Computer Systems*, 2023.
Voir page 80.

[SWR20] Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning : Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 2020-December, 2020.
Voir page 60.

[SY21] Gurmail Singh and Kin Choong Yow. These do not Look like Those : An Interpretable Deep Learning Model for Image Recognition. *IEEE Access*, 2021.
Voir page 73.

[SYCE19] Vivienne Sze, Tien-ju Yang, Yu-hsin Chen, and Joel Emer. Efficient Processing of Deep Neural Networks : from Algorithms to Hardware Architectures. *NeurIPS*, page 138, 2019.
Voir page 53.

[SYS+20] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT : A compact task-agnostic BERT for resource-limited devices. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2020.
Voir page 63.

[Tib11] Robert Tibshirani. Regression shrinkage and selection via the lasso : A retrospective. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, pages 273–282, 2011.
Voir page 54.

[TLBH16] Youssef Tamaazousti, Hervé Le Borgne, and Céline Hudelot. Diverse concept-level features for multi-object classification. In *ICMR 2016 - Proceedings of the 2016 ACM International Conference on Multimedia Retrieval*, 2016.
Voir page 42.

[vEH20] Jesper E van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, pages 373–440, 2020.
Voir page 41.

[VKB18] Stylianos I. Venieris, Alexandros Kouris, and Christos Savvas Bouganis. Toolflows for mapping convolutional neural networks on FPGAS : A survey and future directions, 2018.
Voir page 73.

[VKB19] Stylianos I. Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. Toolflows for Mapping Convolutional Neural Networks on FPGAs. *ACM Computing Surveys*, 2019.
Voir pages 74 et 75.

[VSC23] Roberto Verdecchia, June Sallou, and Luís Cruz. A systematic review of Green AI, 2023.
Voir pages 19 et 50.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
Voir pages 3, 49, 89, 90, 91, 95 et 97.

[VTM+20] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention : Specialized heads do the heavy lifting, the rest can be pruned. In *57th Annual Meeting of the Association for Computational Linguistics*, 2020.
Voir page 61.

[WGDL21] Ran Wu, Xinmin Guo, Jian Du, and Junbao Li. Accelerating neural network inference on fpga-based platforms—a survey. *Electronics (Switzerland)*, 2021.
Voir page 73.

[WJK+12] Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra, and Shirley Moore. Measuring energy and power with PAPI. In *Proceedings of the International Conference on Parallel Processing Workshops*, 2012.
Voir page 65.

[WLCS18] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. In *6th International Conference on Learning Representations*, 2018.
Voir page 41.

[WLZ20] Yifan Wang, Chundian Li, and Chen Zeng. Exploring the performance bound of cambricon accelerator in end-to-end inference scenario. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12093 LNCS, 2020.
Voir page 73.

[WPY+23] Yuntao Wang, Yanghe Pan, Miao Yan, Zhou Su, and Tom H. Luan. A Survey on ChatGPT : AI-Generated Contents, Challenges, and Solutions. *IEEE Open Journal of the Computer Society*, 2023.
Voir page 73.

[XLC+21] Yongjun Xu, Xin Liu, Xin Cao, Changping Huang, Enke Liu, Sen Qian, Xingchen Liu, Yanjun Wu, Fengliang Dong, Cheng Wei Qiu, Junjun Qiu, Keqin Hua, Wentao Su, Jian Wu, Huiyu Xu, Yong Han, Chenguang Fu, Zhigang Yin, Miao Liu, Ronald Roepman, Sabine Dietmann, Marko Virta, Fredrick Kengara, Ze Zhang, Lifu Zhang, Taolan Zhao, Ji Dai, Jialiang Yang, Liang Lan, Ming Luo, Zhaofeng Liu, Tao An, Bin Zhang, Xiao He, Shan Cong, Xiaohong Liu, Wei Zhang, James P. Lewis, James M. Tiedje, Qi Wang, Zhulin An, Fei Wang, Libo Zhang, Tao Huang, Chuan Lu, Zhipeng Cai, Fang Wang, and Jiabao Zhang. Artificial intelligence : A powerful paradigm for scientific research, 2021.
Voir page 52.

[XWvD20] Patrick Xia, Shijie Wu, and Benjamin van Durme. Which *BERT ? A survey organizing contextualized encoders. In *Conference on Empirical Methods in Natural Language Processing*, 2020.
Voir page 12.

[YCF+23] Tim Yarally, Luís Cruz, Daniel Feitosa, June Sallou, and Arie Van Deursen. Uncovering Energy-Efficient Practices in Deep Learning Training : Preliminary Steps Towards Green AI. In *IEEE/ACM 2nd International Conference on AI Engineering - Software Engineering for AI*, 2023.
Voir page 53.

[YMC21] Tan Yigitcanlar, Rashid Mehmood, and Juan M. Corchado. Green artificial intelligence : towards an efficient, sustainable and equitable technology for smart cities and futures. *Sustainability (Switzerland)*, 2021.
Voir page 36.

[ZDZW20] Yipeng Zhang, Bo Du, Lefei Zhang, and Jia Wu. Parallel DNN Inference Framework Leveraging a Compact RISC-V ISA-based Multi-core System. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.
Voir page 71.

[ZHMD16] Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained Ternary Quantization, 2016.
Voir page 57.

[Zhu05] Xiaojin Zhu. Semi-Supervised Learning Literature Survey. *European Space Agency, (Special Publication) ESA SP*, 2005.
Voir page 41.

[Zhu08] Xiaojin Zhu. Semi-Supervised Learning Literature Survey Contents. *SciencesNew York*, 2008.
Voir page 41.

[ZNM+20] Rongpu Zhou, Jeffrey A. Newman, Yao-Yuan Mao, Aaron Meisner, John Moustakas, Adam D. Myers, Abhishek Prakash, Andrew R. Zentner, David Brooks, Yutong Duan, Martin Landriau, Michael E. Levi, Francisco Prada, and Gregory Tarle. The Clustering of DESI-like Luminous Red Galaxies Using Photometric Redshifts. *Monthly Notices of the Royal Astronomical Society*, page 3309–3331, 2020.

Voir page 61.

[ZWL18] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis : A survey. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 2018.

Voir page 103.

[ZYH+23] Junjie Zhang, Qiao Yin, Weicheng Hu, Yunfeng Li, Hu Li, Nan Ye, and Bingyao Cao. EPA : The effective pipeline architecture for CNN accelerator with high performance and computing efficiency based on FPGA. *Concurrency and Computation : Practice and Experience*, 2023.

Voir page 73.

[ZZGX19] Maohua Zhu, Tao Zhang, Zhenyu Gu, and Yuan Xie. Sparse tensor core : Algorithm and hardware co-design for vector-wise sparse neural networks on modern GPUs. In *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, 2019.

Voir page 78.

Vers une Intelligence Artificielle Numérique Responsable

Résumé : Le numérique, et en particulier l'Intelligence Artificielle (IA), est de plus en plus reconnu comme une source de pollution en raison de sa consommation énergétique importante. Cette augmentation de la consommation d'énergie est attribuée à la fois au matériel nécessaire aux solutions d'IA et à l'explosion des services basés sur l'IA, ainsi qu'à l'énorme demande de données. Historiquement, la consommation d'énergie était principalement associée à l'efficacité matérielle, mais désormais, le composant logiciel joue un rôle tout aussi important. Avec l'avènement du Cloud Computing et le passage à des infrastructures hébergées dans le Cloud, de nombreuses entreprises négligent l'optimisation énergétique de leurs solutions d'IA, en se reposant sur des ressources facilement scalables sans tenir compte de l'efficacité énergétique. Cette thèse vise à sensibiliser et à mettre en œuvre des solutions d'IA plus responsables pour réduire la consommation d'énergie et proposer des services plus économes en énergie. Pour ce faire, elle commence par une revue complète de l'évolution des solutions d'IA, mettant en évidence les différents aspects essentiels pour des pratiques numériques responsables, notamment en matière d'IA. La mesure et le suivi de la consommation d'énergie sur l'ensemble du cycle de vie des solutions d'IA posent un défi, mais une fois réalisés, les efforts d'analyse et d'optimisation peuvent commencer pour créer des solutions numériques plus écoénergétiques. Cette thèse propose des solutions concrètes aux entreprises pour rendre l'utilisation des applications d'IA plus responsable, sans nécessiter de changements majeurs dans l'infrastructure matérielle.

Mots clés : Intelligence Artificielle, Deep Learning, NLP, Numérique Responsable, Énergie, Consommation, Durable

Towards Sustainability Artificial Intelligence

Abstract: The digital sphere, particularly Artificial Intelligence (AI), is increasingly recognized as a source of pollution due to its significant energy consumption. This surge in energy usage is attributed to both the hardware required for AI solutions and the explosion of AI-based services, coupled with the immense data demands. Historically, energy consumption was primarily associated with hardware efficiency, but now, the software component plays an equally significant role. With the rise of Cloud Computing and the shift towards Cloud-hosted infrastructures, many companies overlook energy optimization in their AI solutions, relying on easily scalable resources without considering energy efficiency. This thesis aims to promote awareness and implementation of more responsible AI solutions to reduce energy consumption and offer more energy-efficient services. To achieve this, it begins with a comprehensive review of AI solution evolution, highlighting the various aspects essential for responsible digital practices, especially in AI. Measurement and tracking of energy consumption across the entire life cycle of AI solutions pose a challenge, but once achieved, analysis and optimization efforts can begin to create more energy-efficient digital solutions. This thesis provides concrete solutions for enterprises to make AI application usage more responsible, without necessitating major hardware changes.

Keywords: Artificial Intelligence, Deep Learning, NLP, Responsible Digital, Energy, Consumption, Sustainable

Laboratoire Informatique, Image, Interaction
Institut LUDI - La Rochelle Université
Avenue Michel Crépeau

17042 LA ROCHELLE CEDEX 1