

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE  
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE  
UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE



FACULTÉ D'INFORMATIQUE  
DÉPARTEMENT D'INTELLIGENCE ARTIFICIELLE

MEMOIRE DE MASTER

SPÉCIALITÉ  
SYSTÈMES INFORMATIQUES INTELLIGENTS

### THÈME

**Analyse et compréhension automatique de documents pour un problème de classification**

Sujet Proposé par :  
**MELAINE SAMY, CTO, Eden AI**

Présenté par :  
**BOULMAALI Linda Imene**  
**FLICI Syrine Mahdia**

Encadré par :  
**Mme. DELLAL-HEDJAZI BADIÂA**

Soutenu le : 03/07/2023

Devant le jury composé de :

- Mme. KHELLAF FAIZA                      Président
  - Mme. BOUCHENE SABRINA                Membre
- Projet N°11 SII/2023

# **Remerciements**

*Nous remercions Dieu de nous avoir donné la santé et le courage afin de concevoir ce travail.*

*Tout d'abord, nous exprimons notre gratitude envers Mr. Samy MELAINE pour nous avoir proposé ce sujet et nous avoir accompagnés tout au long de notre travail. Nous remercions également Mr. Jérémy LAMBERT pour nous avoir généreusement consacré son temps, son énergie et nous avoir guidés dans la réalisation de notre projet.*

*Nos remerciements vont également à Mme B. DELAL pour ses précieux conseils qui ont grandement contribué à la réussite de ce travail.*

*Nous souhaitons saisir cette occasion pour exprimer notre gratitude envers les membres du jury qui nous ont honorés en acceptant de juger ce travail.*

*De plus, il est important pour nous de témoigner notre reconnaissance envers la communauté open source qui travaille inlassablement pour maintenir la majorité des outils que nous utilisons aujourd'hui.*

*Nous souhaitons également remercier Naila HOUACINE et Aymen KHOUAS, dont les orientations nous ont permis de mener à bien ce travail.*

*Nous ne saurions oublier de remercier l'ensemble des enseignants ayant contribué à notre formation ainsi que de leur suivi tout au long de nos années d'études.*

*Enfin, nous remercions nos familles et amis respectifs pour leurs encouragements et leur soutien qu'ils nous ont apporté.*

# *Dédicaces*

*À mes très chers parents, pour leur amour, leur confiance, leurs prières et pour leur soutien indéfectible tout au long de mes études, que Dieu leur accorde une longue vie et les préserve en bonne santé.*

*À mes très chères soeurs et frères ainsi que mon beau-frère, pour leur amour et soutien inconditionnel à qui je souhaite toute la réussite et le bonheur du monde.*

*À ma très chère tante, à qui je suis reconnaissante pour son amour et affection.*

*À ma chère amie Farida BOUACHI, pour son soutien et son guidage.*

*À mon amie et binôme Béline, qui a été toujours là dans les meilleurs moments comme dans les pires tout au long de cette année qui restera inoubliable pour nous deux. Je te souhaite une réussite éclatante et une profusion de bonheur.*

*À mes chers amis et tous ceux que j'aime et qui m'aiment.*

*Syrine*

*À ma chère mère, qui a été mon enseignante durant mes années de primaire et qui continue à façonner la personne que je suis aujourd'hui.*

*À mon cher père, pour son amour inconditionnel et sa confiance en moi, À mon frère et ma sœur, pour leur soutien inestimable et leurs encouragements constants, Et à mon petit neveu adorable, qui apporte le sourire à tous ceux qui l'entourent.*

*Je dédie également ce travail à mes chères tantes et mon cher oncle, qui sont toujours là pour moi.*

*Je tiens à remercier chaleureusement ma très chère binôme et partenaire, Syrine, qui a travaillé dur tout au long du projet. Travailler avec elle a été une expérience agréable qui restera gravée dans ma mémoire.*

*À mes chers amis que j'adore et qui m'aiment, je les remercie pour leur soutien moral.*

*Beline*

# Table des matières

<b>Introduction Générale . . . . .</b>	<b>1</b>
<b>1 Chapitre 1 Présentation de l'organisme d'accueil . . . . .</b>	<b>3</b>
1.1 Introduction . . . . .	3
1.2 Présentation de l'entreprise DataGenius . . . . .	3
1.3 Le service R&D . . . . .	3
1.4 Analyse de l'existant . . . . .	3
1.4.1 Présentation de la plateforme EdenAI . . . . .	4
1.4.2 Utilisation de la plateforme . . . . .	4
1.4.3 Présentation des fournisseurs . . . . .	5
1.4.4 Présentation des API OCR de la plateforme . . . . .	6
1.5 Analyse du besoin . . . . .	6
1.6 Conclusion . . . . .	6
<b>2 Chapitre 2 État de l'art . . . . .</b>	<b>7</b>
2.1 Introduction . . . . .	7
2.2 Analyse de documents . . . . .	7
2.2.1 Technique d'analyse des images . . . . .	8
2.2.1.1 CNN . . . . .	8
2.2.1.2 Optical Character Recognition (OCR) . . . . .	9
2.2.2 Technique d'analyse du texte . . . . .	10
2.2.2.1 Analyse syntaxique . . . . .	10
2.2.2.2 Représentation du texte . . . . .	10
2.2.2.3 Réseaux de Neurones Récurrents . . . . .	11
2.2.3 Les techniques de pointes de NLP et de vision par ordinateur . . . . .	12
2.2.3.1 Les Encodeurs et les Décodeurs . . . . .	12
2.2.3.2 Mécanisme d'attention . . . . .	13
2.2.3.3 Architecture des transformers . . . . .	14
2.2.3.4 Transformers de vision (ViTs) . . . . .	14
2.2.3.5 Mécanisme d'attention spatiale . . . . .	15
2.2.4 Algorithmes d'optimisation . . . . .	16
2.2.4.1 Stochastic Gradient Descent . . . . .	16
2.2.4.2 Adaptive Moment estimation (Adam) . . . . .	17
2.3 Compréhension automatique de document . . . . .	18
2.3.1 Modèles pré-entraînés . . . . .	18
2.3.2 Fine-tuning sur les tâches en aval . . . . .	21
2.4 Classification automatique de document . . . . .	22
2.4.1 Approche visuelle . . . . .	22
2.4.1.1 VGG (Visual Geometry Group) . . . . .	22

## Table des matières

---

2.4.1.2	ResNet (Residual Network) . . . . .	22
2.4.1.3	DocXClassifier, 2022 . . . . .	24
2.4.2	Approche textuelle . . . . .	24
2.4.2.1	Extraction des éléments pertinents . . . . .	24
2.4.2.2	Classification basée sur les règles de reconnaissance du texte . .	24
2.4.2.3	Classification basée sur les techniques de machine learning . . .	24
2.4.2.4	Classification basée sur des modèles pré-entraînés . . . . .	25
2.5	Conclusion . . . . .	26
<b>3 Chapitre 3 Conception</b>	. . . . .	<b>27</b>
3.1	Introduction . . . . .	27
3.2	La collecte des données . . . . .	27
3.3	Exploration des données . . . . .	29
3.3.1	Analyse des données . . . . .	29
3.3.2	Nettoyage des données . . . . .	29
3.3.3	Augmentation des données . . . . .	29
3.4	Solution basée sur les approches visuelles . . . . .	30
3.4.1	ResNet . . . . .	30
3.4.2	Attention ResNet-34 . . . . .	31
3.4.3	Entraînement . . . . .	33
3.5	Solution basée sur la combinaison des approches visuelles et textuelles . . . . .	33
3.5.1	Pipeline . . . . .	34
3.5.2	Modèle de vision . . . . .	35
3.5.3	OCR . . . . .	35
3.5.4	Modèles de NLP . . . . .	36
3.5.4.1	Modèle de machine learning . . . . .	36
3.5.4.2	Modèles pré-entraînés . . . . .	38
3.6	Solution basée sur un modèle free OCR . . . . .	41
3.6.1	Pipeline . . . . .	41
3.6.2	Pré-Traitement . . . . .	42
3.6.3	Entrainement . . . . .	42
3.7	Conception de l'application . . . . .	42
3.8	Conclusion . . . . .	43
<b>4 Chapitre 4 Implémentation et évaluations</b>	. . . . .	<b>44</b>
4.1	Introduction . . . . .	44
4.2	Implémentation de nos modèles . . . . .	44
4.2.1	Outils . . . . .	44
4.2.2	Caractéristiques de la machine locale . . . . .	45
4.2.3	Exploration des données visuelles . . . . .	45
4.2.3.1	Analyse des données . . . . .	45
4.2.3.2	Nettoyage des données . . . . .	47
4.2.3.3	Augmentation des données . . . . .	47
4.2.4	Implémentation de la première approche . . . . .	48
4.2.5	Implémentation de la deuxième approche . . . . .	48
4.2.5.1	Extraction et exploration des données textuelle . . . . .	48
4.2.5.2	Modèle de machine learning . . . . .	51
4.2.5.3	Modèles pré-entraînés . . . . .	52

---

## *Table des matières*

---

4.2.6	Implémentation de la troisième approche . . . . .	53
4.3	Evaluation . . . . .	53
4.3.1	Métriques d'évaluation d'un problème de classification . . . . .	53
4.3.2	Evaluation de nos différentes solutions . . . . .	54
4.3.2.1	Première approche . . . . .	55
4.3.2.2	Deuxième approche . . . . .	59
4.3.2.3	Troisième approche . . . . .	61
4.3.3	Résumé global des métriques . . . . .	62
4.4	Implémentation de l'application web . . . . .	65
4.4.1	Outils . . . . .	65
4.4.2	Architecture générale . . . . .	66
4.4.3	Backend . . . . .	66
4.4.4	Frontend . . . . .	66
4.4.5	Déploiement . . . . .	68
<b>Conclusion et Perspectives . . . . .</b>		<b>69</b>
<b>Webographie . . . . .</b>		
<b>Bibliographie . . . . .</b>		

# Table des figures

1.1	Logo de DataGenius [1] . . . . .	3
1.2	Logo de Eden AI [2] . . . . .	4
1.3	Fonctionnement des services d'EdenAI [2] . . . . .	5
2.1	Architecture de CNN [3] . . . . .	8
2.2	Architecture de RNN [4] . . . . .	11
2.3	Architecture de Transformer [33] . . . . .	14
2.4	Aperçu du modèle ViT [34] . . . . .	15
2.5	Architecture de layoutLM . . . . .	19
2.6	Schéma représentant le fine tuning . . . . .	21
2.7	Schéma représentant le bloc Résiduel [35] . . . . .	23
3.1	Diagramme représentant l'objectif du système . . . . .	27
3.2	Représentation du contenu de notre dataset . . . . .	28
3.3	Architecture de ResNet-50 . . . . .	30
3.4	Architecture d'un bloc résiduel pour ResNet-50 . . . . .	31
3.5	Architecture d'un bloc résiduel pour ResNet-34 . . . . .	32
3.6	Architecture de ResNet-34 modifiée . . . . .	32
3.7	Architecture de Attention ResNet . . . . .	33
3.8	Pipeline de la 2ème approche . . . . .	34
3.9	Shéma détaillé de notre seconde approche . . . . .	35
3.10	Freezing des embeddings de position . . . . .	40
3.11	Pipeline du modèle Donut . . . . .	42
3.12	Diagramme de séquence de notre application . . . . .	43
4.1	Graphe montrant le nombre des images par classe . . . . .	46
4.2	Graphe montrant la taille moyenne des images dans chaque classe . . . . .	46
4.3	Graphe montrant le nombre moyen des pixels dans les images de chaque classe .	47
4.4	Passeport annoté avec un cadre de détection de texte . . . . .	49
4.5	Nombre moyen de mots dans chaque classe . . . . .	50
4.6	Nuage de mots illustrant les mots les plus fréquent dans chaque classe . . . . .	50
4.7	Graphiques représentant les mots et leur fréquence dans chaque paire de classes	51
4.8	Matrice de confusion . . . . .	53
4.9	L'historique de la perte d'entraînement et d'évaluation de Resnet-50 . . . . .	56
4.10	L'historique de l'accuracy d'entraînement et d'évaluation de Resnet-50 . . . . .	56
4.11	L'historique de la perte d'entraînement et d'évaluation de Attention ResNet-34 .	57
4.12	L'historique de l'accuracy d'entraînement et d'évaluation de Attention ResNet-34	58
4.13	Évaluation F1-score des modèles NLP . . . . .	61

## *Table des figures*

---

4.14	Graphe de F-score de chaque classe pour chaque modèle . . . . .	62
4.15	Graphe du temps d'exécution et F-score pour chaque modèle . . . . .	63
4.16	Graphe du temps d'exécution et F-score Tesseract et GoogleOCR . . . . .	65
4.17	Exemple de classification . . . . .	66
4.18	Exemple de résultat d'un fournisseur choisi par l'utilisateur . . . . .	67
4.19	Exemple de la fenêtre d'évaluation des modèles pour un document donné . . . . .	67

# Liste des tableaux

3.1	Exemple d'une représentation Word2Vec . . . . .	37
4.1	Caractéristiques de la machine locale . . . . .	45
4.2	Paramètres d'entraînement de la première approche . . . . .	48
4.3	Tableau représentatif des fréquences du texte . . . . .	49
4.4	Paramètres de Word2Vec . . . . .	52
4.5	Paramètres d'entraînement de LinearSVM . . . . .	52
4.6	Paramètres d'entraînement des modèles pré-entraînés . . . . .	52
4.7	Comparaison de la taille de nos différents modèles . . . . .	55
4.8	Comparaison du temps d'exécution de nos différents modèles . . . . .	55
4.9	Métriques d'évaluation. ResNet-50 . . . . .	57
4.10	Métriques d'évaluation. Attention ResNet-34 . . . . .	58
4.11	Métriques d'évaluation pour chaque classe dans différents modèles de vision . . . . .	59
4.12	Performance générale des modèles . . . . .	59
4.13	Métriques d'évaluation pour chaque classe dans différents modèles NLP . . . . .	60
4.14	Performance générale des modèles . . . . .	60
4.15	Métriques d'évaluation. Donut . . . . .	61
4.16	Métriques d'évaluation des classes par rapport aux transformers avant le fine tune . . . . .	64
4.17	Comparaison de la performance entre la solution utilisant Tesseract et la solution utilisant GoogleOCR . . . . .	64

# Liste des acronymes

**API** Application Programming Interface. 4, 5, 48

**CNN** Convolutional Neural Networks. 8, 10, 12, 19, 22

**CRNN** Convolutional Recurrent Neural Networks. 10

**IA** Intelligence artificielle. 1, 4, 5

**MLM** Modèle de Langage Masqué. 25

**NLP** Natural Language Processing. 7, 12, 15, 18, 24, 25, 41, 44, 45, 69

**OCR** Optical Character Recognition. 1, 5–7, 18, 20, 24, 26, 44, 48, 55, 69

**ResNet** Residual Neural Network. 22, 30

**RNN** Recurrent Neural Networks. 10, 12

**VGG** Visual Geometry Group, une architecture complexe de CNNVisual Geometry Group,  
une architecture complexe de CNN. 22

**ViT** Vision Transformer. 15, 24

# Introduction Générale

Avec l'émergence des solutions apportées par l'intelligence artificielle, de plus en plus d'industries découvrent les avantages de l'IA et ses multiples domaines d'application. Cet intérêt pour l'IA a suscité l'engagement d'une communauté de développeurs, open source<sup>1</sup> et non open source, à contribuer au développement de modèles encore plus performants et innovants. Cela a rendu les services d'automatisation plus accessibles à un plus grand nombre de personnes et d'entreprises.

Souvent, les tâches manuelles que l'entreprise vise à automatiser, concernent l'exploitation des données collectées par cette dernière. En effet, les entreprises se retrouvent généralement avec plusieurs types de données, des documents et des images, qui sont analysées et traitées manuellement par des employés. Les entreprises s'orientent donc vers les systèmes automatiques, en vue de leur performance à exécuter des tâches répétitives plus rapidement et avec moins d'erreurs. Cela permet aux employés de se concentrer sur des tâches plus importantes, entraînant ainsi une augmentation de la productivité globale de l'entreprise et la réduction d'usage de ressources. De plus, les systèmes automatisés peuvent appliquer des règles et des processus de manière uniforme à chaque exécution, garantissant des résultats cohérents.

La compréhension automatique de document est l'un des domaines les plus convoités dans la réalisation des systèmes automatiques au sein d'une entreprise. Fortement influencé par l'évolution des LLM(Large Language Models)<sup>2</sup>, ce domaine est à nouveau au sommet des tendances actuelles avec divers modèles entraînés sur de vastes quantités de données textuelles provenant d'Internet. Certains de ces modèles arrivent à comprendre le contenu des documents de manière automatique et donc l'utiliser pour des tâches telles que la catégorisation, la reconnaissance optique de caractères (OCR), le résumé automatique ou la traduction.

Avec la popularité croissante des applications SaaS<sup>3</sup>. Les services alimentés par l'IA, fournis via des API<sup>4</sup>, permettent aux entreprises d'exploiter des services avec une technologie de pointe, sans avoir à construire des modèles complexes à partir de zéro et sans la nécessité d'avoir une

---

1. Open source : se réfère à un type de programme dont le code source est accessible librement, permettant à toute personne de le consulter, de le modifier et de le redistribuer.

2. LLM(Large Language Models) : sont des algorithmes d'apprentissage profond qui peut effectuer plusieurs tâches sur la base des connaissances acquises à partir d'ensembles de données massifs. lien : <https://blogs.nvidia.com/blog/2023/01/26/what-are-large-language-models-used-for/>

3. Software as a service (SaaS) permet aux utilisateurs de se connecter et d'utiliser des applications via Internet. lien : <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas/>

4. API : API signifie Interface de Programmation d'Application. Les API sont des mécanismes qui permettent à deux composants logiciels de communiquer entre eux en utilisant un ensemble de définitions et de protocoles.

## *Introduction Générale*

---

expertise étendue en IA. Cette relation étroite entre les API et les systèmes automatiques permet une intégration fluide et facilite l'adoption de fonctionnalités avancées basées sur l'IA dans différents domaines et applications.

Dans le cadre de notre stage au sein de l'entreprise EdenAI, et de notre projet de fin d'étude, nous avons pour objectif de créer un service permettant de classifier plusieurs types de documents de manière automatique afin de les envoyer à l'API adéquate à leur type.

Notre travail se résume à :

- La collecte de documents de différents types, analyse et traitement de ces documents.
- Le développement d'une solution capable de classifier différents types de documents.
- L'évaluation des approches et solutions utilisées.
- La création d'une API intégrée à la plateforme de l'entreprise, permettant d'interagir avec notre solution la plus optimale.

Notre mémoire est organisé comme suit :

- **Chapitre 1 - Présentation de l'organisme d'accueil** : nous présentons l'entreprise et la plateforme EdenAI, ainsi que les besoins qui ont mené à ce projet.
- **Chapitre 2 - Etat de l'art** : Nous présentons dans cette partie du projet, les différentes méthodes existantes et les avancées techniques dans le domaine de la compréhension et classification de documents.
- **Chapitre 3 - Conception** : Nous présentons la conception de nos différentes approches et méthodes.
- **Chapitre 4 - Implémentation et évaluations** : Dans ce dernier chapitre, nous présentons nos résultats finaux ainsi que les détails de notre implémentation.

# Chapitre 1

## Présentation de l'organisme d'accueil

### 1.1 Introduction

Pour mener à bien la création d'un système informatique, il est impératif de comprendre l'environnement de travail dans lequel il sera utilisé. Cette étape revêt une importance primordiale, car elle permet de cerner les besoins spécifiques d'une entreprise. Pour cela, nous dédions ce chapitre à la présentation de l'organisme d'accueil ainsi que des différents services qu'il offre.

### 1.2 Présentation de l'entreprise DataGenius

DataGenius est une entreprise française spécialisée en Data Science et Intelligence Artificielle qui propose des services externes afin d'aider les différentes entreprises dans la mise en œuvre de leurs projets et à exploiter leurs données en bénéficiant des avancées technologiques en matière de Big data. [1]



FIGURE 1.1 – Logo de DataGenius [1]

### 1.3 Le service R&D

Le service R&D (Recherche et Développement) est responsable de la recherche, du développement de nouveaux produits et à l'amélioration des services ou technologies existants pour répondre au besoin du marché et satisfaire les besoins du client.

### 1.4 Analyse de l'existant

Nous allons à présent voir l'analyse de l'existant, où nous examinerons la plateforme développée par l'organisme d'accueil qui décrit les divers services offerts.

### 1.4.1 Présentation de la plateforme EdenAI

Eden AI est une plateforme qui simplifie l'utilisation et l'intégration des technologies d'IA en fournissant une API unique qui donne accès aux meilleures API d'IA et à une plate-forme de gestion puissante [2].



FIGURE 1.2 – Logo de Eden AI [2]

Eden AI couvre une large gamme de technologies d'IA :

**Image** : Eden AI permet de réaliser et d'utiliser des technologies d'analyse d'images comme la détection des objets, la reconnaissance faciale, logo détection etc.

**Texte et NLP** : Eden AI permet de réaliser et d'utiliser plusieurs technologies du traitement du langage naturel, comme l'analyse des sentiments, question/réponse, reconnaissance d'entité nommée etc.

**Discours et audio** : Eden AI regroupe un large éventail de fonctionnalités de traitement audio notamment dans le speech-to-text.

**OCR et analyse de document** : Eden AI donne accès à de nombreuses technologies OCR, comme les analyseurs de facture, CV, passeport etc.

**Vidéo** : L'analyse vidéo est une fonctionnalité qui permet d'extraire des informations et des informations à partir de fichiers vidéo. Il est conçu pour être asynchrone, ce qui signifie que le processus d'analyse se produit en arrière-plan et permet d'accéder au résultat lorsqu'il est prêt.

### 1.4.2 Utilisation de la plateforme

Pour toutes les technologies proposées, EdenAI fournit de nombreuses sous-fonctionnalités. Pour toutes les sous-fonctionnalités, elle fournit un seul point de terminaison : le fournisseur de services n'est qu'un paramètre qui peut être modifié très facilement. Pour chacune de ces technologies, différents fournisseurs sont disponibles, comme Amazon, Google, Microsoft, PicPurify et Clarifai etc.

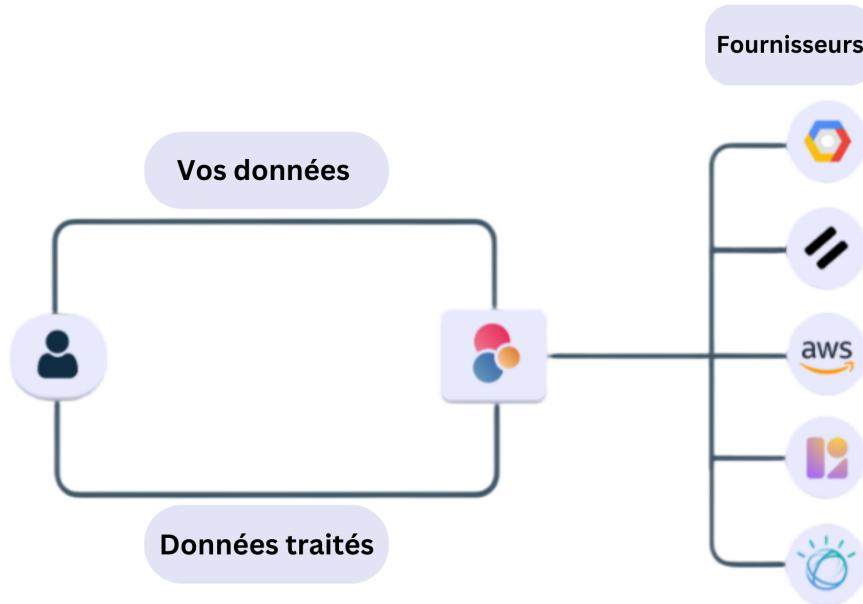


FIGURE 1.3 – Fonctionnement des services d’EdenAI [2]

Les utilisateurs de la plateforme sont souvent des gens qui ont un minimum de connaissance dans le domaine de la programmation. Un utilisateur doit tout d’abord créer un compte pour pouvoir accéder aux outils présents sur la plateforme, comme par exemple les APIs OCR. Une fois que c’est fait, le client peut récupérer la clé d’authentification et l’utiliser pour accéder à l’API voulu.

La documentation de EdenAI facilite la tâche aux utilisateurs les moins expérimentés en programmation, en leur fournissant un exemple de code pour accéder et utiliser l’API, ainsi qu’une petite interface pour pouvoir faire des tests sur l’API avant de choisir le fournisseur le plus adéquat au besoin du client, en termes de résultat, de prix et du temps d’exécution.

### 1.4.3 Présentation des fournisseurs

Comme mentionné précédemment, EdenIA est une plateforme regroupant diverses API d’IA proposées par différents fournisseurs, dont les plus populaires :

**Amazon** propose des différents services qui sont préformés d’AWS fournissant une intelligence prête à l’emploi, tel que : LookOut for vision, Textract, Transcribe etc. [5]

**Google Cloud** donne l’accès à ses différents services d’IA à partir de ses API. Parmi ses services les plus connus : VisionAI, speech-to-text etc. [6]

**Microsoft** propose un service de cognitive computing via son service cloud Azure, mentionnant : speech translation, Azure Cognitive Services for Vision etc.[7]

**OpenAI** propose des API qui ont l’accès aux modèles pré-entraînés les plus récents pour les différentes tâches, comme la conversion de la parole en texte, la traduction etc. [8]

En parallèle à ces grandes entreprises, EdenIA offre l’accès à diverses API d’entreprises spécialisées dans des tâches spécifiques telles que l’extraction de données à partir des images (Affinda ou Dataleon), la détection des différents objets (Clarifai, Api4AI) et la conversion de la parole en texte (IBM).

#### 1.4.4 Présentation des API OCR de la plateforme

Dans le cadre de ce projet, nous allons nous concentrer sur les sous-fonctionnalités du service OCR offerts par l'entreprise, en décrivant les différentes APIs OCR disponibles sur la plateforme :

**OCR générique** La reconnaissance optique de caractères ou lecteur optique de caractères (OCR) est la conversion électronique ou mécanique d'images de texte dactylographié, manuscrit ou imprimé en texte codé par machine, qu'il s'agisse d'un document numérisé ou d'une photo d'un document.

**Invoice Parser** L'API OCR Invoice Parser permet aux utilisateurs de prendre des factures dans une variété de formats et de renvoyer des données structurées pour automatiser le traitement des factures.

**Analyseur de CV** Permet aux clients d'analyser les CV et de renvoyer une représentation structurée du profil et des expériences du candidat.

**Receipt Parser** Permet aux clients d'extraire des informations d'un reçu : produits achetés, quantité, prix, date, montant de la TVA, etc.

**Table OCR** Permet aux clients d'analyser des documents contenant des tables et de retourner une représentation structurée desdites tables sous la forme d'un objet Json.

**OCR ID/Passport** Extraire automatiquement les informations d'un passeport. Il renverra JSON avec toutes les données associées, y compris le nom, la date de naissance, la nationalité et plus encore.

### 1.5 Analyse du besoin

L'entreprise EdenAI connaît une croissance continue et la demande d'utilisation de ses services ne cesse d'augmenter. Parmi les services les plus demandés sur la plateforme figurent les différentes API OCR, comme décrit dans la section précédente.

Souvent, l'entreprise est confrontée à des clients qui possèdent un ensemble de documents de types différents tels que des factures, des CV, des passeports, etc. Même si l'entreprise propose une API OCR pour chaque type de document, le client est continuellement contraint de les classifier manuellement si ces documents ne sont pas pré-étiquetés. Cette tâche peut être fastidieuse, surtout avec un grand ensemble de données.

Pour remédier à ce problème, EdenAI envisage d'intégrer une solution automatique d'analyse et de compréhension de documents. Cette solution permettra de classifier les documents en fonction de leur type et de les transférer vers l'API OCR appropriée.

### 1.6 Conclusion

Dans ce chapitre, nous avons réalisé une analyse de l'existant concernant l'entreprise DataGinus et le service qu'elle propose via la plateforme EdenAI. L'objectif était de mieux comprendre les besoins de l'entreprise afin de proposer des solutions adaptées à la problématique posée.

# Chapitre 2

## État de l'art

### 2.1 Introduction

La compréhension automatique de document a connu une très grande tendance ces dernières années, notamment avec l'arrivée des transformers. De nouvelles méthodes en vision par ordinateur et en NLP se sont développées, particulièrement pour les tâches de classification de documents.

La vision par ordinateur ou computer vision (CV) en anglais se concentre sur la compréhension et l'analyse d'images et de vidéos à partir de données numériques. Elle peut être utilisée pour identifier des objets, des personnes ou des anomalies dans des images ou des vidéos, ainsi que pour le suivi des mouvements ou l'extraction des informations ou de texte à partir des images, comme un OCR.

Le NLP (Natural Language Processing) ou le traitement du langage naturel se concentre sur la compréhension et la génération du langage naturel, il est utilisé dans plusieurs tâches, dont la classification du texte, la traduction automatique et l'analyse des sentiments.

Dans ce chapitre, nous allons voir les méthodes avancées, utilisées pour la compréhension et la classification automatique de document. Cependant, nous allons commencer par l'analyse de documents pour avoir un aperçu sur les connaissances requises aux problèmes nécessitant des solutions dans la vision par ordinateur et le traitement du langage naturel.

### 2.2 Analyse de documents

La classification des images est un domaine de recherche actif dans le domaine de la vision par ordinateur depuis plusieurs décennies. Cependant, cette tâche devient plus complexe lorsqu'il s'agit de documents, et les techniques de vision par ordinateur peuvent rencontrer des difficultés pour y faire face.

L'analyse de documents est un processus fréquemment utilisé pour les tâches de classification. Elle examine, à la fois, l'aspect textuel et visuel d'un document. Dans ce contexte, plusieurs solutions exploitent le domaine de vision par ordinateur et le domaine du traitement du langage naturel pour le même objectif.

Dans cette partie, nous allons aborder les méthodes les plus importantes pour réaliser une analyse de documents dans un contexte général.

### 2.2.1 Technique d'analyse des images

Nous allons explorer certains aspects clés de l'analyse des images, en mettant l'accent sur leur pertinence dans le traitement des documents.

#### 2.2.1.1 CNN

Un CNN (Convolutional Neural Network), ou Réseau Neuronal Convolutif en français, est un type de réseau de neurones artificiels conçu pour le traitement des données, telles que les images, les vidéos et les signaux sonores. Les CNN constituent de 4 couches qu'on peut résumer ci-dessous :[9]

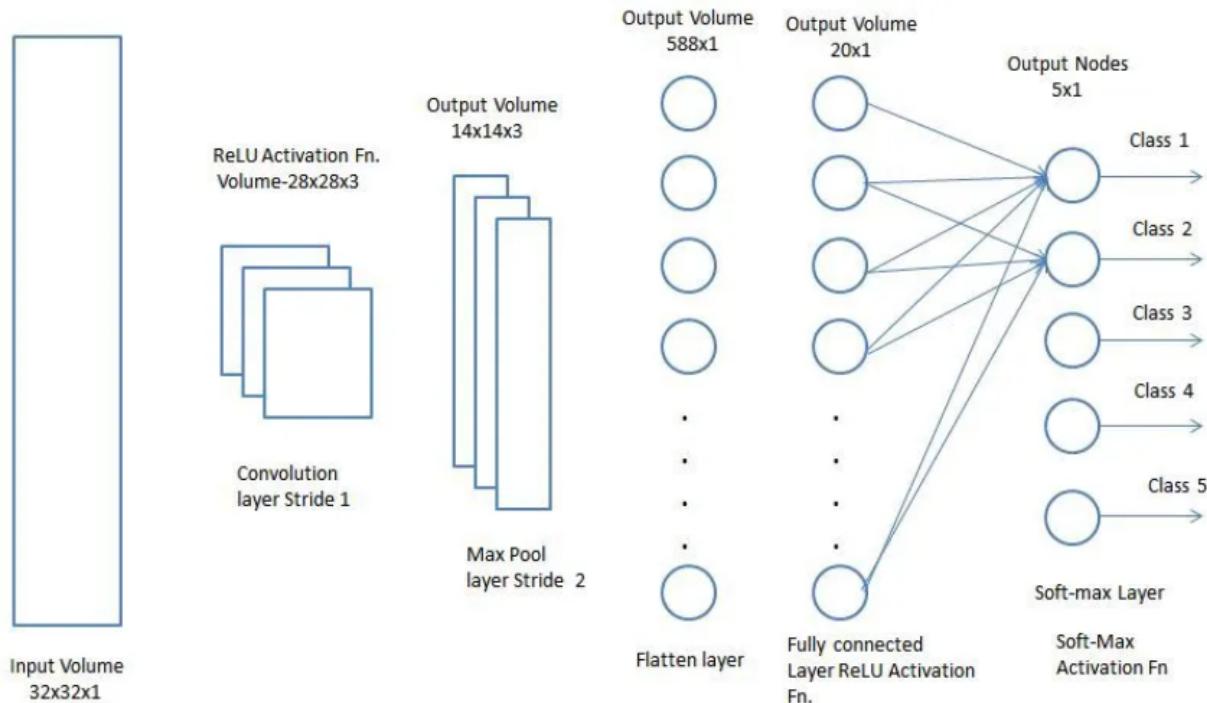


FIGURE 2.1 – Architecture de CNN [3]

1. **Couche convulsive** Consiste à appliquer un ensemble de filtres (ou noyaux) sur une image d'entrée. Cette opération de convolution permet d'extraire des caractéristiques importantes de l'image, telles que les bords, les coins et les motifs répétitifs.

Afin d'introduire de la non-linéarité aux caractéristiques extraites, une fonction d'activation est appliquée. Il existe plusieurs fonctions d'activation couramment utilisées, notamment : [9]

- **ReLU** est la plus utilisée vue sa fiabilité et son accélération à converger. Elle est définie par :

$$Relu(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x > 0 \end{cases} \quad (2.1)$$

L'un des inconvénients de ReLU est la production de sorties "explosives" lorsque les entrées sont très grandes, ce qui peut causer des problèmes lors de l'entraînement

du modèle. Cependant, ce problème peut être évité en choisissant un taux d'apprentissage approprié.

- **Sigmoid** transforme une valeur continue en une valeur comprise entre 0 et 1.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

L'inconvénient de cette fonction est que les résultats tendent vers epsilon, ce qui empêche le réseau de s'entraîner efficacement. Si les entrées sont toujours positives, les résultats seront tous positifs ou négatifs.

- **Tanh** est une fonction probabiliste, ce qui signifie que ses résultats sont compris dans l'intervalle  $[-1, 1]$ .

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.3)$$

Cette fonction a une tendance à centrer ses résultats autour de 0.

- **SoftMax** calcule les probabilités relatives de chaque classe. Elle est souvent utilisée dans la dernière couche pour un problème de classification multi-classe et elle est définie par :

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (2.4)$$

2. **Couche de Pooling** Cette couche a pour but de réduire les dimensions de la matrice des caractéristiques, en choisissant le max ou la moyenne.
3. **Couche complètement connectée (Fully Connected Layer)** L'ajout d'une couche complètement connectée est souvent considéré comme une méthode efficace et peu coûteuse pour permettre au modèle d'apprendre des combinaisons non-linéaires de caractéristiques de haut niveau.

Le CNN est utilisé comme base de plusieurs architectures complexes, telles que : VGG (voir 2.4.1.1), ResNet (voir 2.4.1.2) ...

### 2.2.1.2 Optical Character Recognition (OCR)

Optical Character Recognition, est un processus qui consiste à extraire le texte contenu dans des images. Cette technologie a été développée pour répondre aux besoins de numérisation comme celle des grandes entreprises et faciliter ainsi leur gestion de données.

Plusieurs OCR combinent les techniques traditionnelles de l'OCR, et les méthodes modernes basées sur les réseaux neuronaux pour la reconnaissance des caractères.

**Les techniques d'un OCR comprennent :**

**Le pré-traitement** : améliorer la qualité de la reconnaissance des caractères à l'aide des techniques telles que la suppression du bruit, le lissage et la binarisation des images. [10]

**La segmentation de l'image :** séparer l'image en zones de texte, lignes de texte, mots et caractères. [10]

**La reconnaissance des caractères :** identifier les caractères individuels dans chaque zone de texte à l'aide d'un modèle de deep learning [36] tel que le CNN, le RNN [37] ou le CRNN [38]...

**La correction des erreurs :** corriger les erreurs de reconnaissance de caractères à l'aide de dictionnaires de mots, de modèles linguistiques et de techniques de recherche.

## 2.2.2 Technique d'analyse du texte

Nous allons voir en détail les méthodes fondamentales largement utilisées dans le domaine du traitement du langage naturel.

### 2.2.2.1 Analyse syntaxique

Elle se concentre sur la disposition des mots afin d'assurer la précision grammaticale. Cette technique algorithmique analyse et organise les mots de manière à former des phrases cohérentes, sans aucune erreur de composition. Parmi les techniques de l'analyse syntaxique, nous retrouvons :

**Stemming et lemmatisation :** Le stemming est un processus informel de conversion des mots en leurs formes de base en utilisant des règles heuristiques. La lemmatisation, quant à elle, est une méthode plus formelle pour trouver les racines d'un mot en analysant sa morphologie à l'aide d'un dictionnaire.[11]

**Segmentation des mots :** Divise un texte en unités linguistiquement significatives. C'est évident dans des langues comme l'anglais où la fin d'une phrase est marquée par un point, mais cela reste complexe.[11]

**La suppression des mots vides :** Vise à éliminer les mots les plus couramment utilisés qui n'apportent pas beaucoup d'informations au texte. Par exemple, "le", "un", "une", etc.[11]

**La tokenisation :** Divise le texte en mots individuels et fragments de mots. Le résultat consiste généralement en un index de mots et en un texte tokenisé dans lequel les mots peuvent être représentés sous forme de jetons numériques pour une utilisation dans différentes méthodes d'apprentissage profond.[11]

### 2.2.2.2 Représentation du texte

Dans la majorité des tâches NLP, la conversion des mots en nombre est indispensable pour qu'ils soient interprétable par la machine. Il existe plusieurs méthodes de représentation de texte, parmi ces méthodes, nous retrouvons :

**One-hot encoding :** L'encodage one-hot est une technique de représentation du texte où chaque mot est transformée en un vecteur binaire. Dans cette représentation, un seul élément du vecteur est défini à 1 (chaud) pour indiquer le mot, tandis que tous les autres éléments sont définis à 0 (froid).

**Word2Vec :** Word2Vec est une méthode développée par Google. Elle utilise un réseau neuronal peu profond pour apprendre les embeddings<sup>1</sup> de mots. Les vecteurs sont appris en comprenant le contexte dans lequel le mot apparaît. Plus précisément, il examine les mots qui co-occurrent.

Un réseau de neurones est ensuite entraîné, tel que “continuous bag of words” (CBOW) [39] qui prédit le centre d'un mot en donnant le contexte de ses mots. La fonction objective d'entraînement vise à maximiser la probabilité de prédire le mot cible en fonction de ses mots contextuels.[39]

Les avantages de cette méthode sont sa capacité à capter la relation sémantique entre les mots, ainsi que la possibilité de paramétriser la taille du vecteur, contrairement à la méthode de TF-IDF[40], ce qui nous évitera de manipuler une matrice d'une grande dimension.

Il existe d'autres méthodes telles que le sac de mots (Bag of words), le CountVectorizer, GloVe [41] et les transformers.

### 2.2.2.3 Réseaux de Neurones Récurrents

Les réseaux de neurones récurrents (**Recurrent Neural Networks**) sont utilisés pour des tâches impliquant des données séquentielles, telles que la modélisation du langage, l'analyse des séries chronologiques. L'architecture de base d'un RNN implique une série d'unités répétitives, ou cellules, qui sont connectées les unes aux autres dans une structure en chaîne.[4]

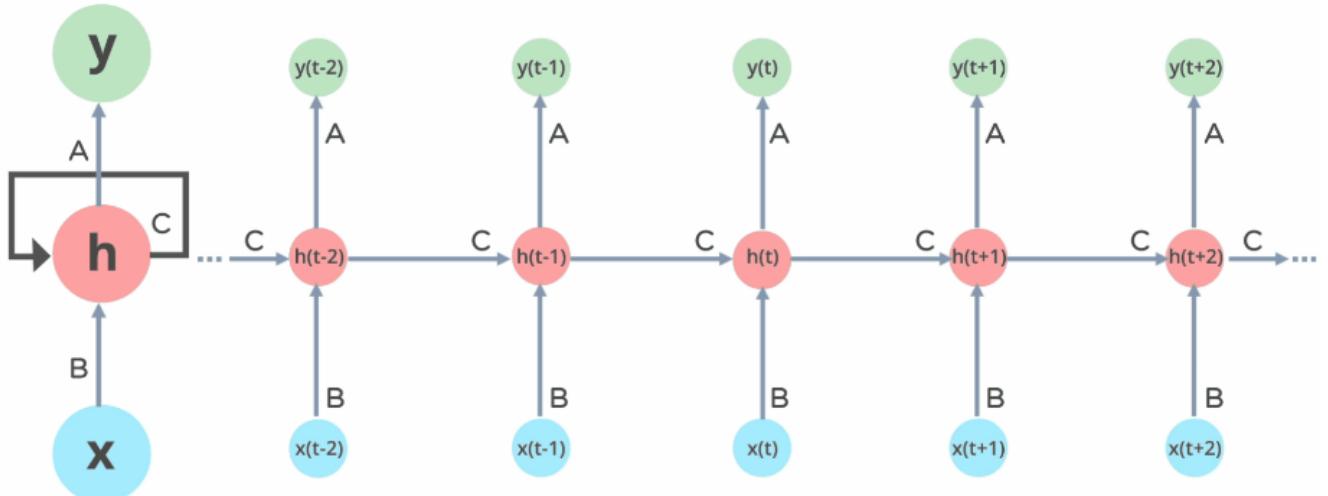


FIGURE 2.2 – Architecture de RNN [4]

$$h(t) = \sigma_c(h(t-1), x(t)) \quad (2.5)$$

### Notation

1. Embedding : ou le plongement sémantique, est une technique qui permet la représentation des mots ou des phrases sous forme de vecteurs numériques dans un espace mathématique, capturant ainsi leur signification et leurs relations sémantiques.

- $x(t)$  : vecteur d'entrée en  $t$
- $h(t)$  : nouvel état
- $h(t - 1)$  : état précédent
- $\sigma$  : fonction d'activation avec le paramètre  $c$

Chaque cellule prend une entrée et produit une sortie, ainsi qu'un état caché qui est transmis en entrée à la cellule suivante de la chaîne. L'état caché permet au réseau de conserver des informations sur la séquence qu'il a vue jusqu'à présent, ce qui est important pour des tâches telles que la modélisation du langage ou la reconnaissance de la parole.

### 2.2.3 Les techniques de pointes de NLP et de vision par ordinateur

Plusieurs méthodes ont été implémentées pour la manipulation du texte dont les techniques de machine learning. Cependant, les transformers ont récemment obtenu des résultats de pointe dans diverses tâches NLP, y compris des tâches de vision par ordinateur.

#### 2.2.3.1 Les Encodeurs et les Décodeurs

En apprentissage profond, les encodeurs et les décodeurs sont des composants clés de certaines architectures comme les modèles de type "séquence à séquence", "auto-encoding" ou des modèles "auto-régressive", qui sont souvent utilisés dans le traitement du langage naturel (NLP).

L'architecture d'un encodeur et d'un décodeur peut varier en fonction de la tâche spécifique et de la conception du modèle. Cependant, les deux peuvent être construits en utilisant un réseau de neurones simple, ou des architectures comme un RNN ou un CNN. [42]

- **Encodeur**

Le rôle principal d'encodeurs est de convertir l'entrée en une représentation de dimension inférieure appelée représentation d'encodage ou espace latent. Cette dernière est aussi connue comme le vecteur des caractéristiques de l'entrée, qui signifie que le modèle est optimisé pour acquérir une compréhension à partir de l'entrée. [42]

Les modèles qui dépendent seulement d'un encodeur sont appelés les modèles "auto-encoding".

- **Décodeur**

Le rôle du décodeur est de produire une séquence de sortie en fonction de ce qui est en entrée, et cela est généralement réalisé en utilisant une architecture de réseau de neurones récurrents tels que LSTM<sup>2</sup> ou GRU<sup>3</sup>. Le décodeur est entraîné à générer les meilleurs résultats possibles en termes de séquences de mots. [42]

Les modèles qui dépendent seulement d'un décodeur sont appelés les modèles "auto-régressive".

- **Encodeur-Décodeur**

Ce sont des modèles qui dépendent d'un encodeur et d'un décodeur, appelé souvent les modèles "sequence to sequence".

---

2. LSTM : Long Short-Term Memory networks, une version améliorée de RNN [43]

3. GRU : Gated Recurrent Unit, une variante de RNN [43]

### 2.2.3.2 Mécanisme d'attention

Le mécanisme d'attention a pour objectif principal de déterminer les informations les plus pertinentes dans la matrice de données afin que le modèle puisse les traiter efficacement. [33]

Il existe plusieurs types de mécanisme d'attention, nous mentionnant les plus importants :

**Attention multi-têtes (multi-head attention)** : divise les vecteurs de requête, de clé et de valeur en plusieurs têtes et applique l'attention à produit scalaire à chaque tête indépendamment.[33]

Pour le calcul du score, nous calculons d'abord les valeurs de requête, clé et valeur :

- **Query(Q)** : Représentation des informations recherchées ou interrogées à partir de l'entrée (chaque tête peut avoir son propre vecteur Q).
- **Key(K)** : Représentation des informations utilisées pour comparer ou correspondre à la requête (chaque tête peut avoir son propre vecteur K).
- **Value(V)** : La représentation des informations utilisées pour produire la sortie ou la somme pondérée des valeurs.

Ensuite, on calcule les poids d'attention (softmax) pour chaque paire Requête-Clé :

$$\text{softmax} \left( \frac{QK}{\sqrt{d_k}} \right) \quad (2.6)$$

Où  $d_k$  est la dimension des vecteurs de clé et de requête.

Nous calculons ainsi le score de l'attention comme suit :

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2.7)$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2.8)$$

$W$  est une matrice de poids apprise lors de l'entraînement.

**Auto-attention (self-attention)** : Ce type d'attention est utilisé pour analyser les relations entre différents éléments d'une même séquence ou phrase, où chaque élément est observé en relation avec les autres éléments de la séquence[33]. Nous pouvons calculer sa valeur comme suit :

$$\text{softmax} \left( \frac{QK}{\sqrt{d_k}} \right) \cdot V \quad (2.9)$$

- **Query(Q)** : Représentation des informations recherchées ou interrogées à partir de l'entrée.
- **Key(K)** : Représentation des informations utilisées pour comparer ou correspondre à la requête.
- **Value(V)** : La représentation des informations utilisées pour produire la sortie ou la somme pondérée des valeurs.

Le choix du type du mécanisme d'attention dépend de la tâche à réaliser et des données utilisées.

### 2.2.3.3 Architecture des transformers

L'architecture des transformers repose sur le mécanisme d'attention et peut comporter deux parties, une partie encodeur et une autre pour le décodeur.

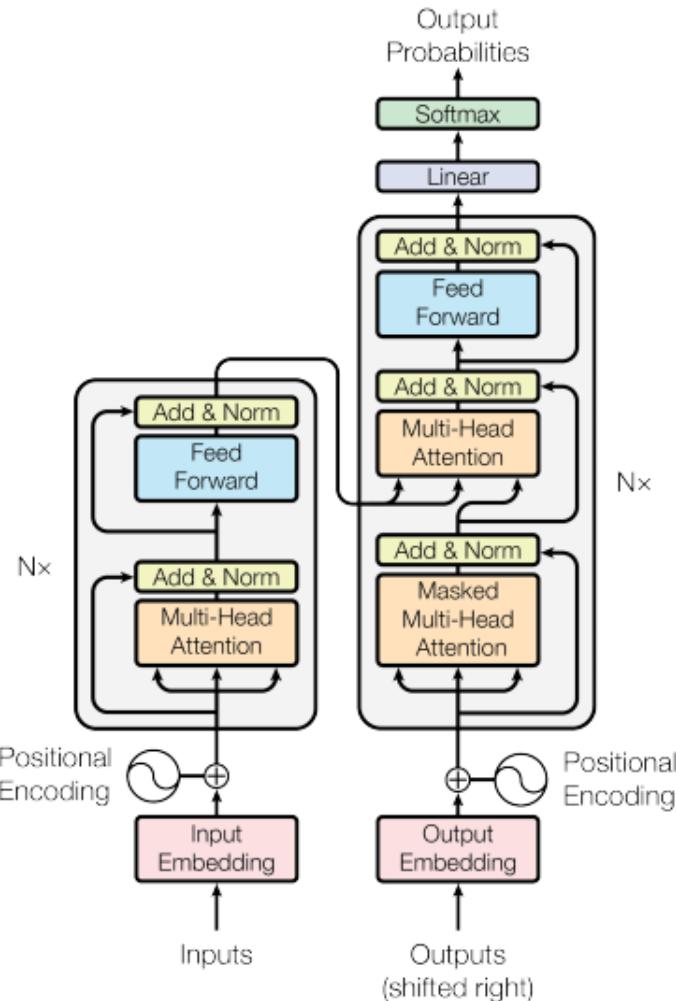


FIGURE 2.3 – Architecture de Transformer [33]

Chaque mot formant une séquence d'entrée est transformé en un vecteur de words embedding. Les données sont ensuite introduites dans le bloc encodeur composé des deux sous-couches. La première sous-couche est un mécanisme d'auto-attention à têtes multiples (Multi-Head attention), et la seconde est un simple réseau de feed-forward entièrement connecté. Sur la sortie du décodeur, le masquage est appliqué dans la première sous-couche, représentant le mécanisme d'attention multi-head, elle assure que la prédiction pour une position  $i$  ne peut dépendre que des outputs connus aux positions inférieures à  $i$  [33].

### 2.2.3.4 Transformers de vision (ViTs)

Le Transformer de vision est une avancée récente de la vision par ordinateur qui consiste à adapter l'architecture d'un transformer, initialement conçue pour le traitement du langage

naturel (NLP), à la tâche d'analyse d'images. L'architecture d'un transformeur est particulièrement bien adaptée aux problèmes basés sur des séquences, et les images peuvent être vues comme une séquence de pixels.

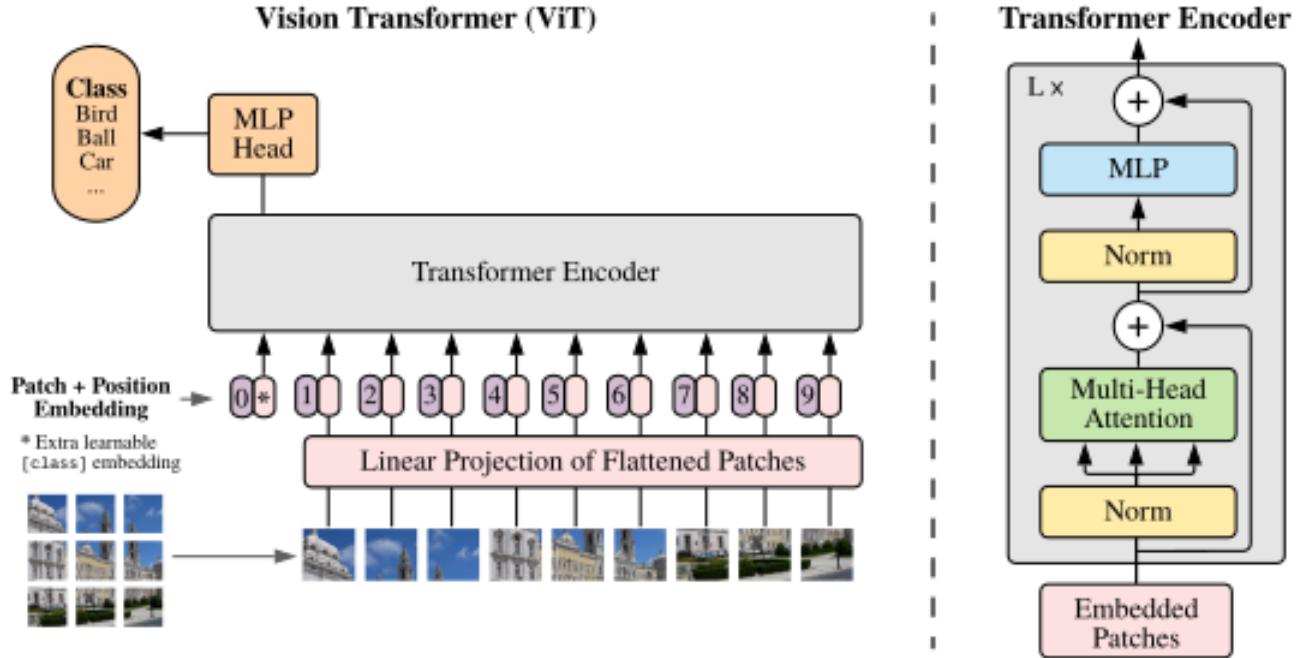


FIGURE 2.4 – Aperçu du modèle ViT [34]

ViT divise l'image en une séquence de patchs et les traite à l'aide d'un encodeur de transformeur, ViT est capable de comprendre les caractéristiques locales et globales de l'image. Les patchs sont représentés par des intégrations positionnelles apprenables, qui permettent au réseau de raisonner sur les relations spatiales entre les patchs.[34]

### 2.2.3.5 Mécanisme d'attention spatiale

Il s'agit d'un type de mécanisme d'attention, utilisé généralement pour capturer les relations et les dépendances entre les différentes locations spatiales d'une image. [44]

Elle prend en entrée une carte de caractéristiques qui correspond au résultat d'une couche du réseau d'apprentissage.

Elle combine ensuite la carte des caractéristiques d'entrée avec les caractéristiques apprises en utilisant une opération de convolution. Cela permet de capturer les relations entre eux et de former enfin une carte d'attention.

Une fonction d'activation est appliquée aux caractéristiques combinées, à savoir sigmoïde, softmax... Si nécessaire, une normalisation de la carte d'attention résultante est utilisée dans le but de garantir que les poids d'attention sont des probabilités valides qui se somment à 1.

Les éléments de la carte des caractéristiques d'entrée sont multipliés un par un avec la carte d'attention. Cette opération donne différents poids aux différentes régions spatiales, et ce, en

mettant en valeur les régions pertinentes et en supprimant les régions non-pertinentes. Les caractéristiques résultantes sont une combinaison pondérée des caractéristiques d'entrée.

Si nécessaire, une agrégation des caractéristiques pondérées est appliquée, pour obtenir une représentation condensée.

La sortie finale du mécanisme d'attention spatiale comprend généralement la carte d'attention et la représentation pondérée, qui peuvent être utilisées dans les couches suivantes ou comme partie de la sortie du modèle.

## 2.2.4 Algorithmes d'optimisation

Ce sont des algorithmes qui permettent d'ajuster les poids d'un réseau de neurones lors de la phase d'apprentissage, en ajustant son taux d'apprentissage de manière à améliorer sa performance et réduire sa perte.

Avant de présenter les différents algorithmes d'optimisation, nous allons tout d'abord mentionner les fonctions de perte utilisées dans un problème de classification :

- **Cross entropy** : Est une fonction de perte définie par :

$$\text{CrossEntropyLoss} = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (2.10)$$

Où  $y_i$  représente la prédiction de l'élément  $i$  et  $\hat{y}_i$  représente la classe réelle de l'élément  $i$

- **Squared\_hinge** : Est une fonction qui calcule la perte en utilisant la formule :

$$loss = \max(0, 1 - y_{true} * y_{pred})^2 \quad (2.11)$$

Tel que :  $y_{true}$  représente les vraies étiquettes des exemples d'entraînement et  $y_{pred}$  représente les prédictions faites par le modèle. Son objectif est de maximiser la marge entre les classes et de minimiser les erreurs de classification. Cette fonction est souvent utilisée avec le modèle SVM.

### 2.2.4.1 Stochastic Gradient Descent

Cet algorithme choisit d'une manière aléatoire un sous-ensemble de données afin d'optimiser leur perte en ajustant les poids du réseau [12]. Ci-dessous l'équation mathématique de l'algorithme :

$$w_j = w_j - lr \frac{\partial L}{\partial w_j} \quad (2.12)$$

#### Notation

- $w_j$  : Le poids du neurone  $j$ .
- $L$  : Loss function, la fonction de perte que l'optimiseur doit minimiser.
- $lr$  : Learning Rate ou le taux d'apprentissage, contrôle la taille chaque pas que nous effectuons à chaque itération.

Dans l'équation précédente, l'estimation de la dérive de la fonction de perte est calculée.

Bien que l'algorithme SGD puisse aider à éviter certains minimums locaux, il ne garantit pas de les éviter tous. Il existe donc une version optimisée de ce modèle appelé 'SGD with momentum'.

Ainsi, une version améliorée de cet algorithme est utilisée dans notre cas qui est appelé ‘SGD with momentum’. Un momentum (moment) est le mouvement moyen du gradient (la dérivée). Ci-dessous la nouvelle équation de SGD : [12]

$$V_t = \beta V_{t-1} + (1 + \beta) \Delta_w L(W, X, y) \quad (2.13)$$

$$W = W - \alpha V_t \quad (2.14)$$

### Notation

- $\beta$  : Momentum ( $\beta \in [0, 1]$ ).
- $L$  : Loss function, la fonction de perte que l'optimiseur doit minimiser.
- $W$  : Weights, les poids du réseau.
- $\alpha$  : Taux d'apprentissage.

Une version plus populaire de l'équation :

$$V_t = \beta V_{t-1} + \alpha \Delta_w L(W, X, y) \quad (2.15)$$

$$W = W - V_t \quad (2.16)$$

Dans ce cas nous aurons :

$$\alpha = \alpha \times (1 - \beta) \quad (2.17)$$

#### 2.2.4.2 Adaptive Moment estimation (Adam)

Adam (Adaptive Moment estimation) [45] est un algorithme simple à implémenter et assez rapide en termes de temps d'exécution. Il consomme moins de mémoire et nécessite moins d'ajustements.[45]

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[ \frac{\delta L}{\delta \omega} \right] \quad (2.18)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[ \frac{\delta L}{\delta \omega} \right]^2 \quad (2.19)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2.20)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (2.21)$$

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}} + \epsilon \quad (2.22)$$

### Notation

- $\beta_1$  et  $\beta_2$  : Représentent le taux de décroissance de la moyenne des gradients.
- $L$  : Loss ou la perte.
- $\omega$  : weights ou les poids.
- $m_t, v_t$  : La moyenne de mouvement.
- $\eta$  : Taux d'apprentissage.

## 2.3 Compréhension automatique de document

La compréhension automatique de documents est le processus d'extraction automatique d'informations significatives à partir de documents tels que du texte, des images et d'autres supports, à l'aide des outils de machine learning et de traitement du langage naturel. D'autres techniques en vision par ordinateur ont aussi été implémentées pour le même objectif et des tâches en aval telles que : la classification de documents, l'extraction d'information, la reconnaissance d'entités, l'analyse des sentiments, la modélisation des sujets, etc...

### 2.3.1 Modèles pré-entraînés

Nous allons voir les différents modèles de l'état de l'art, utilisés pour les différentes tâches de compréhension automatique de documents.

#### LayoutLM, 2019

LayoutLM[46] est un modèle multimodal<sup>4</sup> pré-entraîné développé par Microsoft Research Asia pour l'analyse des images de documents. Il combine la puissance de la vision par ordinateur et du traitement du langage naturel (NLP) pour s'attaquer à la tâche complexe de comprendre la mise en page et le contenu des documents. LayoutLM utilise l'architecture BERT (voir [2.4.2.4 BERT](#)) comme backbone, qui est une technique de traitement du langage naturel basée sur l'architecture des transformers. LayoutLM ajoute deux nouvelles intégrations d'entrée :

- **2D position embeddings** : vise à modéliser la position spatiale relative des éléments dans un document. Le texte et ses coordonnées 2D sont extraits en utilisant un OCR.
- **Image embeddings** : est utilisée pour capturer les caractéristiques visuelles telles que la couleur et la police du texte, ainsi que pour aligner ces caractéristiques avec le texte. Faster R-CNN est utilisé pour extraire les régions de texte liées au document. Faster R-CNN est un modèle d'image utilisé pour la détection d'objets. Dans ce cas, il a été utilisé pour détecter les différentes parties du texte. Enfin, les images segmentées sont passées à travers une couche entièrement connectée pour aider à générer des incorporations (embeddings) pour les images également.

---

4. Multimodal : le concept de multimodalité implique plusieurs parties, telles que le texte et les images comme dans le cas des modèles cités précédemment, ou encore la parole et la vidéo. Son objectif est d'extraire des informations pertinentes de chaque partie des données présentes, en les fusionnant pour obtenir une meilleure compréhension des données.

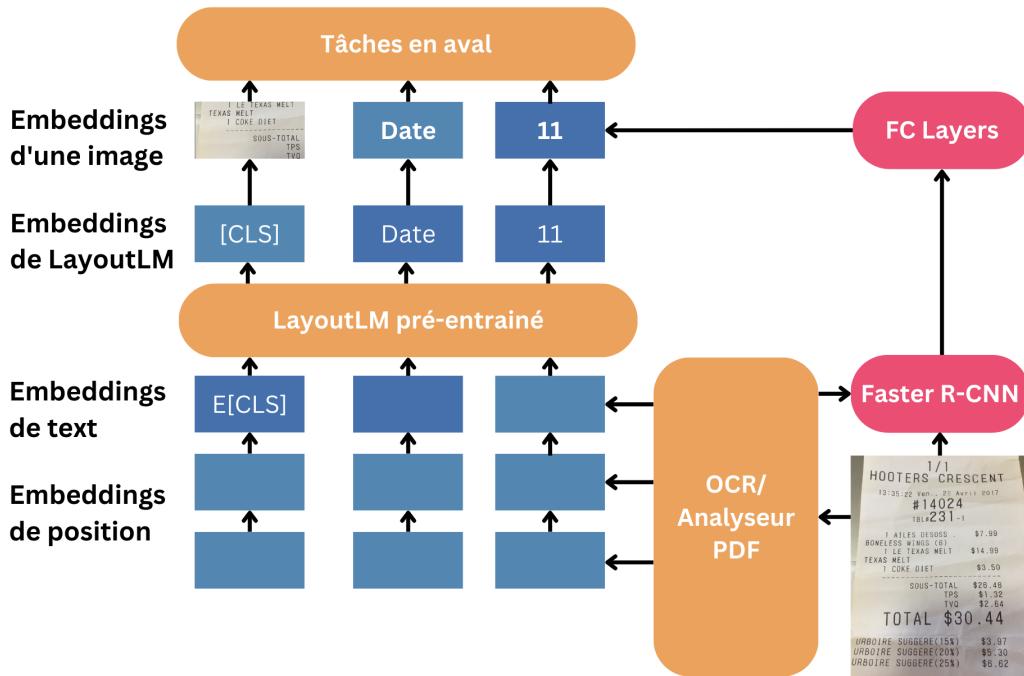


FIGURE 2.5 – Architecture de layoutLM

Le modèle est pré-entraîné à l'aide de **MVLM**, **Masked Visual-Language Model**, un modèle qui combine les capacités du traitement visuel et linguistique en utilisant le concept du masquage afin d'apprendre la représentation du langage avec les indices d'intégrations de position 2D et d'intégrations du texte[46].

### LayoutLMv2, 2021

LayoutLMv2 [47] est une version améliorée de LayoutLM. La principale différence entre LayoutLM et LayoutLMv2 réside dans le fait que ce dernier intègre des incorporations visuelles lors du pré-entraînement (tandis que LayoutLM n'ajoute des incorporations visuelles que lors de l'affinage) et aussi LayoutLMv2 ajoute à la fois un biais d'attention relative 1D et un biais d'attention spatiale 2D aux scores d'attention dans les couches d'auto-attention. En outre, il comprend plusieurs améliorations techniques par rapport au modèle d'origine. Ces améliorations rendent LayoutLMv2 plus puissant et efficace pour les tâches de compréhension de documents.

### DocFormer, 2021

DocFormer[48] est un modèle end-to-end de Amazon. DocFormer représente une architecture de transformer basée uniquement sur l'encodeur. Il dispose également d'un backbone CNN pour l'extraction des caractéristiques visuelles. Tous les composants sont entraînés de bout en bout. Le modèle est pré-entraîné à l'aide **MM-MLM**, **Multi-Modal Masked Language Modeling** pour apprendre la représentation visuelle, linguistique et spatiale de l'image.

## LayoutLMv3, 2022

En termes de traitement des données, LayoutLMv3 [49] est identique à son prédécesseur, LayoutLMv2, à l'exception des points suivants :

- Les images doivent être redimensionnées et normalisées avec les canaux dans le format RVB<sup>5</sup> standard. LayoutLMv2, en revanche, normalise les images en interne et attend les canaux dans le format BVR.
- Le texte est tokenisé en utilisant le codage par paires de caractères (BPE)<sup>6</sup>, contrairement à WordPiece comme dans le cas de layoutLM et layoutLMv2.
- Les résultats obtenus dans les tâches en aval avec layoutLMv3 est supérieur à celle obtenue avec ses précédentes versions.

## ERNIE-Layout, 2022

ErnieLayout[50] est un modèle alimenté par BAIDU WenXin Document Intelligence Team, qui étant donné un document, le modèle réorganise la séquence de jetons avec les connaissances de mise en page et extrait les caractéristiques visuelles de l'encodeur visuel, le modèle est adapté aux documents en langue chinoise.

## Donut Free-OCR model, 2021

Les modèles les plus populaires que nous avons trouvés nécessitent souvent un OCR pour extraire le texte de ce dernier. Cependant, il existe d'autre type de modèle avec un principe différent comme le modèle de Donut.

Donut[51] est un modèle end to end, qui n'utilise pas d'OCR pour la compréhension des documents. Son architecture est une combinaison entre un encodeur et un décodeur.

**Encodeur :** Les blocs Swin Transformer [52], constitués d'un module d'auto-attention multi-têtes basées sur une fenêtre décalée qui permet d'éviter d'ignorer certaines parties de l'image et de capturer des motifs plus larges. Les blocs sont aussi constitués d'un MLP<sup>7</sup> à deux couches, qui sont appliqués aux patchs<sup>8</sup>. Ensuite, des couches de fusion de patchs sont appliquées aux jetons de patch à chaque étape.

Cet encodeur visuel convertit l'image du document d'entrée  $x \in \mathbb{R}^{H \times W \times C}$  en un ensemble  $\{z_i | z_i \in \mathbb{R}^d, 1 \leq i \leq n\}$ , où  $n$  est la taille de la carte caractéristique ou le nombre de patchs d'image et  $d$  est la dimension des vecteurs latents de l'encodeur.

La sortie du bloc Swin Transformer final  $\{z\}$  est introduite dans le décodeur textuel suivant.

**Décodeur :** Le transformer **BART**(voir **2.4.2.4 BART**) est utilisé comme décodeur, qui, à partir des  $\{z\}$ , génère une séquence de jetons  $(y_i)$  où  $y_i \in \mathbb{R}^v$  est un vecteur **one-hot** pour le  $i$ -ème jeton.

---

5. BVR : format de couleurs des images indiquant Bleu, Vert et Rouge.

6. BPE : consiste à découper le texte en sous-unités plus petites, appelées "paires de caractères", afin de représenter le vocabulaire de manière plus efficace.

7. MLP : "Multi-Layer Perceptron", c'est un type de réseau de neurones artificiels où plusieurs couches de neurones sont empilées les unes sur les autres pour effectuer des tâches d'apprentissage.

8. Patch : un patch est une petite région carrée ou rectangulaire d'une image.

### Notation

- $v$  est la taille du vocabulaire de jetons.
- $m$  est un hyperparamètre.

Une stratégie d'entraînement, appelée Teacher-forcing a été implémentée, celle-ci utilise la vérité terrain comme entrée, en outre, le modèle est entraîné à prédire le jeton suivant dans une séquence en se basant sur le jeton précédent correct plutôt que sur son propre jeton généré précédemment. Le modèle est entraîné sur IIT-CDIP<sup>9</sup>, il est entraîné pour lire tout le texte de l'image dans l'ordre de lecture du haut à gauche au bas à droite. L'objectif est de minimiser la perte d'entropie croisée.[51]

### 2.3.2 Fine-tuning sur les tâches en aval

Le fine-tuning est un processus de transfert des connaissances acquises de l'ensemble de données source vers l'ensemble de données cible. Le modèle cible copie la conception et les paramètres à partir du modèle source, à l'exception de la couche de sortie, et affine ces paramètres en fonction du jeu de données ciblé. En revanche, la couche de sortie du modèle cible doit être formée à partir de zéro.[53]

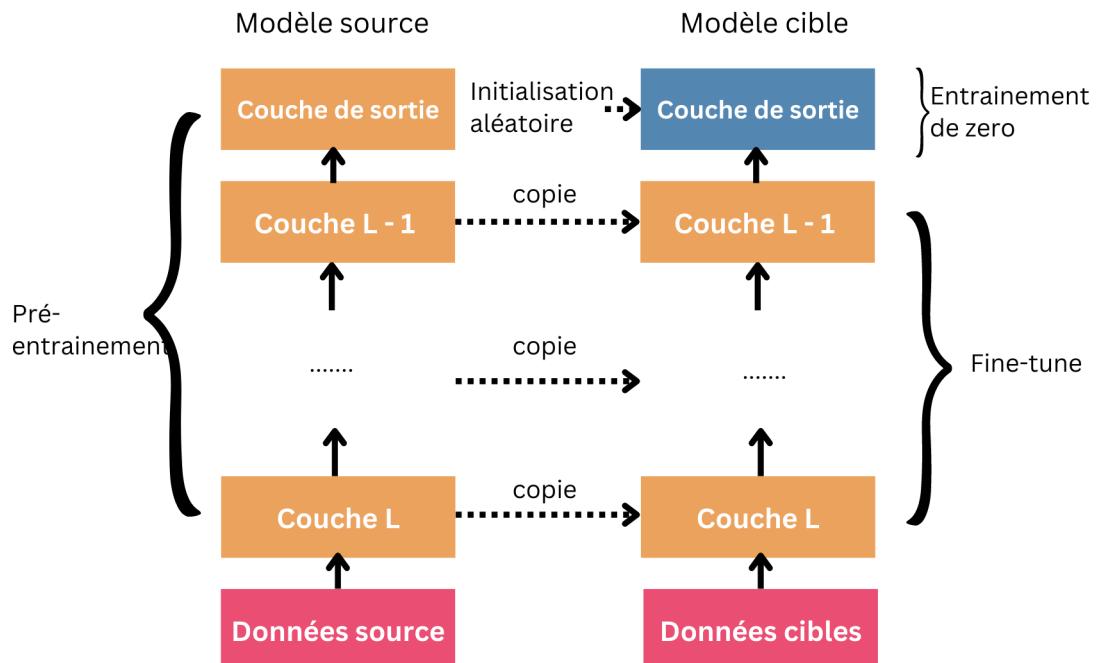


FIGURE 2.6 – Schéma représentant le fine tuning

Le fine-tuning peut être utilisé pour les tâches en aval dans le domaine de la compréhension automatique de documents. Dans le cadre de notre projet de fin d'études, nous allons nous intéresser à la classification automatique de documents.

9. IIT-CDIP : Illinois Institute of Technology-Complex Document Information Processing, est une grande dataset de document qui consiste en un ensemble de 11 millions d'images de documents en anglais numérisés.

## 2.4 Classification automatique de document

### 2.4.1 Approche visuelle

Dans cette approche, les techniques de vision par ordinateur identifient les caractéristiques des différents types de document pour enfin les classer en conséquence. L'idée derrière cette approche est d'avoir un modèle capable d'analyser la structure en pixels d'une image et d'essayer de trouver comment les blocs ou les cases sont repartis sans lire le texte.

Les modèles basés CNN, comme VGG[54], ResNet ou les transformers de vision, peuvent être utilisés dans cette approche.

#### 2.4.1.1 VGG (Visual Geometry Group)

VGG[55] est une architecture composée de réseaux de neurones convolutifs profond inspiré de AlexNet [55]. Elle est composée de plusieurs couches. Nous résumons son architecture comme suit :

- **Entrée** : initialisée en une image de  $224 \times 224$ .
- **Couche conventionnelle** : Les développeurs de VGG ont cherché à minimiser la taille des filtres. Ainsi, un filtre de  $3 \times 3$  suffit à capturer les informations sur les pixels adjacents, à savoir leur position en haut, en bas, à gauche et à droite. Le résultat est ensuite passé par une fonction de ReLu.
- **Couches cachées** : Toutes les couches cachées (hidden layer) de VGG utilisent ReLu.
- **Couches entièrement connectées** : Le réseau VGG contient 3 couches entièrement connectées. Les deux premières contiennent 4096 canaux, tandis que la troisième contient un canal pour chaque classe à classifier.

#### 2.4.1.2 ResNet (Residual Network)

ResNet[35] est une architecture standard de CNN, proposée pour résoudre le problème de la disparition du gradient qui peut survenir lors de l'utilisation de nombreuses couches de convolutions.

L'architecture de ResNet est principalement composée de 2 différents blocs :

**Bloc convolutif** : composée d'une couche de convolution qui est normalisée à l'aide d'un batch de normalisation, suivie par une fonction d'activation (ReLu). Après ce bloc, une opération de max pooling est appliquée.

**Blocs Résiduels** : elles sont composées de plusieurs couches convolutives suivies d'une normalisation par lot et une fonction d'activation (ReLu). Ci-dessous la formule de la normalisation :

$$y = \frac{x - E(x)}{\sqrt{\text{Var}[x] + \epsilon}} \cdot \gamma + \beta \quad (2.23)$$

#### Notation

- $x$  : Est l'entrée.
- $\epsilon$  : Un paramètre ajouté pour la stabilité du calcul.

- $\gamma$  et  $\beta$  : Paramètres apprenables.
- $E[x]$  : La moyenne d'entrée  $x$  sur le mini-lot actuel.
- $Var[x]$  : La variance d'entrée  $x$  sur le mini-lot actuel.

La particularité du bloc résiduel repose dans l'ajout d'une connexion directe qui contourne ces couches. Cette connexion résiduelle permet au gradient de se propager plus facilement à travers le réseau. Ainsi, un réseau de neurones profond pourra éviter la disparition du gradient.

Nous présentant un bloc résiduel dans la Figure 2.7 ci-dessous :

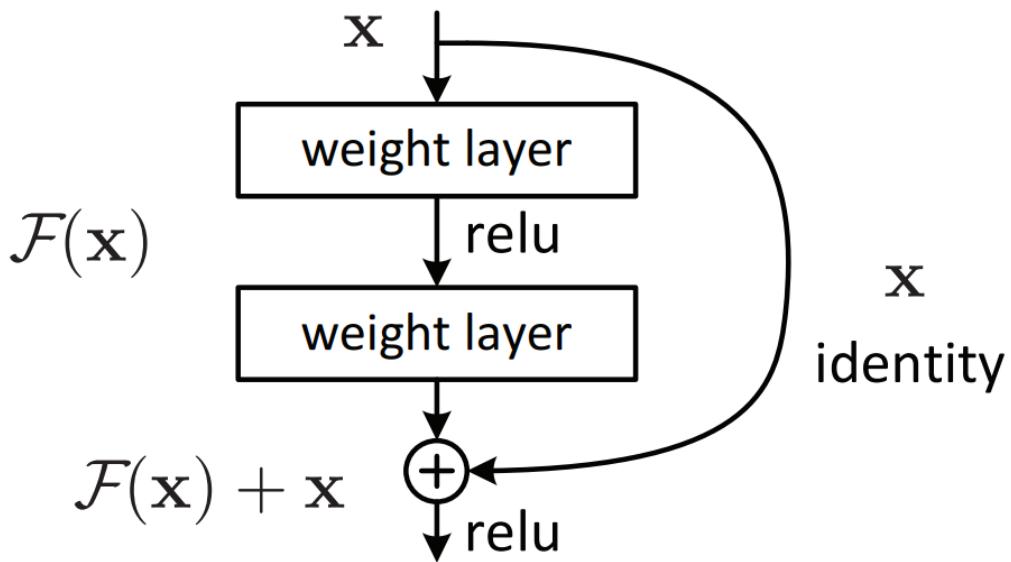


FIGURE 2.7 – Schéma représentant le bloc Résiduel [35]

Le calcul de résidu mène à la possibilité de sauté des couches :

$$Residu = Output - Input \quad (2.24)$$

$$R(x) = F(x) - x \quad (2.25)$$

$$F(x) = R(x) + x \quad (2.26)$$

La séquence des blocs résiduels est généralement suivie par une couche de pooling moyen afin de réduire la dimension spatiale des caractéristiques. Enfin, une couche entièrement connectée est utilisée pour la sortie finale du modèle.

Nous notons qu'il existe plusieurs variantes de ce modèle en fonction du nombre des couches qu'il contient. Ainsi, nous trouvons le ResNet-18, ResNet-34...

### 2.4.1.3 DocXClassifier, 2022

DocXClassifier[56], une approche basée sur les réseaux convolutionnels qui applique les stratégies modernes d'augmentation des données et de d'entraînement. DocXClassifier est capable de générer des cartes d'attention de type transformer, qui le rend intrinsèquement interprétable. Ce modèle a été développé en apportant diverses modifications au modèle ResNet standard [35], modifications inspirées à la fois par les ConvNets modernes et le récemment introduit Swin Transformers [52], une variante de ViTs.

## 2.4.2 Approche textuelle

Dans cette approche, un OCR est souvent utilisé pour l'extraction et la reconnaissance du texte. Les documents sont ensuite classés en fonction des informations dérivées, en utilisant les techniques de NLP ou des algorithmes basés sur les mots-clés et les règles de reconnaissance du texte.

### 2.4.2.1 Extraction des éléments pertinents

Il existe plusieurs modèles d'OCR pour l'extraction des éléments pertinents d'une image, tel que le texte ou la position 2D de ce dernier, parmi les OCRs open source, nous retrouvons :

- **Tesseract** : Est un outil open source développé par Google qui permet l'extraction du texte à partir des différentes images et de documents scannés dans plusieurs langues. Il est désormais maintenu par une communauté open source active et il est utilisé sans licence commerciale.
- **EasyOCR** : Un logiciel open source utilisé généralement pour l'extraction du texte à partir des images en temps réel ou des documents scannés.
- **Keras-OCR** : Développé par KERAS, utilisé pour des documents manuscrits ou scannés.

Nous allons voir les différentes méthodes utilisées pour la classification des documents après l'extraction des éléments pertinents du document.

### 2.4.2.2 Classification basée sur les règles de reconnaissance du texte

Une méthode plus simple pour la classification de documents consiste à utiliser un système de règles (système expert). Ces systèmes utilisent des scripts pour exécuter des tâches et appliquer un ensemble de règles préalablement définies par des experts.

La première étape consiste à collecter et classer manuellement les documents en fonction de leurs catégories. Ensuite, une analyse des documents est réalisée pour extraire les mots clés spécifiques à chaque catégorie. Enfin, le système classe les nouveaux documents en fonction du nombre de mots clés appartenant à chaque catégorie.[13]

### 2.4.2.3 Classification basée sur les techniques de machine learning

#### • Régression logistique

La régression logistique[14] est une technique de modélisation statistique utilisée pour

estimer les relations entre une variable dépendante (elle peut prendre 2 ou plusieurs valeurs) et un ou plusieurs prédicteurs (variables explicatives).

- **SVM**

La machine à vecteurs de support (SVM)[15] est une méthode d'apprentissage supervisée qui trouve une frontière de décision en maximisant la marge entre les classes dans un espace de haute dimension transformé par un noyau. SVM est particulièrement adapté à l'analyse de données avec de très grands nombres de champs prédicteurs.

- **Arbre de décision**

Les arbres de décision[16] sont une méthode d'apprentissage supervisée utilisée dans l'analyse prédictive et la modélisation statistique. Ils se présentent sous la forme d'un diagramme de flux en arborescence qui représente les décisions et les conséquences potentielles qui en découlent.

- **Random Forest**

RandomForest[17] est une méthode d'apprentissage automatique qui combine plusieurs arbres de décision en un modèle prédictif. Chaque arbre est entraîné sur un échantillon aléatoire des données d'entraînement, et la prédiction finale est obtenue par vote majoritaire des prédictions de chaque arbre.

#### 2.4.2.4 Classification basée sur des modèles pré-entraînés

Parmi les modèles pré-entraînés utilisés dans cette partie, nous retrouvons les modèles cités dans la section de la compréhension automatique de documents tels que LayoutLM, DocFormer et Ernie-Layout. Ces modèles sont fine tuned pour la tâche de classification des documents. Il existe cependant d'autres modèles pour la tâche de classification de test tel que :

**BART** : (BiDirectional and AutoRegressive Transformers) [57] est un autoencodeur de débruitage, introduit en 2019 par des chercheurs de Facebook. Il est utilisé pour le préapprentissage de modèles de séquence à séquence. Il a été entraîné en corrompant le texte à l'aide d'une fonction de bruit arbitraire, puis en apprenant à reconstruire le texte original.[57]

L'architecture neuronale de BART repose sur le modèle standard de traduction automatique basé sur un Transformer.

Il est principalement utilisé pour la génération de textes, la traduction automatique, le résumé de textes abstraits, les systèmes de « question answering », etc. mais il peut également être utilisé pour des tâches de classification.

**BERT** : Bidirectional Encoder Representations from Transformers [58] est un modèle de langage développé par Google AI Language. L'architecture du modèle de BERT est un Transformer encoder bidirectionnelle multicouche. Il utilise le « modèle de langage masqué » (MLM), qui est un type de modèle de langage utilisé pour le pré-entraînement de certaines tâches de NLP. Dans MLM, certains des jetons du texte d'entrée sont masqués de manière aléatoire, et le modèle est ensuite formé pour prédire la valeur d'origine des jetons masqués.

**RoBERTA** : Robustly Optimized BERT Pretraining Approach [59] est également un modèle de langage basé sur l'architecture de BERT, qui contrairement à la méthode de

tokenisation WordPiece utilisée dans BERT, il utilise une tokenisation dynamique pour réduire le nombre de tokens inutiles. De plus, il utilise des embeddings de caractères non encodés pour former les embeddings initiaux.[59]

Il existe d'autres méthodes end-to-end qui ne dépendent pas d'un OCR mais qui prennent en considération le sens du texte pour la classification du document, comme dans le cas du modèle Donut vu dans la section précédente, qui peut aussi être fine tuned pour la tâche de classification de document.

## **2.5 Conclusion**

Dans ce chapitre, nous avons fourni un aperçu global des méthodes existantes, y compris les approches visuelles et textuelles, pour la classification de documents. Nous avons exploré les avancées et les modèles les plus récents dans ce domaine afin de fournir un contexte complet.

Dans le prochain chapitre, nous allons présenter en détail la conception nos solutions.

# Chapitre 3

## Conception

### 3.1 Introduction

L'objectif de notre projet est de réaliser un système automatique capable de classifier plusieurs types de documents d'un client, afin de faciliter l'utilisation des différentes APIs OCR présentes sur la plateforme EdenAI.

La Figure 3.1 donne un aperçu sur l'objectif global de notre système. Ce dernier prend en entrée le document du client, il le traite, il le classe, ensuite, il l'envoie à l'OCR le plus adéquat à son type pour enfin envoyer les données extraites, à partir de l'OCR, au client.

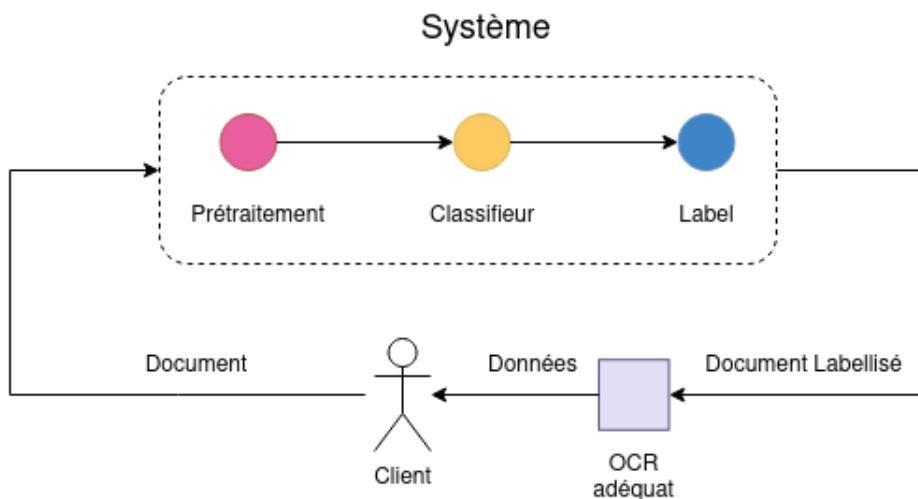


FIGURE 3.1 – Diagramme représentant l'objectif du système

Dans ce chapitre, nous allons expliquer en détail chaque étape entreprise pour la réalisation de ce système, ainsi que les différentes approches et méthodes testées.

### 3.2 La collecte des données

Comme nous l'avons mentionné dans la partie analyse du besoin de l'entreprise, les classes que nous devons prendre en considération dans notre projet dépendent des APIs OCR présentes sur la plateforme EdenAI.

Nous collectons des images pour chaque API : invoice (facture), receipt (reçu), resume (CV), ID, handwriting (écriture à la main) et enfin des documents contenant des tableaux. Les images collectées sont majoritairement en anglais et doivent être en corrélation avec les tendances d'aujourd'hui, notamment pour les CV. Elles doivent aussi être de bonne qualité pour faciliter l'extraction du texte.

Vu que l'entreprise n'avait pas de données au préalable, nous dépendons des plateformes telles que RoboFlow [18], Kaggle [19] et GitHub [20] pour la collecte de nos données. Nous les organisons de la manière suivante :

- Données structurées** : C'est un ensemble de données qui a un schéma bien défini, comme dans le cas de la classe table.
- Données semi-structurées** : Ce sont des types de données qui ont certaines caractéristiques structurées et d'autres caractéristiques non structurées de données, comme dans le cas de resume, id, invoice et receipt.
- Données non structurées** : Nous retrouvons dans cette catégorie les documents qui n'ont pas de structure bien définie telle que la classe handwriting.

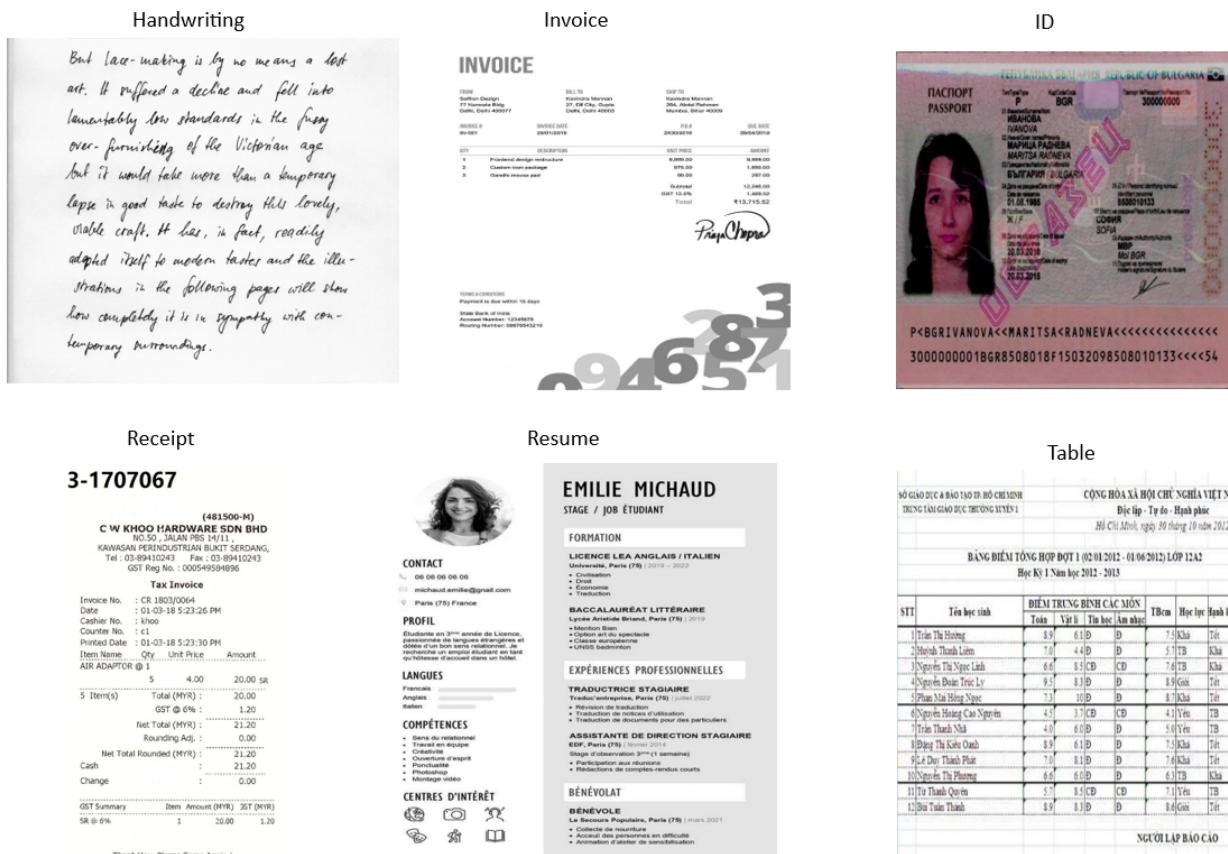


FIGURE 3.2 – Représentation du contenu de notre dataset

### 3.3 Exploration des données

Étant donné que les données collectées proviennent de différentes sources, il est nécessaire de faire une exploration et une analyse de ces derniers, et ce, afin de mieux comprendre leur contenu et de connaître le prétraitement à effectuer avant la phase de la modélisation.

#### 3.3.1 Analyse des données

L'analyse des données implique la vérification des différentes caractéristiques des images selon leurs classes.

##### Vérification du nombre de document par classe

Dans un problème de classification, l'équilibre entre les classes est une nécessité pour éviter certains problèmes, tels que le sur-apprentissage ou l'entraînement d'un modèle biaisé.

##### Vérification de la taille des images

Il est crucial de prendre en compte la taille des images lors de la modélisation, car celle-ci donne un aperçu de la qualité des images. De plus, certains modèles de vision nécessitent d'avoir des images de même taille en entrée avant l'entraînement.

##### Vérification de la distribution de l'intensité des pixels

Nous pouvons voir la variation du contenu des différentes images présentes dans notre dataset, en examinant la distribution de l'intensité des pixels.

#### 3.3.2 Nettoyage des données

Nous présentons dans cette partie les tâches manuelles et les autres automatiques que nous effectuons de manière générale pour traiter nos données avant d'entamer la conception de nos approches.

**Traitement manuel :** Nous vérifions le contenu des images, une par une, et nous avons supprimé celles qui n'étaient pas concernées par l'étude, comme des images d'email, des images contenant des graphes, des images altérées ou coupées...

**Traitement automatique :**

- Nous supprimons les images dupliquées pour éviter les redondances.
- Suppression des images de faible qualité ou résolution pour éviter les problèmes pendant l'entraînement.

#### 3.3.3 Augmentation des données

Nous créons un dataset augmenté à celui que nous collectons, et ce, pour varier nos données et augmenter la taille du dataset.

Nous appliquons plusieurs techniques de rotation, de translation, de retournement et d'injection de bruit avec des paramètres randomisés pour chaque image. [60]

Dans ce qui suit, nous allons présenter nos différentes approches pour atteindre notre objectif.

## 3.4 Solution basée sur les approches visuelles

Comme souligné dans le chapitre précédent, il existe peu de modèles qui se contentent seulement de la vision par ordinateur dans le domaine de la classification de documents.

La caractéristique intrinsèque d'un modèle de vision, qui ne dépend pas de la langue, mais plutôt de l'aspect structurel des documents, le rend avantageux en termes de généralisation.

Ainsi, notre proposition consiste à mettre en œuvre une approche reposant uniquement sur un modèle de vision afin d'évaluer si son entraînement est suffisant pour la tâche de classification de documents.

### 3.4.1 ResNet

Tout d'abord, nous proposons une architecture d'un réseau de neurones profond, telle que l'architecture classique de ResNet, expliquée en détails dans le chapitre précédent.

Nous optons pour un réseau de 50 couches en raison de sa complexité adaptée à la taille de nos données. En effet, une architecture plus simple risquerait de ne pas capturer les différentes caractéristiques de notre jeu de données, tandis qu'une architecture plus complexe pourrait entraîner un sur-apprentissage.

Voici l'architecture de notre réseau :

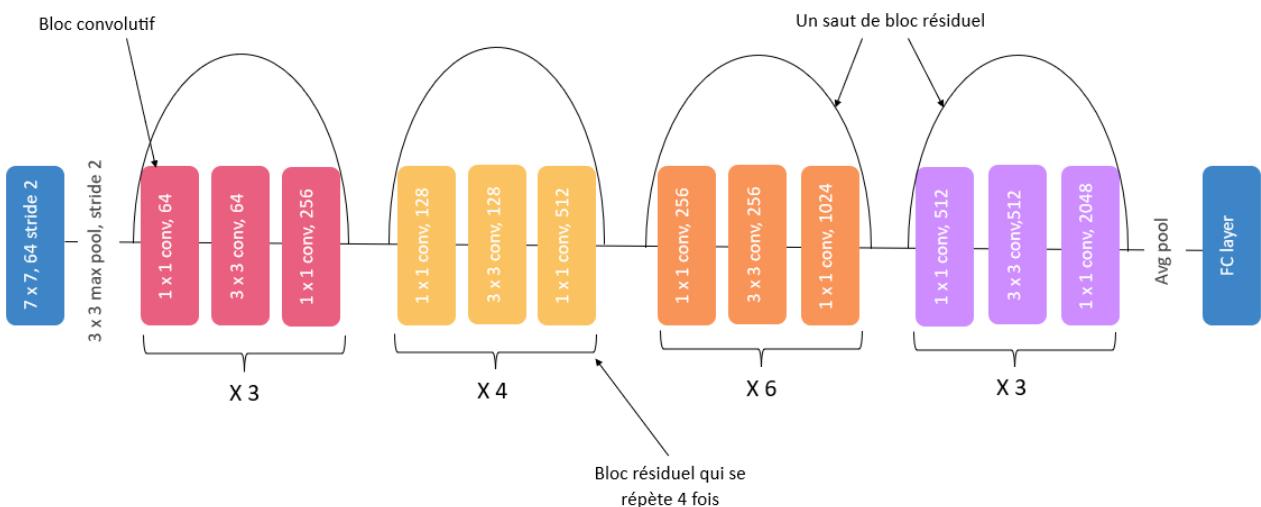


FIGURE 3.3 – Architecture de ResNet-50

Nous pouvons schématiser le bloc résiduel comme ci-dessous :

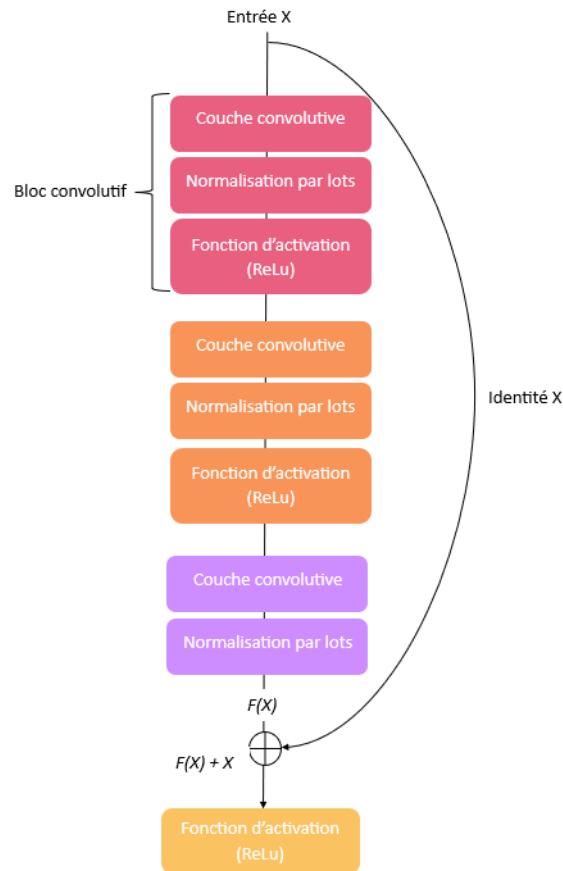


FIGURE 3.4 – Architecture d'un bloc résiduel pour ResNet-50

### 3.4.2 Attention ResNet-34

Nous proposons un nouveau réseau de neurones profonds en prenant l'architecture de ResNet-34 classique, qui consiste en une couche de convolution suivie de 16 blocs résiduels qui ont tous 2 blocs de convolutions. Ci-dessous une illustration du bloc résiduel pour un ResNet-34 :

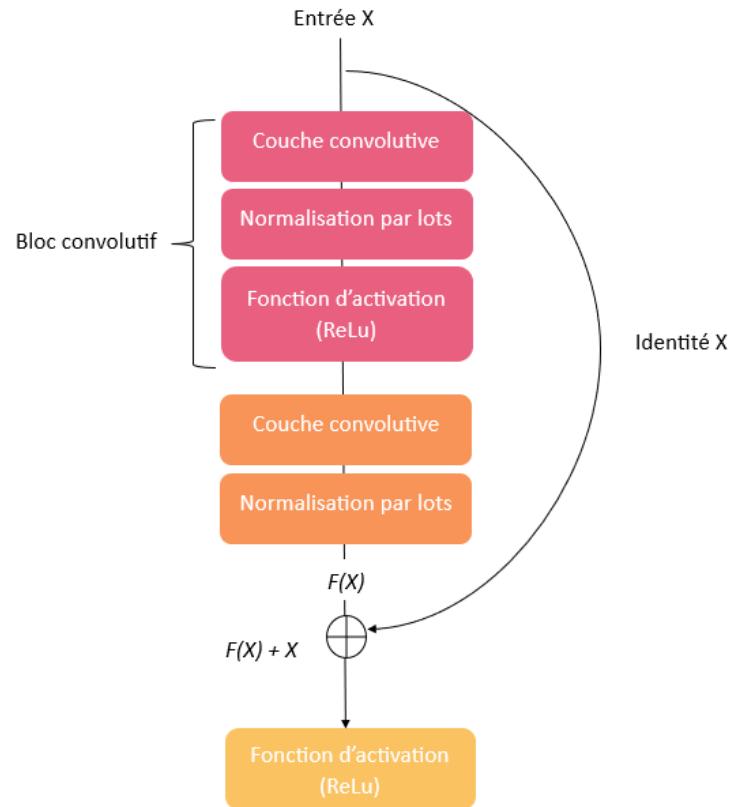


FIGURE 3.5 – Architecture d'un bloc résiduel pour ResNet-34

À la fin du réseau, au lieu d'appliquer une couche ‘fully connected’ (une couche totalement connectée), nous optons pour une couche de convolution comme montré dans la Figure 3.6 ci-dessous :

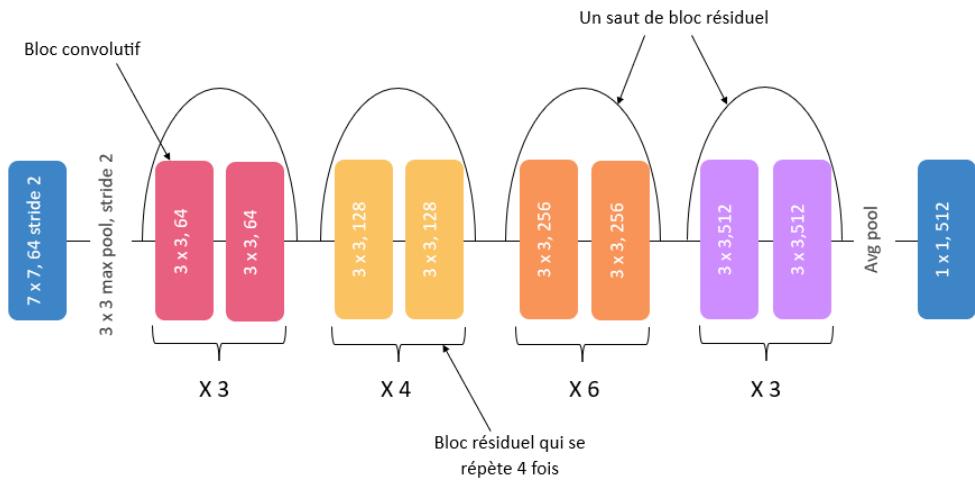


FIGURE 3.6 – Architecture de ResNet-34 modifiée

Ensuite, nous ajoutons une amélioration au modèle de ResNet-34 en introduisant un bloc de mécanisme d'attention avant la couche de sortie, comme présenté dans la Figure 3.7 suivante :

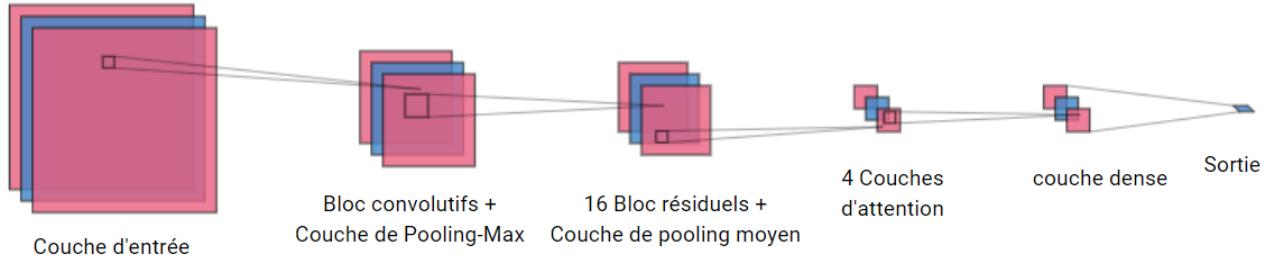


FIGURE 3.7 – Architecture de Attention ResNet

Nous résumons les 4 couches de mécanisme d’attention comme suit :

**Projector layer** : Une seule couche utilisée pour extraire les caractéristiques les plus importantes afin de réduire la matrice des caractéristiques (feature map).

**Couches d’attention spatiale** : Nous optons pour ce type de mécanisme d’attention pour les 3 couches restantes, car c’est le plus adéquat pour une classification d’images, vu qu’il aide le réseau à se concentrer sur les régions les plus pertinentes dans les documents.

Et enfin, nous utilisons la fonction d’activation SoftMax pour la décision de la classe.

Il est important de souligner que nous optons pour l’architecture de ResNet-34 plutôt que le ResNet-50, dans le but de prévenir tout risque de sur-apprentissage après l’ajout des couches de mécanisme d’attention.

### 3.4.3 Entrainement

Pour les deux modèles, nous utilisons l’optimiseur descente de gradient stochastique (stochastic gradient descent) pour réduire la perte définis par la méthode d’entropie croisée (cross entropy).

À notre connaissance, c’est la première méthode basée sur l’architecture ResNet avec des couches d’attention qui a été utilisée pour la classification de documents.

Cependant, comme nous l’avions mentionnée dans le chapitre précédent, peu de solutions dépendent seulement d’un modèle de vision pour classifier les documents. Nous pensons donc à une autre solution qui dépend des modèles plus adapté à ce genre de problème.

## 3.5 Solution basée sur la combinaison des approches visuelles et textuelles

Dans notre deuxième approche, nous nous concentrons sur les caractéristiques, la structure et le contenu des documents présents dans chaque classe.

En vue des différents types de documents, nous avons jugé que les données structurées comme la classe *table* et non structuré tel que la classe *handwritten* peuvent être facilement différencier par un modèle de vision. D’autant plus que ces deux classes ne contiennent pas de mots clés contrairement aux données semi-structurées telles que *id*, *resume*, *invoice* et *receipt*.

Nous avons donc pensé à une combinaison de deux modèles, un modèle de vision et un autre pour le traitement du langage naturel.

### 3.5.1 Pipeline

En premier lieu, nous avons utilisé un modèle de vision qui pourrait classifier les documents *handwritten*, *table* et une autre classe que nous appelons “*Autre*”, cette dernière va contenir un ensemble de données des classes restantes, à savoir, *resume*, *receipt*, *id* et *invoice*.

En deuxième lieu, une fois que le document est classé “*Autre*”, il va être transmis à un OCR pour extraire les éléments pertinents de l'image pour enfin les envoyer à un modèle de traitement du langage naturel qui va classifier ces éléments en fonction de son type.

Nous résumons la pipeline de notre deuxième approche par le diagramme suivant :

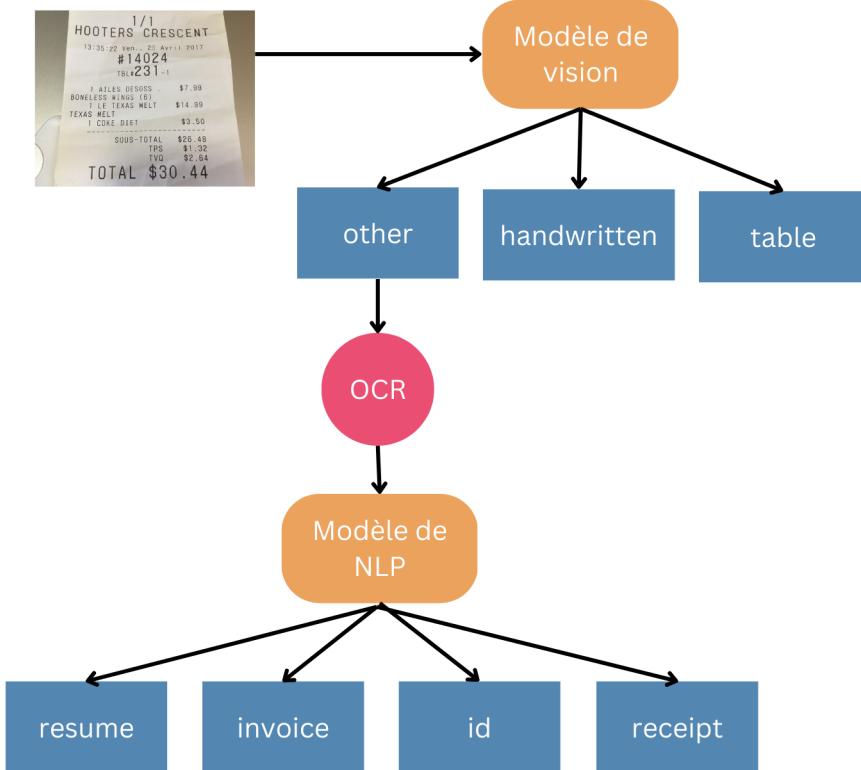


FIGURE 3.8 – Pipeline de la 2ème approche

Ce diagramme donne un enchaînement global sur l'idée que nous voulons modéliser. Dans ce qui va suivre, nous présentons le schéma détaillé de la deuxième approche.

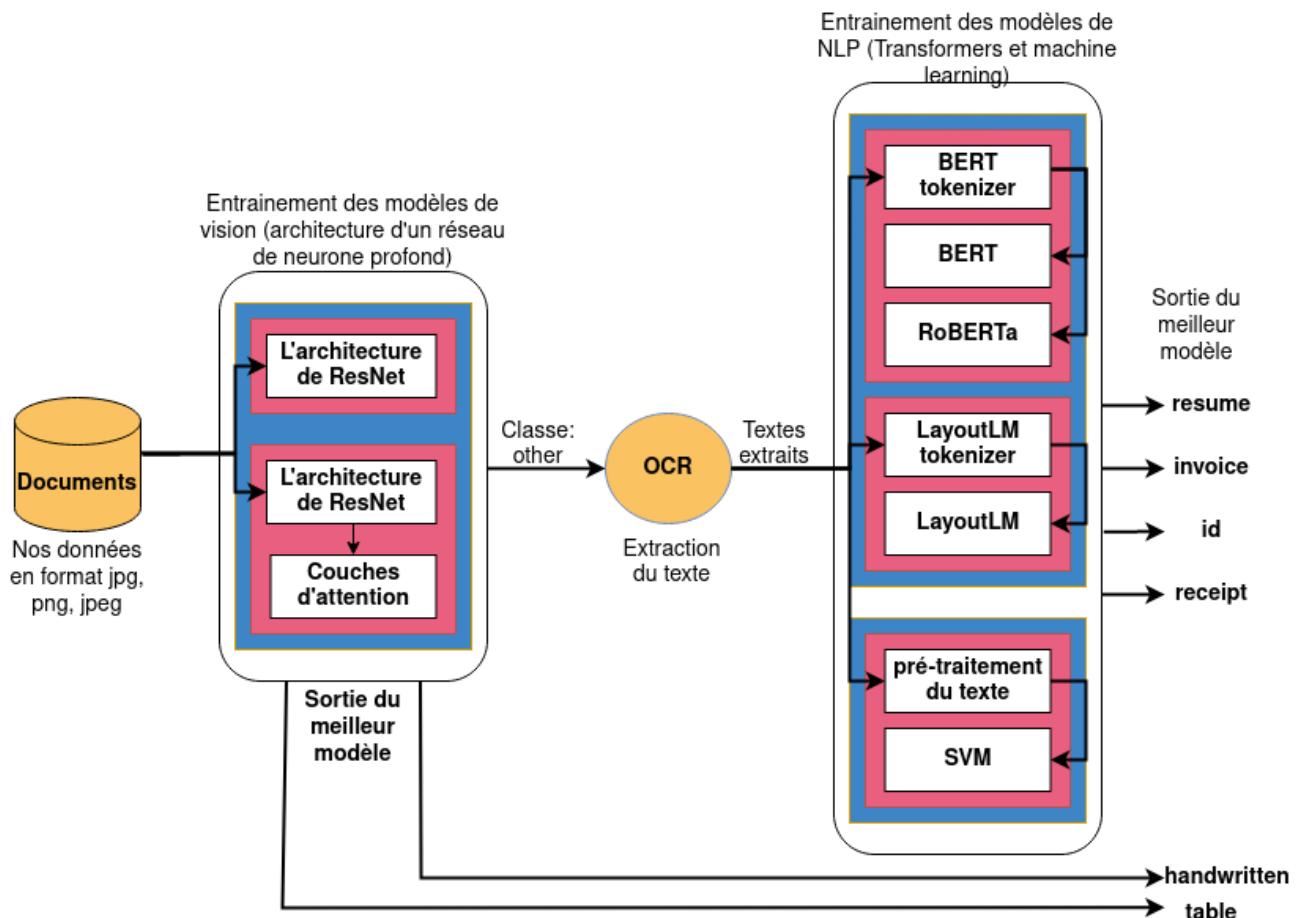


FIGURE 3.9 – Schéma détaillé de notre seconde approche

### 3.5.2 Modèle de vision

Dans cette section, nous adaptons les modèles qui ont été réalisés dans la première approche. Comme nous pouvons le voir dans le schéma ci-dessus, les deux modèles de vision, ResNet-50 et Attention ResNet-34 sont entraînés séparément. Seul le modèle avec les meilleures performances est pris en considération à la sortie de cette partie de l'architecture.

Le changement que nous effectuons au niveau des modèles de vision concerne le type de classe. En effet, comme nous l'avions mentionné dans l'introduction de notre pipeline, la sortie du modèle de vision consiste en trois classes, 'other', 'hanswriten' et 'table'. Rappelons que les deux dernières classes représentent une décision finale, contrairement à la classe 'other' qui indique que le document doit passer par un autre processus.

### 3.5.3 OCR

Nous utilisons dans cette partie un OCR existant, basé sur les techniques traditionnel d'un OCR vu dans le chapitre précédent.

#### Analyse des données

Cette partie nous est utile pour déterminer le prétraitement adéquat à appliquer à nos données.

- Nous extrayons le texte et les coordonnées 2D de ce dernier à partir des documents en utilisant l'OCR.
- Nous continuons notre analyse en déterminant la fréquence moyenne des mots par document pour chaque classe.
- Pour finaliser notre analyse, nous nous intéressons à la similarité des mots dans chaque paire de classe. Cela nous permet de repérer les mots en communs qui peuvent induire la classification du texte en erreur.

### Pré-traitement global

- Suppression les données vide extraites de l'OCR ainsi que les lignes dupliquées de notre dataset.
- Suppression de la ponctuation et des mots vides.

## 3.5.4 Modèles de NLP

Dans cette partie, nous entraînons plusieurs modèles de NLP. Chaque modèle a sa propre manière de représenter le texte et nous allons le voir en détail dans ce qui va suivre.

Il est important de souligner que seule la performance du meilleur modèle est prise en compte à la sortie de cette étape de l'architecture.

Nous avons entraîné deux types de modèles : un modèle de machine learning et d'autres modèles pré-entraînés qui dépendent des réseaux de neurones.

### 3.5.4.1 Modèle de machine learning

En vue de son adaptation aux données à grande dimension, nous choisissons le modèle SVM, qui est largement reconnu comme l'un des modèles de machine learning les plus utilisés dans la classification de texte.

En effet, l'algorithme est largement étudié avec des recherches approfondies, particulièrement dans la classification de texte.[61].

Cependant, nous allons tout d'abord parler de comment nous appliquons le pré-traitement sur le texte extrait de l'OCR.

### Prétraitement des données

- **Lemmatisation des termes :** Nous utiliserons cette méthode pour réduire la taille des mots. La lemmatisation ramène le mot à sa forme en dictionnaire (e.g. ‘certification’ devient ‘certify’).
- **Tokenization :** Nous divisons chaque texte d'un document en un vecteur de mots.
- **Représentation du texte :** Nous utilisons la méthode **Word2Vec** pour construire le vocabulaire. Cette méthode collecte des mots uniques du vocabulaire et leur attribue un identifiant. Une taille de fenêtre de contexte est définie pour déterminer le nombre de mots avant et après le mot cible à considérer comme contexte.

### Exemple :

*Tokens* : ['invoice', 'total', 'item', 'email', 'price']

On prend une taille de fenêtre égale à deux mots, on aura une matrice comme ceci :

	<b>invoice</b>	<b>total</b>	<b>item</b>	<b>email</b>	<b>price</b>
invoice	0	1	1	0	0
total	1	0	1	1	0
item	1	1	0	1	1
email	0	1	1	0	1
price	0	0	1	1	0

TABLE 3.1 – Exemple d’une représentation Word2Vec

Nous entraînons ensuite le modèle “continuous bag of words” (CBOW) pour construire nos embeddings de mots. Nous les mettons ensuite en entrée pour entraîner notre classifieur.

## SVM

Nous utilisons le Noyau (Kernel)<sup>1</sup> Linéaire, qui représente une frontière de décision (un hyperplan) qui sépare les différentes classes de textes de manière linéaire.

### Pseudo-code

Ci-dessous le pseudo-code du modèle que nous implémentons.

---

#### Algorithme 1 : SVM (Machine à Vecteurs de Support)

---

**Entrée :** Ensemble d’entraînement  $X$ , étiquettes  $y$ , paramètre de régularisation  $C$

**Sortie :** Poids de l’hyperplan  $w$  et terme de biais  $b$

```

1 : Initialiser  $w$  et  $b$  à zéro;
2 : Déterminer le nombre d’échantillons d’entraînement  $m$ ;
3 : Définir le taux d’apprentissage  $\eta$ ;
4 : Définir le nombre d’itérations  $T$ ;
for  $t = 1$  à  $T$  do
    for  $i = 1$  à  $m$  do
        Calculer la prédiction  $f(x_i) = \text{signe}(w \cdot x_i + b)$ ;
        Calculer la perte de charnière  $L = \max(0, 1 - y_i \cdot f(x_i))^2$ ;
        Calculer le gradient de la fonction de perte :  $\nabla_w L = \begin{cases} -y_i \cdot x_i & \text{si } L > 0 \\ 0 & \text{sinon} \end{cases}$ ;
        Mettre à jour les poids :  $w = w - \eta(w - C \cdot \nabla_w L)$ ;
        Mettre à jour le terme de biais :  $b = b - \eta(w - C \cdot \nabla_w L)$ ;
    end
end
5 : retourner  $w$  et  $b$ ;
```

---

Rappelons que le modèle de SVM est utilisé pour la classification binaire et non multi-classe, et donc il est nécessaire d’utiliser une stratégie qui nous permettra de l’adapter à un problème multi-classe. Nous optons pour ‘one-vs-rest’, une méthode qui divise notre

1. Noyau : prends en entrée les données et les transforme en une dimension supérieure dans le but de trouver une frontière de décision non-linéaire pour séparer les données d’origine.

dataset d'une manière à évaluer chaque classe par rapport à toutes les autres classes, considérées comme une seule classe.

## Entraînement

Nous utilisons l'algorithme ‘Stochastic Gradient Descent’ pour ajuster les poids et la fonction “squared\_hinge” comme fonction de perte.

### 3.5.4.2 Modèles pré-entraînés

Bien que le SVM soit adapté aux tâches de classification de texte, il repose sur un ensemble fixe de fonctionnalités qui peuvent ne pas capturer toutes les nuances du langage, contrairement aux transformers, tel que BERT, qui capturent le sens et les relations entre les mots avec plus de précision.

Les transformers sont également pré-entraînés sur de grandes quantités de données et peuvent être affinés, ce qui les rend plus adaptables et efficaces pour un plus large éventail de données que les SVM.

Avant de faire le fine tune sur nos modèles, les données doivent passer l'étape de tokenisation.

## Tokenisation

Les transformers disposent de tokenizers intégrés qui peuvent effectuer un pré-traitement sur le texte. Ces tokenizers divisent le texte d'entrée en jetons et les convertissent en représentations numériques pouvant être utilisées comme entrée dans le modèle. Le processus de tokenisation implique généralement la mise en minuscules, la suppression des caractères spéciaux et la division du texte en sous-mots ou mots, selon la configuration du tokenizer.

Dans cette partie, nous allons voir le processus de tokenisation de BERT qui est utilisé dans plusieurs autres modèles.

Le texte d'entrée est d'abord segmenté en mots à l'aide du tokenizer WordPiece, les jetons de mots résultants sont ensuite segmentés en jetons de sous-mots en utilisant la même technique.

**Exemple :** Supposons que nous voulions appliquer le tokenizer WordPiece à l'exemple donné, et que la taille de vocabulaire souhaitée est de 10.

**Phrase :** "invoice terms total service price invoice"

“i”	“n”	“v”	“o”	“t”	“e”	“s”	“r”	“c”	“p”	“w”	“l”	“f”	“u”	“a”	“ ”
3	3	3	3	3	2	2	1	1	1	1	1	1	1	1	5

Ensuite, nous fusionnons de manière itérative les paires d'unités de sous-mots les plus fréquentes en une seule unité jusqu'à ce que nous atteignions la taille du vocabulaire souhaitée de 10 :

```
Merge 1 : "in" + "voice" -> "invoice"
Merge 2 : "t" + "erms" -> "terms"
Merge 3 : "to" + "tal" -> "total"
Merge 4 : "ser" + "vice" -> "service"
Merge 5 : "pr" + "ice" -> "price"
```

Le tokenizer ajoute des jetons spéciaux tels que **[CLS]** et **[SEP]** pour définir le début et la fin de la séquence constituant le vocabulaire résultant.

```
["[CLS]"] + ["in", "voice", "terms", "total", "service", "price", "o", "n", "v", "t"] + ["[SEP]"]
```

Trois étapes succèderont la tokenisation :

1. **Token Embeddings** : Représentent chaque jeton sous la forme d'un vecteur de grande dimension. Ces imbrications sont apprises lors du prétraitement. Ainsi, quand nous utilisons le tokenizer sur notre input data, chaque mot est remplacé par l'identifiant numérique correspondant.
2. **Segment Embeddings** : Afin de faire la distinction entre les deux séquences, BERT ajoute un segment incorporant à chaque jeton indiquant à quelle séquence il appartient.
3. **Position Embeddings** : Sont ajoutées aux incorporations de jetons et fournissent des informations sur les positions relatives des jetons dans la séquence.

## BERT

Dans notre problème de classification, le séquencement n'est pas très important. En revanche, les mots-clés définissant chaque document le sont, d'où l'idée de geler les embeddings de position afin de permettre au modèle de se concentrer davantage sur l'apprentissage des schémas spécifique de la tâche de classification à partir des autres couches. Par conséquent, le freezing des embeddings rend l'entraînement plus rapide.

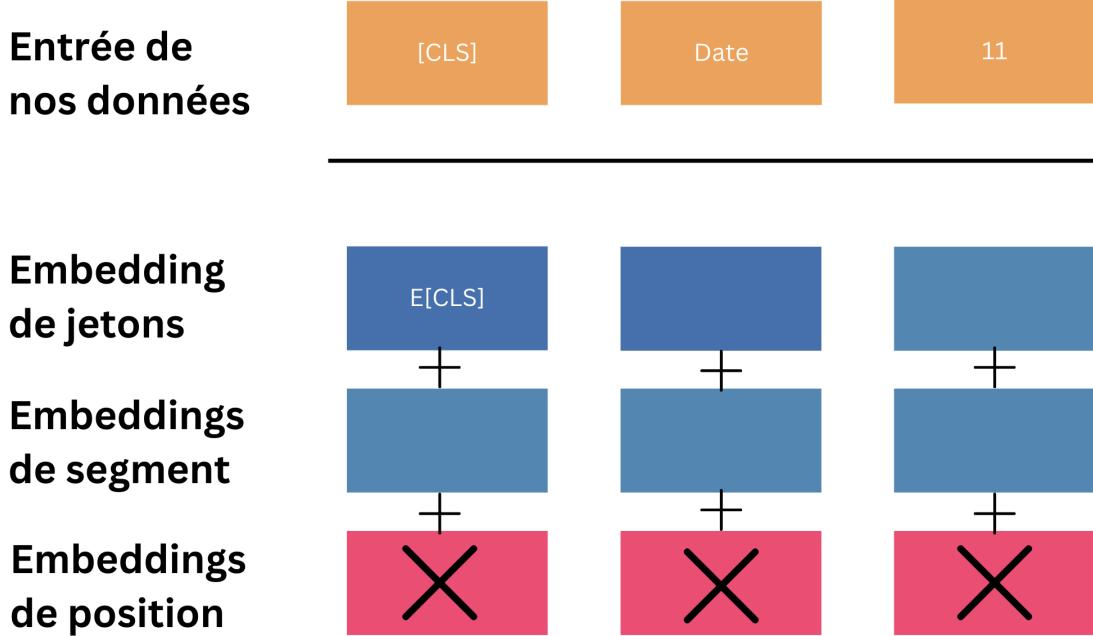


FIGURE 3.10 – Freezing des embeddings de position

## RoBERTa

Nous suivons le même principe concernant l'architecture de RoBERTA, pour voir si un modèle plus robuste de BERT pourrait mieux classifier nos données.

## LayoutLM

Afin de nous assurer d'un bon résultat, nous avons vu plusieurs modèles pré-entraînés spécialement conçus pour traiter la classification des documents. Beaucoup de ces modèles se basent sur la multimodalité, qui exploite l'aspect contextuel et visuel du document pour obtenir de meilleures performances.

Nous choisissons LayoutLM, car ce dernier n'a pas de licence commerciale, donc nous pourrons le déployer en production, contrairement aux autres versions du modèle comme LayoutLMv2 et LayoutLMv3.

Nous écartons aussi Ernie-Layout puisqu'il est plus adapté aux documents en chinois qu'en latin.

En ce qui concerne le modèle DocFormer, le point de contrôle pré-entraîné n'était pas disponible, ce qui nous aurait obligé d'entraîner le modèle à partir de zéro, qui est un processus long et gourmand en ressources. Il était donc nécessaire de prendre en compte la capacité de nos ressources à entraîner certains modèles.

Nous faisons donc le fine-tuning de LayoutLM. Comme nous l'avions mentionné dans le chapitre de l'état de l'art, LayoutLM utilise l'architecture de BERT comme backbone, cependant il n'est pas entraîné que sur le texte extrait du document, mais aussi sur les coordonnées de positions du texte dans ce dernier.

Tout comme BERT et RoBERTA, laoutLM utilise **wordpiece** pour générer les token et suit le même principe de tokenisation. En plus des jetons de mots et de sous-mots, LayoutLM génère également un ensemble de jetons visuels qui encodent les informations de position et de mise en page du texte.

## Entrainement

Nous utilisons l'optimiseur Adam avec cross entropy comme fonction de perte. Nous précisions que cela concerne tous les modèles pré-entraînés cités que nous entraînons.

**Méthode de régularisation :** Nous utilisons des techniques de régularisation lors de l'entraînement de nos modèles d'apprentissage automatique afin d'éviter le sur-apprentissage. Nous mentionnons ci-dessous les techniques que nous utilisons dans nos modèles.

- **Dropout**

À travers cette technique, nous désactivons d'une manière aléatoire un certain pourcentage de neurones à chaque mise à jour. Cela réduit les relations mutuelles entre les neurones et permet au modèle d'apprendre des représentations plus robustes et généralisables.

- **EarlyStopping**

Le but de cette méthode est d'arrêter l'entraînement du modèle juste avant le sur-apprentissage. Le sur-apprentissage peut être détecté par un résultat très petit de fonction de perte ou si la perte de la validation croisée a diminué, mais après un certain point, elle commence à augmenter.

Dans notre cas, nous ajoutons les deux conditions de détection.

Nous pensons à une troisième approche end-to-end afin d'éviter de passer par plusieurs étapes dans l'inférence.

## 3.6 Solution basée sur un modèle free OCR

### 3.6.1 Pipeline

Pour cette approche, nous proposons de fine-tune le **transformer Donut**. Comme nous l'avions mentionné dans la partie état de l'art, donut est un des modèles end-to-end les plus récents dans le domaine de la compréhension automatique de documents, qui intègre à la fois un transformer de vision et un autre transformer de NLP.

Donut est pré-entraîné avec des images de documents et leurs annotations de texte. Pendant la phase de fine-tuning, Donut apprend comment comprendre l'ensemble du document en fonction de la tâche en aval, dans notre cas, la classification de document.

La Figure 3.11 illustre le processus global du modèle Donut.

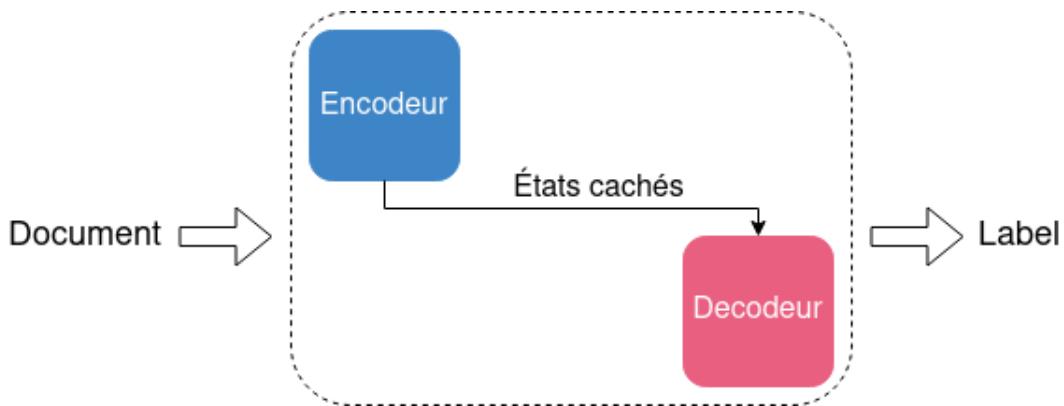


FIGURE 3.11 – Pipeline du modèle Donut

Donut ne dépend d'aucun module lié à la fonctionnalité de reconnaissance optique de caractères (OCR), mais utilise un encodeur visuel pour extraire des caractéristiques à partir d'une image de document donnée.

Le décodeur textuel suivant mappe les caractéristiques dérivées en une séquence de sous-tokens de mots pour construire un format structuré souhaité (par exemple, JSON). Chaque composant du modèle est basé sur un Transformer, ce qui permet un entraînement facile du modèle de bout en bout.

Le décodeur est entraîné à générer une séquence de tokens qui peut être convertie en un JSON représentant les informations de sortie souhaitées. Dans notre cas de classification de documents, le décodeur est entraîné à générer une séquence de tokens : [START class][invoice][END class], qui est inversible de manière bijective en : {« class »: « invoice »}.

Certains tokens spéciaux sont introduits (Par exemple, [invoice] est utilisé pour représenter la classe « invoice » ), et ce, contrairement à d'autres modèles qui prédisent l'étiquette de classe via un softmax sur l'encodage des embeddings.

### 3.6.2 Pré-Traitement

Nous utilisons les données collectées et non pas les données augmentées. En effet, la bonne qualité des images joue un rôle très important sur les résultats et la performance du modèle de Donut lors du finetuning.

### 3.6.3 Entrainement

Tout comme pour les modèles LayoutLM, BERT et RoBERTa, nous optons pour l'optimiseur **Adam** pour la descente du gradient, aussi, nous choisissons **l'entropie croisée** pour la fonction de perte.

## 3.7 Conception de l'application

Dans le but d'intégrer une de nos solutions dans la plateforme EdenAI, nous concevons une application web qui simule le fonctionnement de la plateforme de EdenAI.

Notre application est composée d'une interface web, qui offre à l'utilisateur la possibilité d'interagir avec nos différentes solutions et d'afficher certaines informations.

Nous mettons également en place une API REST<sup>2</sup>, qui assure la communication entre nos solutions et notre interface.

Pour finir, nous connectons nos solutions avec les différentes APIs OCR disponibles sur la plateforme EdenAI afin de réaliser des tests complets.

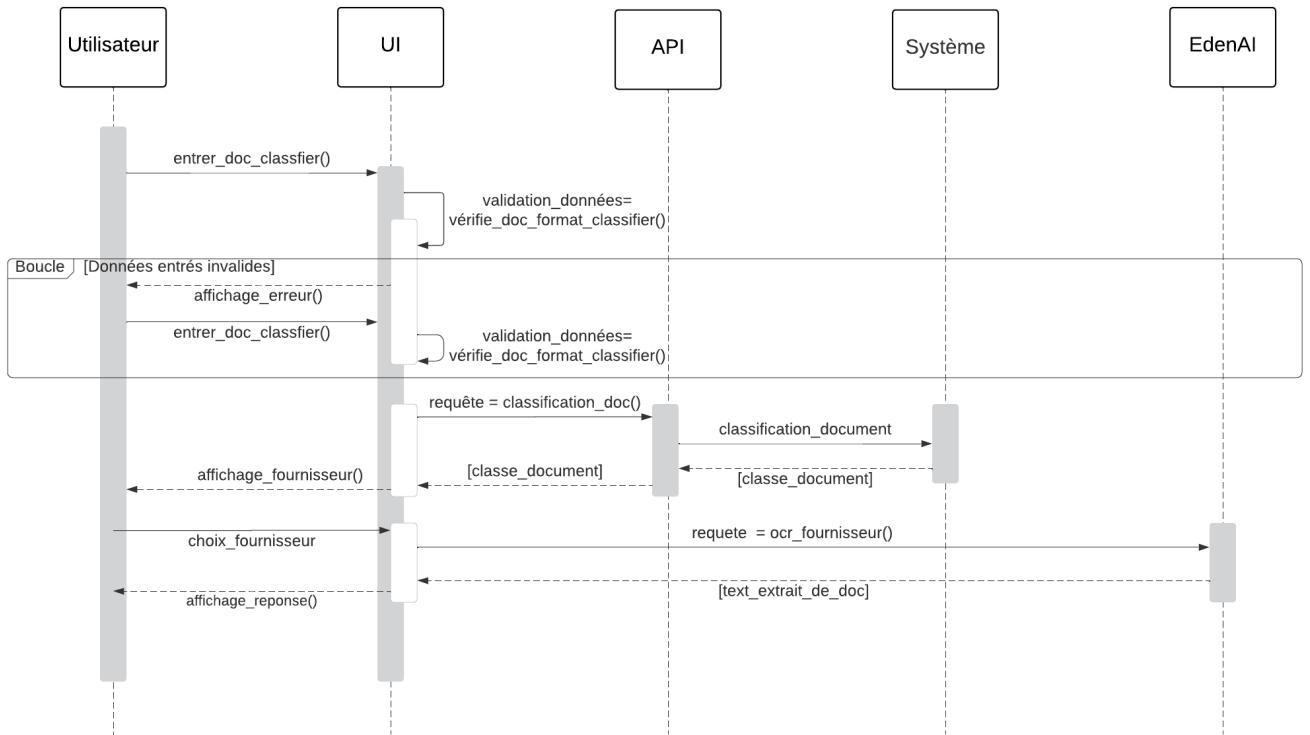


FIGURE 3.12 – Diagramme de séquence de notre application

L’utilisateur peut saisir le document dont il souhaite extraire des informations et choisir parmi les classifieurs disponibles, tels que ResNet, ResNet & BERT ou ResNet & LayoutLM... Si les données saisies sont valides, elles seront transmises à l’API qui fera appel aux modèles pour classifier le document à l’aide du classifieur choisi par l’utilisateur. Ensuite, les fournisseurs correspondant à la classe du document seront affichés, et l’utilisateur pourra sélectionner le fournisseur de son choix. Une requête sera ensuite transmise à EdenAI et le résultat sera alors affiché.

### 3.8 Conclusion

Dans ce chapitre, nous exposons les différentes solutions que nous avons développés pour résoudre notre problème. Dans le chapitre suivant, nous allons passer à l’implémentation de ces solutions et évaluer chaque approche. Cette évaluation nous permettra de déterminer la meilleure approche à déployer au sein de l’entreprise.

---

2. API REST : (Representational State Transfer) est un style d’architecture permettant l’échange de données entre systèmes via des requêtes HTTP standard, favorisant la communication simple et sans état entre le client et le serveur.

# Chapitre 4

## Implémentation et évaluations

### 4.1 Introduction

Après l'entraînement de nos différents modèles, il est important d'évaluer leur performance avant de choisir la solution idéale à déployer au niveau de l'entreprise. Dans ce chapitre, nous allons voir les différents outils utilisés pour implémenter nos solutions, le prétraitement que nous devons appliquer pour chaque approche, ainsi que les différentes métriques que nous avons appliquées pour les évaluer, et enfin, nous discuterons des résultats obtenus et de la réalisation de l'application finale.

### 4.2 Implémentation de nos modèles

#### 4.2.1 Outils

Nous utilisons une variété d'outils pour l'implémentation et l'entraînement de nos modèles, nous allons présenter quelques-uns :

##### NLTK

Natural Language Toolkit (NLTK)[21] est une bibliothèque open-source écrite en Python qui est largement utilisée pour le traitement automatique du langage naturel (NLP). Elle offre de nombreuses fonctionnalités et outils pour le prétraitement, l'analyse et la visualisation des données textuelles.

##### PyTesseract

PyTesseract [22] est une bibliothèque open-source en Python qui permet d'interagir avec le moteur OCR Tesseract, permettant ainsi d'extraire du texte à partir d'images dans différents formats tels que JPEG, PNG, TIFF, etc.

##### Pillow

Pillow [23] est une bibliothèque Python open-source largement utilisée pour le traitement d'images. Elle fournit de nombreuses fonctionnalités pour la manipulation et la modification des images.

##### Matplotlib

Matplotlib [24] est une bibliothèque de visualisation de données qui offre un large éventail de fonctionnalités pour créer des graphiques, des diagrammes et des visualisations de données de manière flexible et personnalisée.

##### Scikit-Learn

Scikit-Learn [25] est une bibliothèque open source, écrite en Python, C++ et Cython,

qui propose une large gamme d’algorithmes d’apprentissage automatique et métrique d’évaluations.

### PyTorch

Pytorch [26] est une bibliothèque open-source d’apprentissage automatique, dérivée de la bibliothèque Torch, développée par Meta et est implémentée par Python et C++. Elle permet d’effectuer des calculs tensoriels accélérés à l’aide de GPU, et fournit également une interface pour le développement des modèles de deep learning à un niveau élevé d’abstraction.

### Hugging Face

Hugging Face[27] est une plateforme qui offre un accès à une large collection de modèles pré-entraînés ainsi que des datasets spécifiquement conçus pour le traitement automatique du langage naturel (NLP) et de vision par ordinateur (CV). De plus, elle fournit une bibliothèque nommée **Transformers** qui est dédiée à l’utilisation et au développement personnalisés des modèles de traitement du langage naturel et des modèles de vision.

### Gensim

Gensim [28] est une bibliothèque open-source en Python utilisée pour le traitement du texte et la modélisation des sujets. Elle fournit des outils pour la création, la formation et l’utilisation des modèles de représentation des mots tels que Word2Vec.

### Google Colab

Google colab[29] est une plateforme gratuite qui permet d’exécuter et de développer des notebooks Jupyter directement depuis le navigateur. Il fournit un espace cloud prêt à l’emploi avec un accès gratuit à des ressources de calcul tel qu’un GPU de type NVIDIA Tesla K80 et une RAM à 12GO.

## 4.2.2 Caractéristiques de la machine locale

Système d’exploitation	Ubuntu 20.04.6 LTS x86_64
CPU	Intel i5-8250U
CPU fréquence	3.400GHz
CPU nombre de cœurs	8
RAM	8GO

TABLE 4.1 – Caractéristiques de la machine locale

## 4.2.3 Exploration des données visuelles

Dans cette partie, nous allons voir les détails de toutes les étapes qui ont été abordées dans le chapitre précédent concernant l’exploration de données.

### 4.2.3.1 Analyse des données

Nous utilisons la bibliothèque Matplotlib pour la réalisation de nos différents graphes.

#### Vérification du nombre de document par classe

Ci-dessous un graphe montrant le nombre des images dans chaque classe :

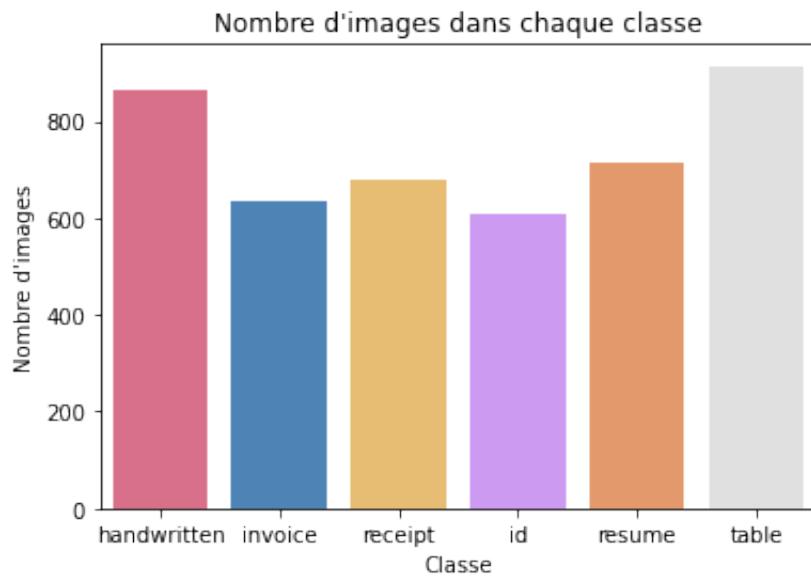


FIGURE 4.1 – Graphe montrant le nombre des images par classe

Comme nous pouvons le voir, il y a un léger déséquilibre entre les différentes classes présentes. Ce problème peut être réglé en sélectionnant un nombre adéquat d'images pour chaque classe, ou par une augmentation de données que nous allons voir plus tard dans cette partie.

#### Vérification de la taille des images

Nous présentons dans le graphe ci-dessous la taille des images de chaque classe en mégabytes.

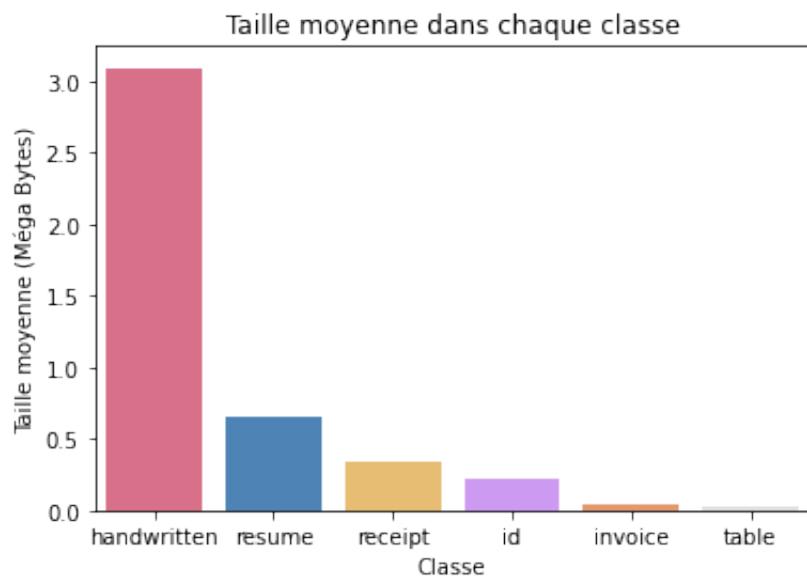


FIGURE 4.2 – Graphe montrant la taille moyenne des images dans chaque classe

L'analyse du graphe révèle des différences significatives de taille entre les différentes

classes. Il est donc essentiel de redimensionner les images afin de garantir une taille uniforme lors de l'entraînement de certains modèles.

Par ailleurs, il convient de prendre en compte les contraintes liées aux ressources disponibles, notamment la mémoire RAM et la capacité GPU.

### Vérification de la distribution de l'intensité des pixels

Ci-dessous un graphe montrant le nombre moyen des pixels dans les images de chaque classe :

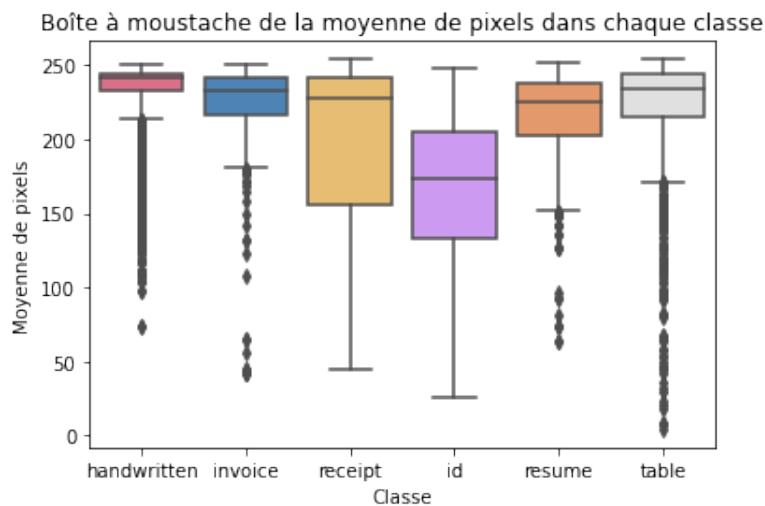


FIGURE 4.3 – Graphe montrant le nombre moyen des pixels dans les images de chaque classe

Nous pouvons constater, à partir de la boîte à moustache dans Figure 4.3 ci-dessus, que les classes id et receipt ne contiennent pas des outliers (valeurs aberrantes), ce qui indique que les images de ces classes sont similaires les unes aux autres. D'un autre côté, les autres classes contiennent un nombre considérable de outliers, ce qui suggère que les images d'une même classe présentent une structure différente.

#### 4.2.3.2 Nettoyage des données

Nous utilisons la bibliothèque Pillow pour les différents traitements faite sur les images.

**Traitement manuel :** Notre dataset contient beaucoup d'images non concernées par l'étude, de ce fait, nous avons supprimé manuellement, après vérification, plus de 16 images dans chaque classe.

**Traitement automatique :**

- Plus de 50 images dupliquées ont été supprimés dans chaque classe.
- Dans le but d'éliminer les images de faible résolution, nous procédons à la suppression de toutes les images dont la résolution est inférieure à  $150 \times 150$ . Nous avons choisi ce seuil, car nous jugeons cette taille comme étant la minimale pour préserver la précision et les détails de l'image.

#### 4.2.3.3 Augmentation des données

Nous rajoutons plus de 500 images augmentées dans chaque classe.

#### 4.2.4 Implémentation de la première approche

Pour implémenter la première solution, nous utilisons l'environnement de Google colab.

Nous entraînons les modèles ResNet-50 ainsi que Attention Resnet-34 de zéro avec notre dataset augmenté en prenant 940 images par classe.

Afin d'éviter l'arrêt soudain de l'entraînement causé par la consommation totale de la RAM ou du GPU dans notre environnement de travail, nous réduisons la taille de nos images à  $150 \times 150$  pixels avant de les envoyer en entrée aux modèles.

Nous utilisons la bibliothèque de PyTorch et nous ajustons les paramètres des modèles comme suit :

	ResNet-50	Attention ResNet-34
<b>Nombre d'epoch</b>	15	8
<b>Learning rate</b>	0.001	0.001
<b>Momentum</b>	0.9	0.9

TABLE 4.2 – Paramètres d'entraînement de la première approche

#### Notation

- **Nombre d'epoch** : chaque epoch correspond à une itération complète sur l'ensemble des données d'entraînement.
- **Learning rate** : ou le taux d'apprentissage, est un paramètre qui joue un rôle très important dans les calculs de l'optimiseur.
- **Momentum** : comme mentionné dans le chapitre précédent, nous avons utilisé l'optimiseur SGD avec momentum qui est une version optimisée de SGD classique.

#### 4.2.5 Implémentation de la deuxième approche

Dans la deuxième solution, nous travaillons sur deux environnements différents. Les modèles de machine learning ne nécessitaient pas un environnement GPU, donc nous les avons entraînés en locale, et en ce qui concerne les modèles pré-entraînés, nous utilisons Google colab.

Dans cette approche, nous avons eu besoin d'un OCR et donc nous avons utilisé la bibliothèque PyTesseract pour extraire le texte. Nous avons aussi exploité les différentes fonctionnalités présentes dans la bibliothèque NLTK pour le pré-traitement du texte.

##### 4.2.5.1 Extraction et exploration des données textuelle

Nous avons plusieurs choix d'API OCR que nous pouvons utiliser pour extraire les éléments pertinents de nos documents. Cependant, notre choix se porte sur Tesseract, en raison de sa grande communauté, sa grande précision dans la reconnaissance du texte dans les images particulièrement les textes en anglais, et aussi. De plus, il est considéré comme plus rapide que d'autres OCR tel que EasyOCR et KerasOCR en termes de temps d'exécution.

La Figure 4.4, illustre le résultat d'un petit programme que nous avons appliqué aux éléments extraits du document, à savoir le texte et ses coordonnées de position, et ce, en utilisant l'OCR Tesseract.



FIGURE 4.4 – Passeport annoté avec un cadre de détection de texte

### Analyse des données

Nous regroupons ces données, extraits de l'OCR tesseract, par classe et nous obtenons les informations suivantes :

Label	Fréquence du texte	Fréquence du text unique
id	574	536
invoice	351	347
receipt	620	612
resume	655	652

TABLE 4.3 – Tableau représentatif des fréquences du texte

Nous remarquons certains textes dupliqués, particulièrement dans la classe *id*.

Nous continuons notre analyse avec un diagramme déterminant la fréquence moyenne des mots par document pour chaque classe.

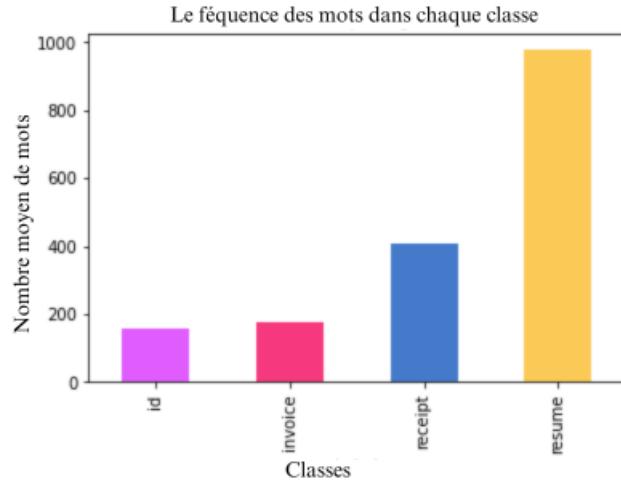


FIGURE 4.5 – Nombre moyen de mots dans chaque classe

Nous observons une disparité dans la fréquence des mots entre les différentes classes.

Afin d'avoir un aperçu global sur les mots présents dans nos données, nous utilisons un nuage de mots, généré à l'aide de la bibliothèque WordCloud, comme présenté dans la Figure 4.6.

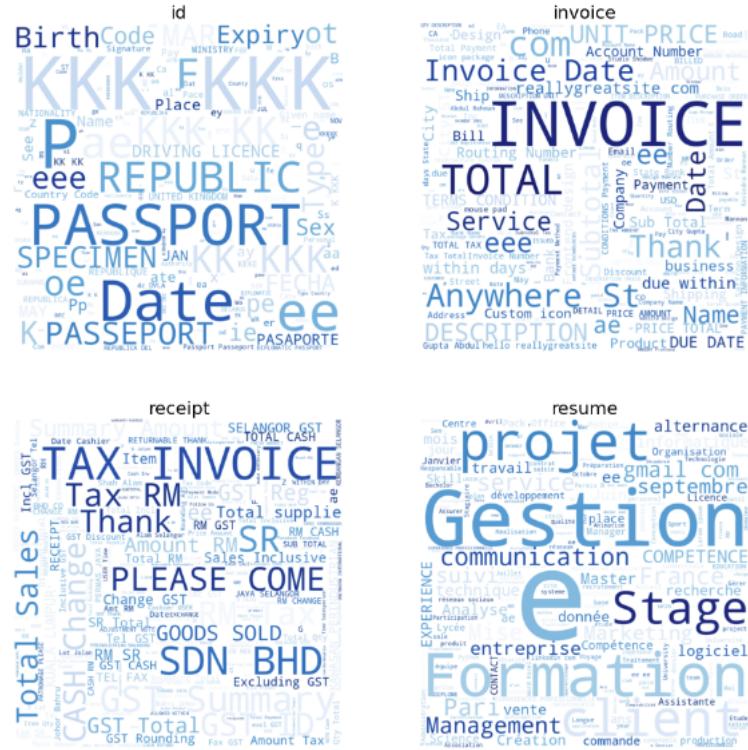


FIGURE 4.6 – Nuage de mots illustrant les mots les plus fréquent dans chaque classe

Ces nuages de mots permettent de visualiser les mots les plus fréquents dans chaque classe, en les représentant en taille plus grande. Nous pouvons constater que certains

mots apparaissent dans d'autres classes, mais avec une fréquence différente.

Pour finaliser notre analyse, nous nous intéressons à la similarité des mots dans chaque paire de classe. La Figure 4.7 illustre cette analyse.

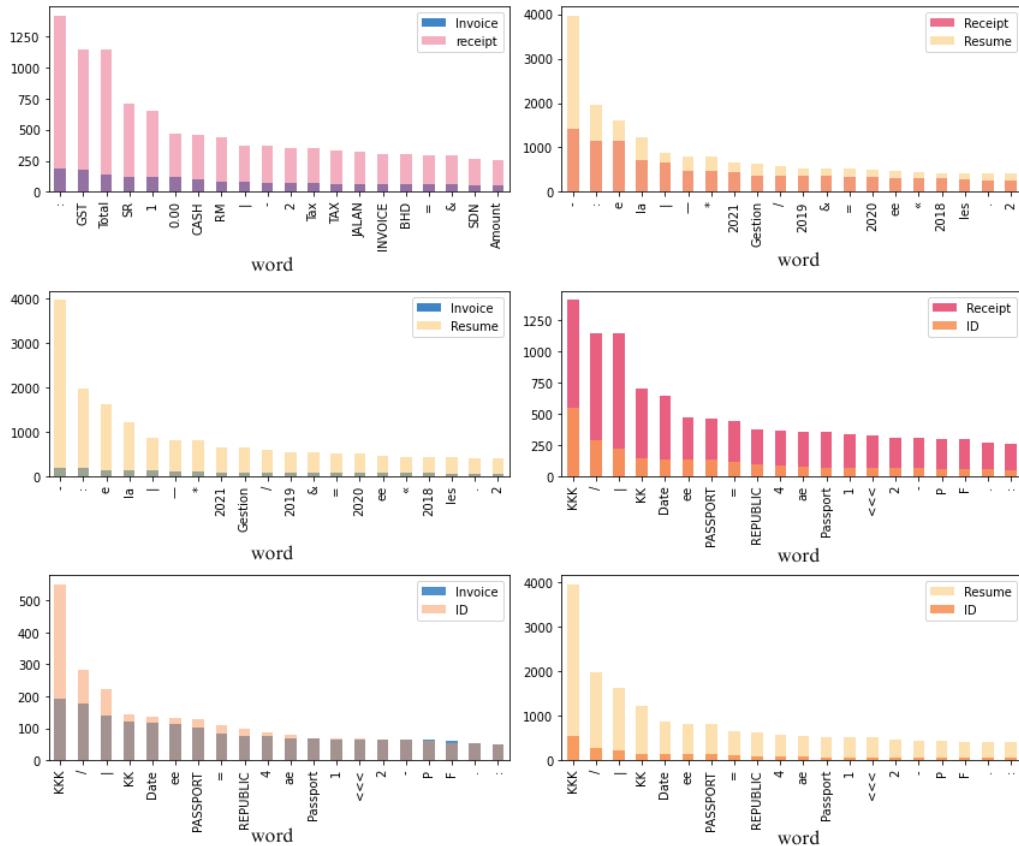


FIGURE 4.7 – Graphiques représentant les mots et leur fréquence dans chaque paire de classes

En comparant les graphes de différentes paires de classes, nous repérons certaines similitudes et différences dans la distribution des mots, par exemple le mot “total” qui est assez fréquent dans la classe *invoice* et *receipt*. Cela nous permet de repérer les mots en communs qui peuvent induire la classification du texte en erreur.

### Nettoyage des données

- Suppression de la ponctuation et des mots vides tels que ‘the’, ‘and’, ‘by’, etc. Nous avons particulièrement considéré les mots en anglais.

#### 4.2.5.2 Modèle de machine learning

Dans cette partie, nous appliquons la Lemmatisation des termes avec la base de données WordNet<sup>1</sup>.

Nous divisons ensuite nos textes en mots en utilisant le tokeniser de NLTK.

1. WordNet : est une base de données lexicale qui a pour but de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise.

Et enfin, nous appliquons la méthode Word2Vec pour la représentation du texte extrait des documents, et nous utilisons la bibliothèque **gensim** avec les paramètres suivants :

<b>vector_size</b>	25
<b>window</b>	10
<b>min_count</b>	1

TABLE 4.4 – Paramètres de Word2Vec

### Notation

- **vector\_size** : Ce paramètre représente la longueur du vecteur qui représente chaque mot dans le modèle.
- **window** : Ce paramètre spécifie la distance maximale entre le mot cible et les mots de contexte qui l'entourent.
- **min\_count** : Ce paramètre fixe le seuil de fréquence minimum des mots à inclure dans le vocabulaire.

Nous avons ensuite utilisé les mots encodés sur le classifieur LinearSVM, pour cela, nous avons utilisé la fonction présente dans la bibliothèque de scikit-learn avec les paramètres :

<b>C (paramètre de pénalité)</b>	1.0
<b>Penalty (Regularization)</b>	10

TABLE 4.5 – Paramètres d’entraînement de LinearSVM

### Notation

- **C (Paramètre de pénalité)** : Si le C est plus élevé, l’optimisation choisira un hyperplan de marge plus petit, de sorte que le taux de données mal classées soit faible. D’un autre côté, si le C est faible, la marge sera grande et elle peut entraîner davantage de mauvaises classifications.
- **Pénalité** : C'est un paramètre qui permet de réduire le sur-ajustement (overfitting). L2 également appelée "Ridge regularization" est un type de régularisation qui consiste à ajouter une pénalité proportionnelle à la norme L2 (c'est-à-dire la somme des carrés) des vecteurs de poids du modèle.

SVM est principalement utilisé comme classifier binaire. Nous l’adaptions donc à un problème multi-classe en utilisant la méthode **one-vs-rest** en divisant notre dataset d’une manière à évaluer chaque classe par rapport à toutes les autres classes, considérées comme une seule classe.

#### 4.2.5.3 Modèles pré-entraînés

	LayoutLM	BERT	RoBERTa
<b>Nombre d'epochs</b>	3.0	3.68	3.68
<b>hidden_dropout_prob</b>	0.1	0.4	0.3
<b>attention_probs_dropout_prob</b>	0.1	0.5	0.5

TABLE 4.6 – Paramètres d’entraînement des modèles pré-entraînés

**Notation :**

- **Nombre d'epoch** : nous avons utilisé le “early stopping”.
- **hidden\_dropout\_prob** : Spécifie la proportion de neurones qui sont aléatoirement ignorés (désactivés) lors de la phase d’entraînement.
- **attention\_probs\_dropout\_prob** : Spécifie la proportion de valeurs de probabilité d’attention qui sont aléatoirement ignorées (désactivées) lors de la phase d’entraînement.

#### 4.2.6 Implémentation de la troisième approche

La troisième approche, nous redimensionnons toutes nos classes, *handwritten*, *table*, *invoice*, *receipt*, *id* et *resume* en  $600 \times 400$  pixels.

Cette approche nécessite un environnement GPU avec suffisamment de RAM pour pouvoir l’entraîner, nous utilisons donc l’environnement de Google colab, et avons entraîné le modèle sur 10 epochs.

### 4.3 Evaluation

#### 4.3.1 Métriques d'évaluation d'un problème de classification

##### Matrice de confusion

Il s’agit d’une mesure de performance utilisée dans les problèmes de classification en apprentissage automatique, où la sortie peut prendre deux classes ou plus.

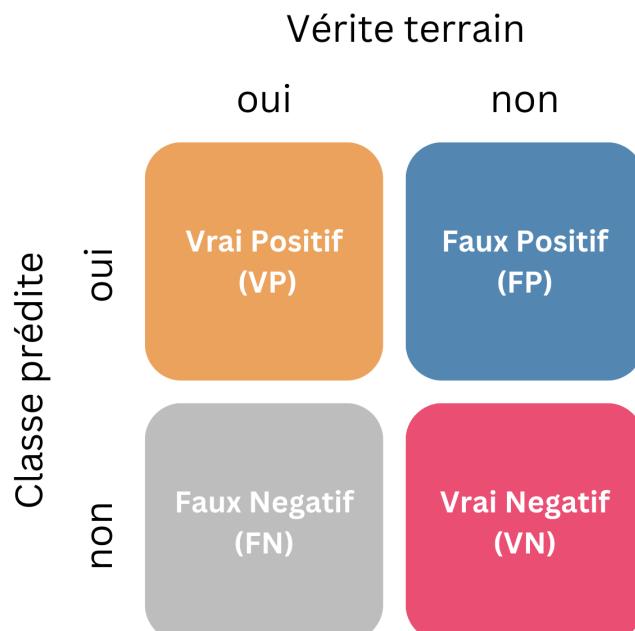


FIGURE 4.8 – Matrice de confusion

## Accuracy

Est une mesure permettant d'évaluer les modèles de classification. De manière informelle, l'accuracy est la fraction des prédictions que notre modèle a eu raison.

$$\text{accuracy} = \frac{VP + VN}{VP + FN + FP + VN}$$

## Sensitivité

Où le rappel, répond à la question, quelle proportion de vrais positifs a été identifiée correctement ?

$$\text{recall} = \frac{VP}{VP + FN}$$

## Précision

Répond à la question, quelle proportion d'identifications positives était réellement correcte ?

$$\text{precision} = \frac{VP}{VP + FP}$$

## Spécificité

Fait référence à la probabilité d'un test négatif, à condition qu'il soit réellement négatif.

$$\text{specificity} = \frac{VN}{VN + FP}$$

## F1-score

Également appelé score F1 est une mesure de la précision d'un modèle sur un ensemble de données. Il est utilisé pour évaluer les systèmes de classification binaires, qui classent les exemples en « positifs » ou « négatifs »

$$f1 - score = \frac{2 * VP}{2 * VP + FN + FP}$$

Les métriques ci-dessus sont principalement utilisées pour la classification binaire. Nous avons donc utilisé la stratégie **one-vs-rest** pour traiter le problème de multi-classe.

### 4.3.2 Evaluation de nos différentes solutions

Nous évaluons nos solutions en utilisant les métriques mentionnées précédemment.

Nous comparons la taille de chaque modèle ainsi que le temps d'exécution associé à chaque modèle.

Modèle	Taille (MB)
ResNet	171.1
Attention ResNet	163.3
LinearSVM	123.1
BERT	439.0
RoBERTa	502.0
LayoutLM	450.8
Donut	771.7

TABLE 4.7 – Comparaison de la taille de nos différents modèles

La comparaison de la taille de nos modèles nous donne un aperçu sur l'efficacité en termes de consommation de mémoire et d'espace de stockage.

Modèle	Temps d'exécution (secondes)
ResNet	0.235
Attention ResNet	0.217
ResNet & LinearSVM	1.32
ResNet & BERT	1.67
ResNet & RoBERTa	1.9
ResNet & LayoutLM	4.1

TABLE 4.8 – Comparaison du temps d'exécution de nos différents modèles

La comparaison du temps d'exécution nous permet de choisir le modèle le plus adapté en fonction des exigences spécifiques de l'application, en équilibrant la précision et la vitesse d'exécution.

Rappelons que l'OCR tesseract est utilisé dans l'inférence des modèles NLP (LinearSVM, BERT, RoBERTa, LayoutLM) et que le temps d'exécution de ce dernier varie d'une image à une autre.

#### 4.3.2.1 Première approche

##### ResNet-50

Ci-dessus les graphes représentant l'historique de l'entraînement :

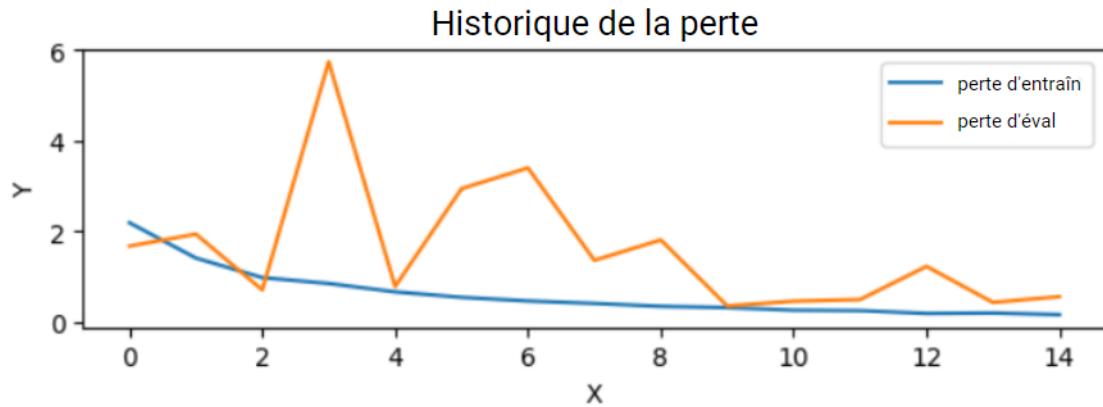


FIGURE 4.9 – L'historique de la perte d'entraînement et d'évaluation de Resnet-50

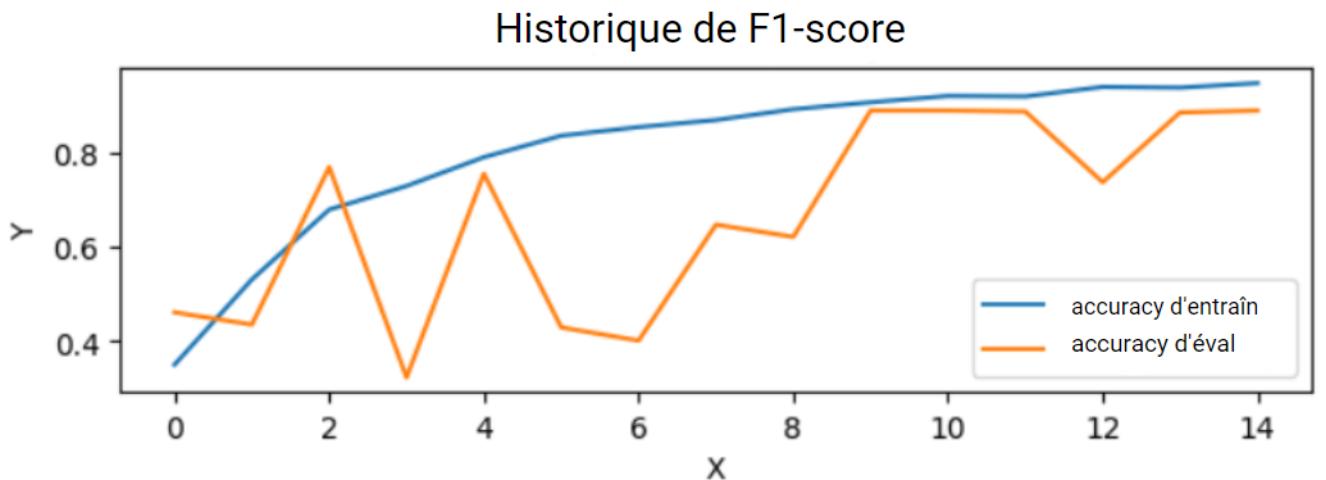


FIGURE 4.10 – L'historique de l'accuracy d'entraînement et d'évaluation de Resnet-50

Nous observons une instabilité du modèle, traduite à partir des oscillations des deux métriques présentées, ce qui indique que les poids du modèle continuent d'être ajustés dans le but de trouver une solution stable. Nous pouvons juger que le modèle n'a pas encore convergé vers une solution optimale.

	Precision	Recall	Specificity	F1-Score
Handwritten	0.96	0.95	0.94	0.96
ID	0.92	0.89	0.99	0.90
Invoice	0.98	0.87	0.98	0.92
Receipt	0.88	0.94	0.98	0.91
Resume	0.99	0.82	0.99	0.90
Table	0.73	0.94	0.98	0.82
Accuracy				0.90
Macro Avg	0.91	0.90	0.97	0.90

TABLE 4.9 – Métriques d'évaluation. ResNet-50

Nous remarquons que ce modèle a donné des résultats satisfaisants. Cependant, comme nous l'avons mentionné, il présente une instabilité dans ses prédictions.

Il est important de souligner que persister dans l'entraînement peut causer un sur-apprentissage, impliquant une précision très faible lors de la phase du test.

#### Attention ResNet-34

Ci-dessus les graphes représentant l'historique de l'entraînement :

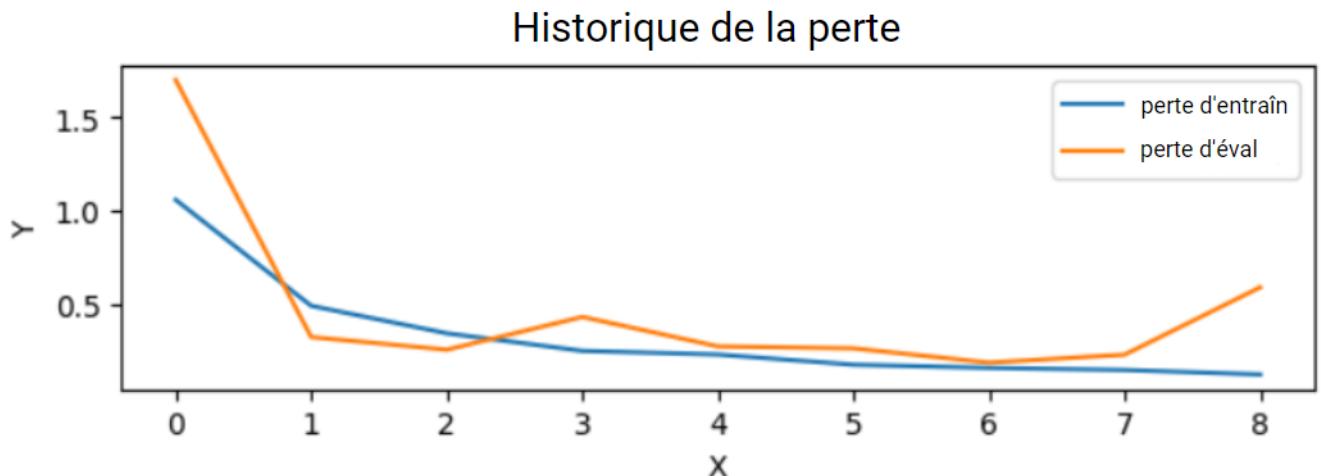


FIGURE 4.11 – L'historique de la perte d'entraînement et d'évaluation de Attention ResNet-34

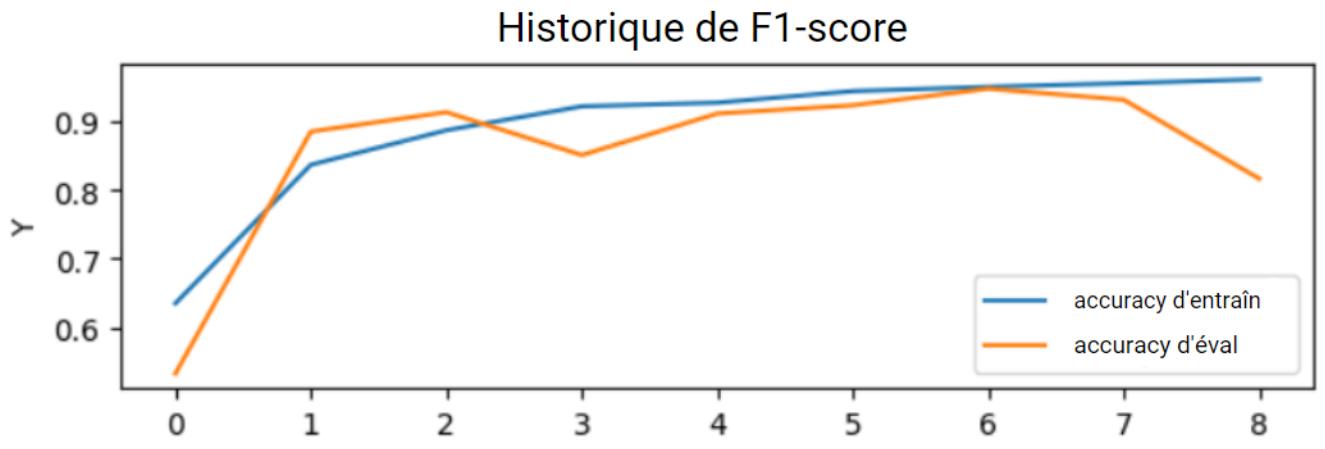


FIGURE 4.12 – L'historique de l'accuracy d'entraînement et d'évaluation de Attention ResNet-34

Nous pouvons constater que le modèle proposé est plus stable durant la phase de l'entraînement. Cependant, nous remarquons un léger sur-apprentissage dans ce modèle, malgré les nombreux tests effectué sur le nombre des epochs.

	Precision	Recall	Specificity	F1-Score
Handwritten	0.95	0.95	0.99	0.95
ID	0.93	0.94	0.98	0.93
Invoice	0.97	0.90	0.99	0.93
Receipt	0.98	0.88	0.99	0.92
Resume	0.84	0.99	0.96	0.91
Table	0.89	0.89	0.98	0.89
Accuracy				0.93
Macro Avg	0.93	0.93	0.98	0.93

TABLE 4.10 – Métriques d'évaluation. Attention ResNet-34

Nous pouvons voir que le modèle Attention ResNet-34 a donné de meilleurs résultats pour la première approche.

#### 4.3.2.2 Deuxième approche

##### Modèle de vision

Métriques		ResNet-50	Attention ResNet-34
Handwritten	Précision	0.95	<b>1.00</b>
	Rappel	0.97	<b>0.99</b>
	Spécificité	0.97	<b>1.00</b>
	F-score	0.96	<b>1.00</b>
Other	Précision	0.92	<b>0.97</b>
	Rappel	0.90	<b>0.96</b>
	Spécificité	<b>0.96</b>	0.94
	F-score	0.91	<b>0.96</b>
Table	Précision	0.93	<b>0.95</b>
	Rappel	0.93	<b>0.97</b>
	Spécificité	<b>0.96</b>	0.93
	F-score	0.93	<b>0.96</b>

TABLE 4.11 – Métriques d'évaluation pour chaque classe dans différents modèles de vision

Modèle	Precision	Recall	Specificity	F1-Score	Accuracy
ResNet-50	0.93	0.93	0.96	0.93	0.93
Attention ResNet-34	<b>0.97</b>	<b>0.97</b>	<b>0.97</b>	<b>0.97</b>	<b>0.97</b>

TABLE 4.12 – Performance générale des modèles

Nous remarquons que le modèle Attention ResNet-34 a donné les meilleurs résultats. Ainsi, nous allons le prendre pour le reste de nos évaluations.

## Modèle NLP

Métriques		LinearSVC	BERT	RoBERTa	LayoutLM
ID	Précision	0.87	<b>1.00</b>	0.98	0.98
	Rappel	0.84	0.94	0.93	<b>0.98</b>
	Spécificité	0.95	<b>0.99</b>	0.98	<b>0.99</b>
	F-score	0.86	0.97	0.95	<b>0.98</b>
Invoice	Précision	0.88	<b>1.00</b>	0.94	0.98
	Rappel	0.72	0.92	0.92	<b>0.98</b>
	Spécificité	0.96	0.98	0.97	<b>0.99</b>
	F-score	0.79	0.96	0.93	<b>0.98</b>
Receipt	Précision	0.86	0.97	0.94	<b>1.00</b>
	Rappel	0.97	<b>1.00</b>	0.94	<b>1.00</b>
	Spécificité	0.94	0.95	0.94	<b>0.99</b>
	F-score	0.91	0.96	0.94	<b>1.00</b>
Resume	Précision	0.87	0.97	0.92	<b>1.00</b>
	Rappel	0.95	<b>1.00</b>	0.98	<b>1.00</b>
	Spécificité	0.94	<b>1.00</b>	0.98	0.99
	F-score	0.91	0.98	0.95	<b>1.00</b>

TABLE 4.13 – Métriques d'évaluation pour chaque classe dans différents modèles NLP

Nous le résumons par le tableau ci-dessous :

	Precision	Reppel	Specificity	F1-Score	Accuracy
LinearSVC	0.87	0.87	0.95	0.87	0.87
BERT	0.97	0.97	0.98	0.97	0.97
RoBERTa	0.94	0.94	0.97	0.94	0.94
LayoutLM	<b>0.99</b>	<b>0.99</b>	<b>0.99</b>	<b>0.99</b>	<b>0.99</b>

TABLE 4.14 – Performance générale des modèles

## Résumé des métriques des modèles de la deuxième approche

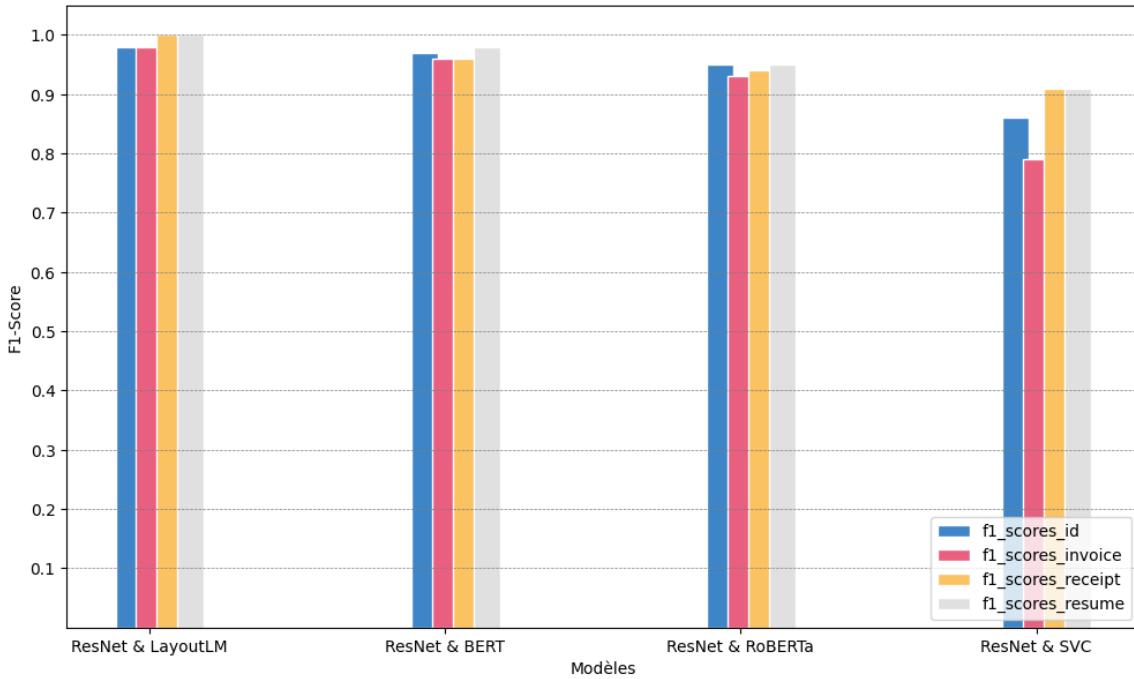


FIGURE 4.13 – Évaluation F1-score des modèles NLP

Nos observations indiquent que LayoutLM a obtenu les meilleurs résultats en termes de rappel, précision, spécificité et score F1, et ce, dans chaque classe.

L'une des erreurs que peuvent rencontré cette approche concerne la classe *invoice* qui parfois pourrait être classée comme classe *table*, car les factures contiennent souvent des tableaux.

### 4.3.2.3 Troisième approche

	Precision	Recall	Specificity	F1-Score
Handwritten	0.98	0.84	0.60	0.91
ID	0.74	1.00	0.80	0.85
Invoice	1.00	0.96	0.70	0.98
Receipt	0.89	0.75	0.65	0.81
Resume	0.92	1.00	0.90	0.96
Table	0.84	0.80	0.75	0.82
Accuracy				0.89
Macro Avg	0.89	0.89	0.73	0.89

TABLE 4.15 – Métriques d'évaluation. Donut

Nous remarquons que le modèle de Donut a donné d'assez bons résultats. Cependant, il est à noter que ce modèle est généralement entraîné sur un minimum de 14 epochs avec des images de grande résolution pour une meilleure performance.

L'approche de Donut est assez puissante, mais très coûteuse en ressources RAM/GPU.

### 4.3.3 Résumé global des métriques

Dans cette section, nous procéderons à une comparaison de toutes les solutions présentées en utilisant le score F1 comme métrique d'évaluation.

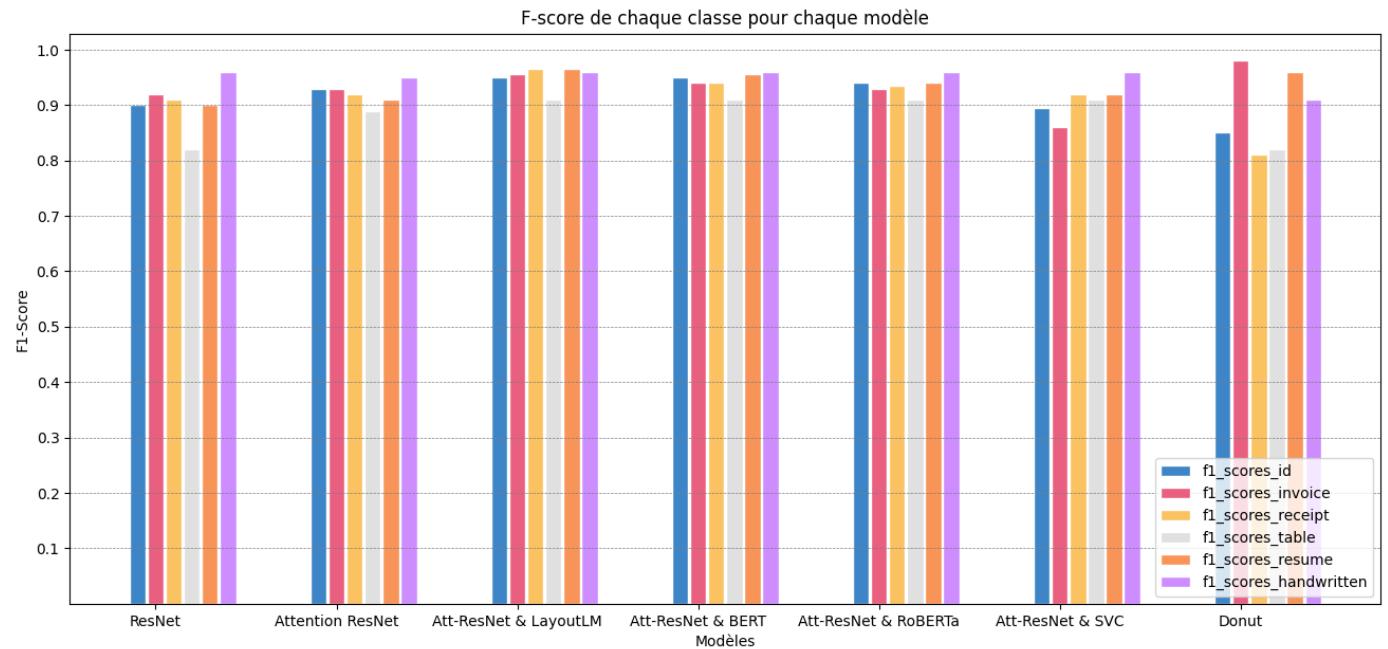


FIGURE 4.14 – Graphe de F-score de chaque classe pour chaque modèle

Nous présentons ci-dessous un graphe résumant le temps d'exécution et la performance en termes de F-score de tous les modèles implémentés.

Nous observons que tous les types de documents sont mieux classés avec la combinaison de ResNet et layoutLM, contrairement à donut par exemple qui contient plus de classe mal classé comparé aux autres modèles.

Nous remarquons aussi que notre modèle amélioré Attention ResNet34 a une meilleure performance que l'architecture de Resnet50 et qu'il réalise avec les modèles pré-entraîné.

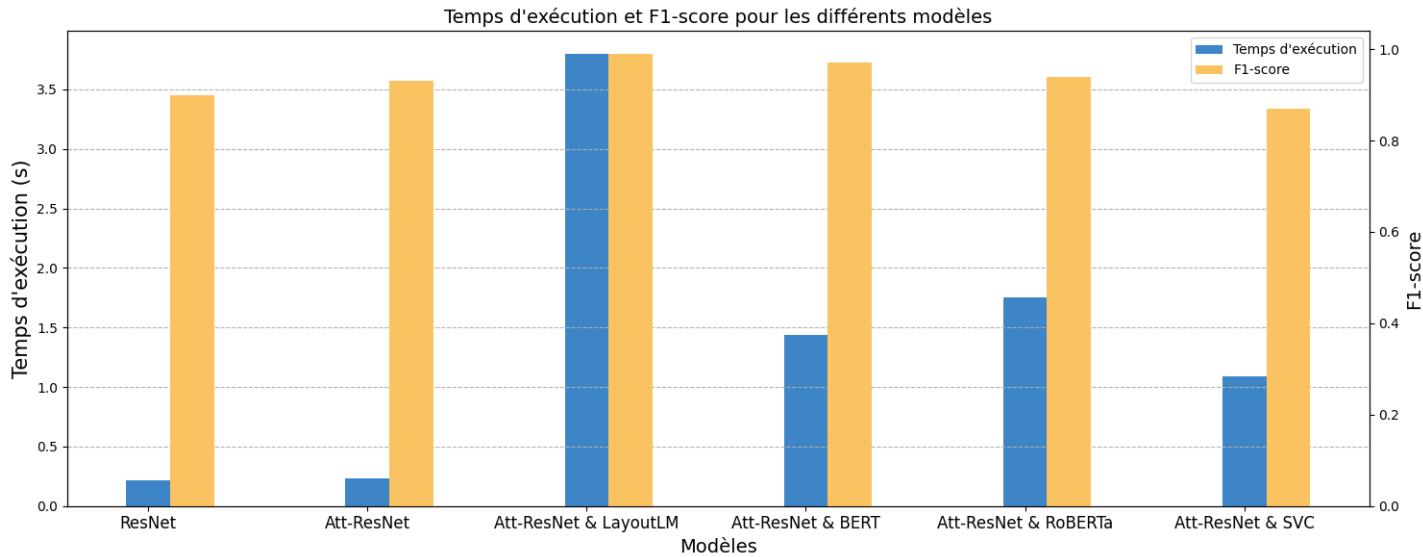


FIGURE 4.15 – Graphique du temps d'exécution et F-score pour chaque modèle

Nous remarquons qu'en termes de temps d'exécution, ResNet est le plus rapide, tandis que LayoutLM présente un temps d'exécution considérablement plus élevé.

Toutefois, BERT parvient à trouver un compromis entre le temps d'exécution et sa performance à classifier de nouveaux éléments.

Nous soulignons que le modèle Donut n'a pas été testé localement en raison de contraintes de ressources.

### Performances des modèles pré-entraînés sans fine-tune

Dans le tableau ci-dessous, nous comparons les différents classificateurs NLP, entraîné au préalable, avec le modèle de BERT que nous avons fine-tune.

Métriques		BERT	RoBERTa	LayoutLM	Donut
ID	Précision	0.32	0.28	0.00	0.44
	Rappel	0.97	1.00	0.00	1.00
	Spécificité	0.17	0.00	1.00	0.00
	F-score	0.48	0.44	0.00	0.62
Invoice	Précision	0.00	0.00	0.00	0.00
	Rappel	0.00	0.00	0.00	0.00
	Spécificité	1.00	1.00	1.00	1.00
	F-score	0.00	0.00	0.00	0.00
Receipt	Précision	0.00	0.00	0.33	0.00
	Rappel	0.00	0.00	0.17	0.00
	Spécificité	1.00	1.00	0.00	1.00
	F-score	0.00	0.00	0.22	0.00
Resume	Précision	0.68	0.00	0.14	0.00
	Rappel	0.34	0.00	0.5	0.00
	Spécificité	0.94	1.00	0.5	1.00
	F-score	0.45	0.98	0.22	0.00

TABLE 4.16 – Métriques d'évaluation des classes par rapport aux transformers avant le fine tune

Nous avons constaté une nette différence de performance entre le modèle sans fine-tuning et le modèle entraîné sur un jeu de données personnalisé. Le fine-tuning joue un rôle crucial dans la personnalisation des modèles pré-entraînés en les aidant à exploiter les caractéristiques spécifiques de la base de données utilisée.

### Exploration de GoogleOCR

À la demande de notre organisme d'accueil, nous avons essayé une autre méthode d'extraction de texte en utilisant l'OCR de Google afin d'améliorer la précision de nos modèles de NLP. Nous avons utilisé l'API de Google via la plateforme EdenAI pour extraire le texte, puis nous avons suivi les mêmes étapes de modélisation pour le fine-tuning de BERT. Le choix du modèle était basé sur la solution optimale obtenue avec l'OCR Tesseract.

	Tesseract	GoogleOCR
Temps d'exécution (secondes)	1.44	2.79
F1-score	0.97	0.97

TABLE 4.17 – Comparaison de la performance entre la solution utilisant Tesseract et la solution utilisant GoogleOCR

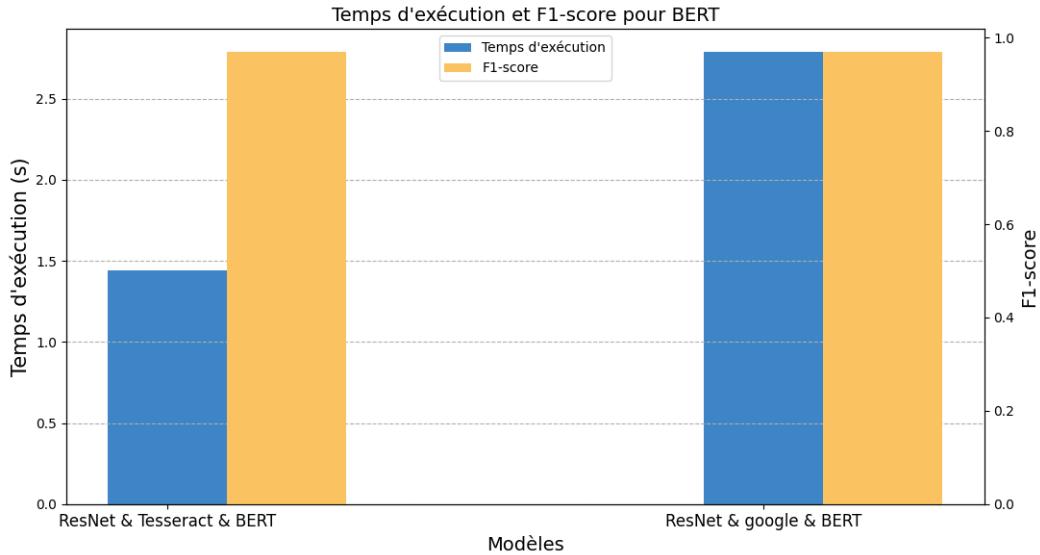


FIGURE 4.16 – Graphe du temps d'exécution et F-score Tesseract et GoogleOCR

Les deux modèles ont démontré des performances similaires. Cependant, il est important de noter que le modèle Tesseract présente un avantage en termes de vitesse d'exécution par rapport à l'OCR de Google.

## 4.4 Implémentation de l'application web

### 4.4.1 Outils

#### Git

Git [20] est un système de contrôle de version distribué, libre et open source, développé par le créateur de GNU/Linux, Linus torvald. Git est devenu au fil des années un outil indispensable pour gérer un projet, maintenir différentes versions et avoir un historique complet des modifications, ainsi que collaborer efficacement sur le projet.

#### FastAPI

FastAPI [30] est un web framework python open source pour le développement et la mise en œuvre des Restful API. Il prend en charge la programmation asynchrone, ce qui permet des performances élevées et une gestion efficace des requêtes.

#### Gradio

Gradio [31] est une bibliothèque python qui offre un développement d'application web simplifié pour la présentation des modèles d'apprentissage automatique.

#### Docker

Docker [32] est un outil open source qui permet d'encapsuler des logiciels dans des conteneurs. Ces conteneurs sont autonomes et comprennent leurs propres logiciels, bibliothèques et fichiers de configuration. Docker simplifie le cycle de développement, de déploiement et d'exécution des applications, en évitant les problèmes d'incompatibilité avec l'infrastructure en séparant l'application de cette dernière.

#### 4.4.2 Architecture générale

Comme décrit dans la conception de l'application (voir 3.7), notre application se compose d'une interface web créée avec le framework Gradio, qui permet aux utilisateurs d'interagir avec nos solutions disponibles via notre API, développée avec le framework FastAPI, ainsi qu'avec les APIs OCR disponibles sur la plateforme EdenAI.

#### 4.4.3 Backend

Notre API joue un rôle central en assurant la communication entre notre interface utilisateur et les composants responsables du pré-traitement des données et des différents modèles de classification. Elle permet à notre interface d'envoyer des requêtes et de recevoir des réponses pour traiter les documents.

#### 4.4.4 Frontend

Nous avons développé une application web utilisant Gradio pour simuler le fonctionnement de notre solution au sein de la plateforme EdenAI. Dans ce qui suit, nous présentons des captures d'écran qui illustrent l'interface de notre application et son utilisation :

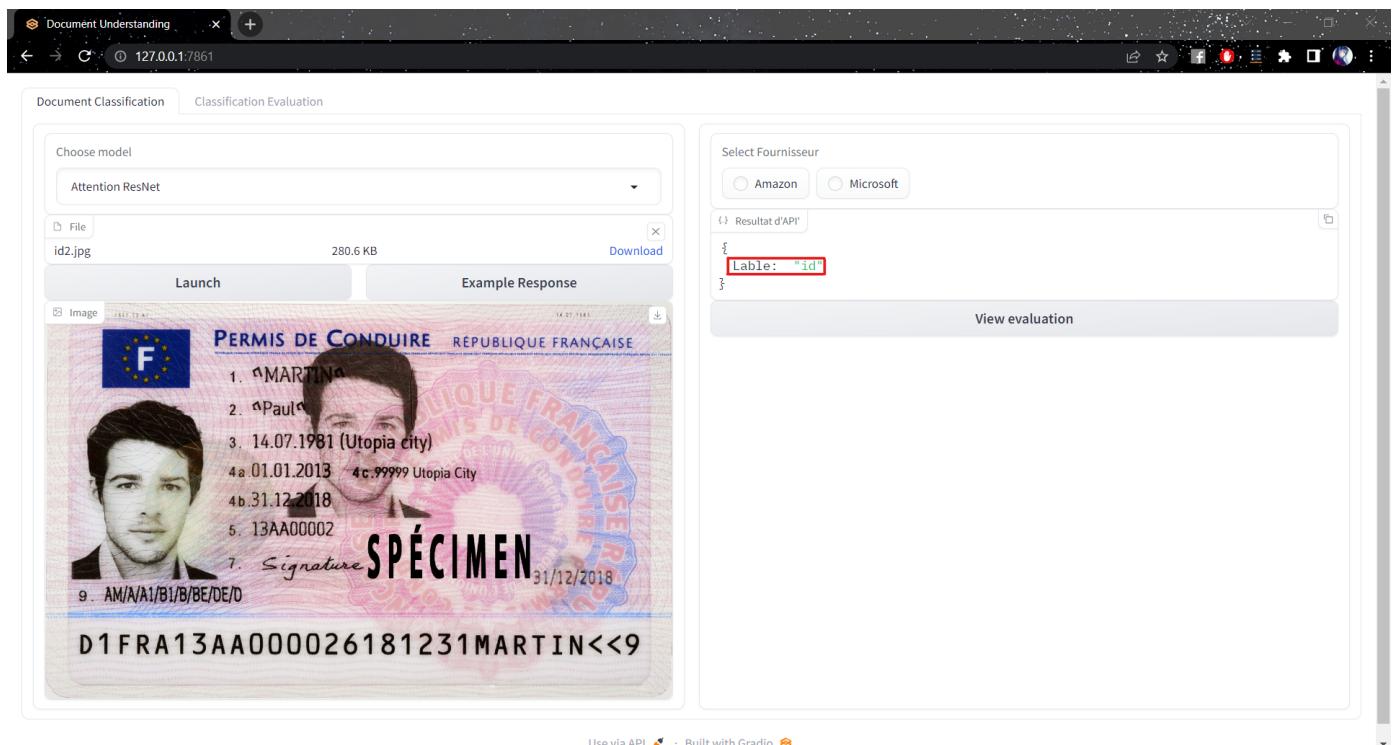


FIGURE 4.17 – Exemple de classification

Dans la Figure 4.17, l'utilisateur a choisi une image (de type *ID*) ainsi qu'un modèle de classification. Le résultat est affiché à droite, montrant une liste de fournisseurs correspondant à la classe *ID*, ainsi que la classe affichée juste en bas des fournisseurs sous-forme d'un JSON.

Dans le but de simplifier l'utilisation de l'interface, nous avons implémenté une fonctionnalité de sélection aléatoire d'images prédéfinies.

## Chapitre 4. Implémentation et évaluations

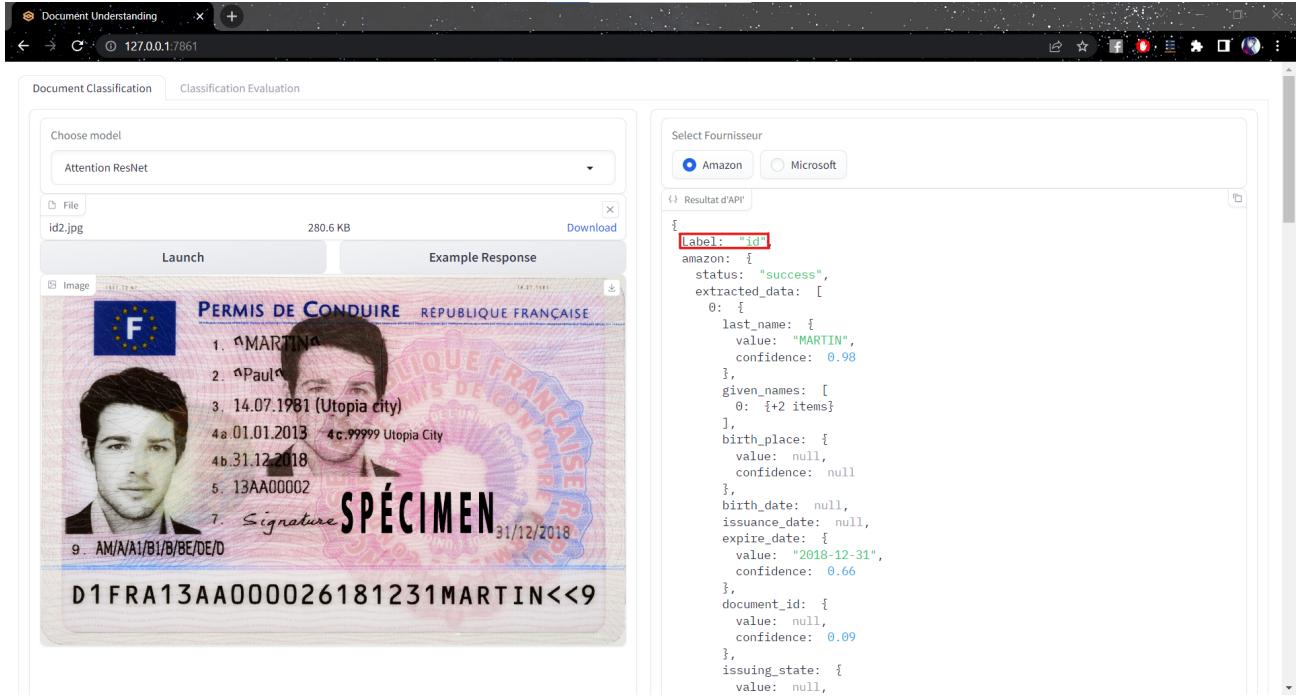


FIGURE 4.18 – Exemple de résultat d'un fournisseur choisi par l'utilisateur

Dans la Figure 4.18, l'utilisateur a choisi le fournisseur *Amazon*. Nous rajoutons la classe du document au résultat du fournisseur pour la démonstration. Enfin l'utilisateur peut copier le code JSON.

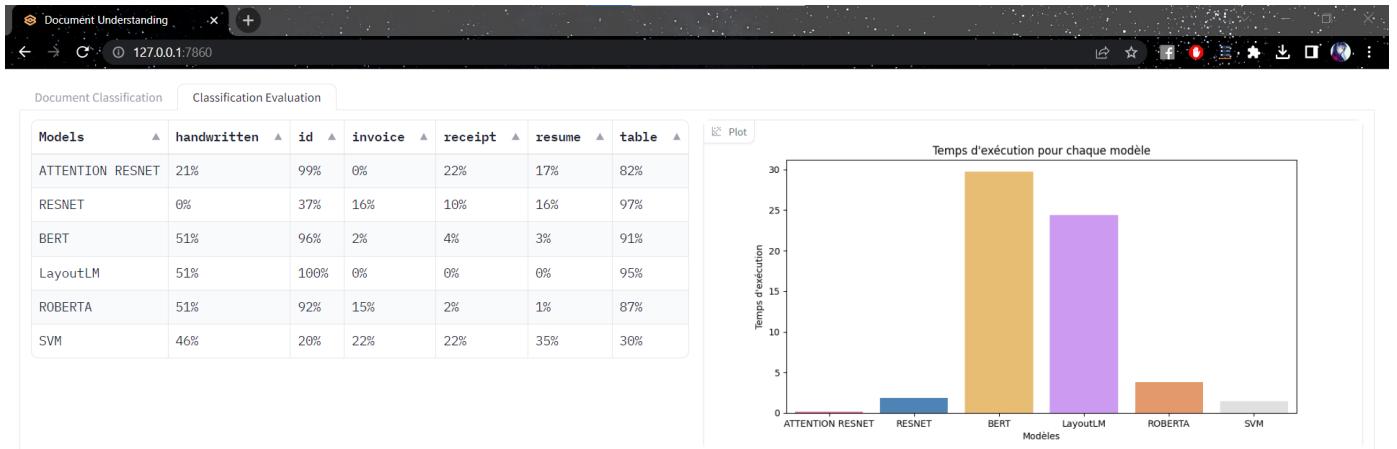


FIGURE 4.19 – Exemple de la fenêtre d'évaluation des modèles pour un document donné

Dans la Figure 4.19, nous présentons la partie évaluation. Les performances de chaque modèle sont présentées à l'aide d'un graphique représentant le temps d'exécution, ainsi que d'un tableau adjacent indiquant le pourcentage de classification de chaque classe par modèle.

#### **4.4.5 Déploiement**

Pour faciliter le déploiement de notre application ainsi que nos modèles, nous avons utilisé Docker pour conteneuriser les différentes composantes de notre application. Cette approche nous permet de simplifier le processus de déploiement et assure que l'application fonctionne de manière cohérente sur différents environnements.

# Conclusion et Perspectives

Dans le cadre de ce projet, nous avons développé plusieurs solutions qui reposent sur les techniques de vision par ordinateur et de traitement du langage naturel (NLP). Le but est de fournir une classification complète et précise des documents, en exploitant les forces spécifiques de chaque modèle pour chaque type de document.

Nous avons consacré une partie significative de notre travail à la recherche et à l'étude approfondie de l'état de l'art dans le domaine de la compréhension automatique des documents. Étant donné que ce domaine est vaste et a connu de nombreuses avancées ces dernières années, il était essentiel pour nous d'explorer les techniques existantes, en particulier dans le domaine de la classification des documents, qui est l'objectif principal de notre projet. Cette recherche approfondie nous a permis de réaliser une étude comparative entre nos différentes approches et méthodes, afin de mieux comprendre les forces et les limitations de chacune d'entre elles.

La phase de conception du projet a été particulièrement intense, car de nombreuses approches étaient envisageables et il était difficile de déterminer laquelle choisir et pourquoi. Nous avons donc pris la décision de mener une étude comparative en proposant trois approches différentes et en testant plusieurs méthodes. Nous avons commencé par une approche directe de classification d'image en entraînant le modèle ResNet, auquel nous avons ajouté nos propres améliorations en incorporant un mécanisme d'attention. Ensuite, nous avons développé une approche plus personnalisée en mettant en place une pipeline de classification. Cette pipeline divise les classes en "*table*", "*handwritten*" et la classe "*other*" qui regroupe les autres types de documents. Pour cette dernière catégorie, nous avons entraîné le modèle ResNet. Si la prédiction est "*other*", le document est traité par l'OCR Tesseract. Une fois le texte prétraité, nous l'avons envoyé aux différents modèles NLP que nous avons testés, à savoir, SVM, BERT, RoBERTa et LayoutLM. Enfin, nous avons exploré une approche end-to-end, en entraînant le modèle Donut. Ce modèle ne dépend pas d'un OCR, mais utilise un encodeur de vision pour traiter l'image et un décodeur NLP pour traiter le texte extrait.

En effectuant cette étude comparative, nous avons pu évaluer les performances et les avantages de chacune de nos approches en utilisant diverses métriques de classification, telles que le rappel (recall), la précision (precision), l'exactitude (accuracy), la spécificité (specificity) et le score F1 (f1-score) avec la méthode "one-vs-rest". Ces mesures nous ont permis d'avoir une vision globale des performances de chaque approche et de prendre une décision éclairée quant à la meilleure solution à adopter pour notre projet.

Après évaluation approfondie, nous avons initialement envisagé de déployer la combinaison du modèle d'attention ResNet pour la classification d'images avec le modèle BERT pour le tra-

tement du texte extrait, spécifiquement pour les documents en anglais. Cependant, étant donné que le modèle de vision n'est pas dépendant de l'aspect textuel et qu'il offre des performances similaires, nous choisissons de le déployer en raison de son avantage à généraliser sur un large éventail de langues.

En conclusion, nous considérons que les objectifs de notre projet ont été atteints, et ce, après avoir réussi le développement de la solution, ainsi que l'obtention de résultats satisfaisants avec nos approches par rapport aux modèles pré-entraînés existants.

Toutefois, il convient de souligner qu'il existe quelques points qui pourraient être améliorés. Parmi les principales idées et fonctions que nous pouvons rajouter :

- Les modèles que nous avons affinés ont démontré les performances les plus élevées, il serait donc pertinent d'élargir la classification des documents en plusieurs langues, au-delà de l'anglais.
- Représenter les textes avec d'autres techniques que word2vecs comme les sentences transformers (S-BERT) pour capturer la semantique des phrases avec plus de précision.
- Essayer une approche Zero-shot (NLI) pour classifier des documents jamais vu durant la phase d'entraînement (menu de restaurants, cartes de visites...)
- Travailler sur un environnement plus puissant en termes de ressource RAM/GPU pour pouvoir entraîner les modèles coûteux comme Donut et les modèles de vision en général.

# Webographie

- [1] Data genius. <https://www.datagenius.fr/>. Consulté le 2023-03-31.
- [2] Eden ai doc. <https://docs.edenai.co/docs>. Consulté le 2023-03-31.
- [3] A comprehensive guide to convolutional neural networks — the eli5 way. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Consulté le 2023-02-19.
- [4] Avijeet biswal, recurrent neural network(rnn), 2023. <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>. Consulté le 2023-03-16.
- [5] Amazon. <https://aws.amazon.com/>. Consulté le 2023-04-23.
- [6] Googl cloud. <https://cloud.google.com/apis>. Consulté le 2023-04-23.
- [7] Azure. <https://azure.microsoft.com/>. Consulté le 2023-04-23.
- [8] Open ai. <https://openai.com/>. Consulté le 2023-04-23.
- [9] Convolutional neural networks. <https://www.ibm.com/topics/convolutional-neural-networks>. Consulté le 2023-02-19.
- [10] What is optical character recognition (ocr)? [https://www.ibm.com/cloud/blog/optical-character-recognition?fbclid=IwAR3-pdwQG5Ux8hS5I35m2fMpcH0uoLJ5ligwnIETXrzealun5CBV05W6F\\_c](https://www.ibm.com/cloud/blog/optical-character-recognition?fbclid=IwAR3-pdwQG5Ux8hS5I35m2fMpcH0uoLJ5ligwnIETXrzealun5CBV05W6F_c). Consulté le 2023-03-23.
- [11] Avijeet biswal, recurrent neural network(rnn), 2023. <https://www.deeplearning.ai/resources/natural-language-processing/>. Consulté le 2023-03-10.
- [12] Build your ai career. <https://www.deeplearning.ai/>. Consulté le 2023-05-03.
- [13] Document classification with machine learning : Computer vision, ocr, nlp, and other techniques (2023). <https://affitob-com.ngontinh24.com/article/document-classification-with-machine-learning-computer-vision-ocr-nlp-and-other-techniques>. Consulté le 2023-03-15.
- [14] Qu'est-ce que la régression logistique? <https://www.ibm.com/fr-fr/topics/logistic-regression>. Consulté le 2023-04-15.
- [15] Support vector machine models. <https://www.ibm.com/docs/en/spss-modeler/saas?topic=nodes-support-vector-machine-models>. Consulté le 2023-04-15.

## Webographie

---

- [16] What is a decision tree ? <https://www.ibm.com/topics/decision-trees>. Consulté le 2023-04-17.
- [17] What is random forest ? <https://www.ibm.com/topics/random-forest>. Consulté le 2023-04-16.
- [18] Roboflow. <https://roboflow.com/>.
- [19] kaggle. <https://www.kaggle.com/>.
- [20] Github. <https://github.com/>.
- [21] Nltk. <https://www.nltk.org/>.
- [22] Pytesseract. <https://tesseract-ocr.github.io/tessdoc/>.
- [23] Pillow. <https://python-pillow.org/>.
- [24] Matplotlib. <https://matplotlib.org/>.
- [25] Scikit-learn. <https://scikit-learn.org/stable/index.html>.
- [26] Pytorch. <https://pytorch.org/>.
- [27] Huggingface. <https://huggingface.co/>.
- [28] Gensim. [https://radimrehurek.com/gensim/auto\\_examples/index.html](https://radimrehurek.com/gensim/auto_examples/index.html).
- [29] Google colaboratory. <https://colab.research.google.com/>.
- [30] Fastapi. <https://fastapi.tiangolo.com/lo/>.
- [31] Gradio. <https://gradio.app/>.
- [32] Docker. <https://www.docker.com/>.

# Bibliographie

- [33] 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*, pages 5998–6008, 2017.
- [34] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words : Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.
- [35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [36] Ekraam Sabir, Stephen Rawls, and Prem Natarajan. Implicit language model in LSTM for OCR. *CoRR*, abs/1805.09441, 2018.
- [37] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018.
- [38] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *CoRR*, abs/1507.05717, 2015.
- [39] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.
- [40] Sang-Woo Kim and Jun-Mo Gil. Research paper classification systems based on tf-idf and lda schemes. *Human-centric Computing and Information Sciences*, 9(1) :30, 2019.
- [41] Philipp Siebers, Christian Janiesch, and Patrick Zschech. A survey of text representation methods and their genealogy. *IEEE Access*, 10 :96492–96513, 2022.
- [42] Tapas Nayak and Hwee Tou Ng. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, New York, NY, USA, February 2020. AAAI Press.
- [43] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. *CoRR*, abs/1909.09586, 2019.
- [44] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM : Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

## Bibliographie

---

- [45] Jimmy Lei Ba Diederik P. Kingma. Adam : A method for stochastic optimization. Retiré de [arXiv:1412.6980](https://arxiv.org/abs/1412.6980), 2017.
  - [46] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm : Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1192–1200, New York, NY, USA, 2020. Association for Computing Machinery.
  - [47] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. Layoutlmv2 : Multi-modal pre-training for visually-rich document understanding, 2022.
  - [48] Srujan Appalaraju, Bhargav Jasani, Bharath Urala Kota, Yuntian Xie, and R. Manmatha. Docformer : End-to-end transformer for document understanding. Retrieved from <https://arxiv.org/abs/2106.11539>, 2021.
  - [49] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3 : Pre-training for document ai with unified text and image masking, 2022.
  - [50] Qi Peng, Yufei Pan, Weihua Wang, Binjie Luo, Ziyang Zhang, Zhihao Huang, Tao Hu, Wei Yin, Yan Chen, Yue Zhang, Shuaicheng Feng, Yifan Sun, Hui Tian, Hua Wu, and Haifeng Wang. Ernie-layout : Layout knowledge enhanced pre-training for visually-rich document understanding. Retrieved from <https://paperswithcode.com/paper/ernie-layout-layout-knowledge-enhanced-pre>, 2022.
  - [51] Gyuwan Kim, Taejoong Hong, Myungjun Yim, Jinyoung Park, Jiho Yim, Woochan Hwang, Sangdoo Yun, Dongyoon Han, and Sunghun Park. Donut : Document understanding transformer without ocr. *CoRR*, abs/2111.15664, 2021.
  - [52] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer : Hierarchical vision transformer using shifted windows, 2021.
  - [53] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023.
  - [54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.
  - [55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
  - [56] Saifullah Saifullah, Stefan Agne, Andreas Dengel, and Sheraz Ahmed. Docxclassifier : Towards an interpretable deep convolutional neural network for document image classification. *TechRxiv*, 2022. Preprint.
  - [57] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART : Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv :1910.13461 [cs, stat]*, 2019.
-

## Bibliographie

---

- [58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.
- [59] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta : A robustly optimized bert pretraining approach. Retiré de [arXiv:1907.11692](https://arxiv.org/abs/1907.11692), 2019.
- [60] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1) :60, 2019. Retiré de <https://doi.org/10.1186/s40537-019-0197-0>.
- [61] Sayar Ul Hassan, Jameel Ahamed, and Khaleel Ahmad. Analytics of machine learning-based algorithms for text classification. *Sustainable Operations and Computers*, 3 :238–248, 2022.

---

## Résumé

Nous nous sommes concentrés lors de la réalisation de notre projet de fin d'étude à l'automatisation de la classification de documents afin de répondre aux besoins de l'entreprise EdenAI. Notre objectif principal était de créer une API capable de prendre en entrée un document et de retourner le type correspondant parmi les catégories suivantes : handwritten, id, invoice, receipt, resume ou table. Pour atteindre cet objectif, nous avons exploré différentes solutions et approches, en nous appuyant notamment sur les avancées technologiques les plus récentes dans le domaine, telles que la compréhension automatique de documents.

Nous avons envisagé plusieurs approches pour notre projet, en commençant par une approche directe utilisant le modèle ResNet pour classer les images. Ensuite, nous avons développé une solution plus personnalisée en mettant en place une pipeline de classification. Cette pipeline repose sur la combinaison de deux modèles : un modèle de vision et un modèle de traitement automatique du langage naturel (NLP). Enfin, nous avons exploré une approche end-to-end, en entraînant le modèle Donut. Ce modèle ne dépend pas d'un OCR, cependant, il utilise un encodeur de vision pour traiter l'image et un décodeur NLP pour traiter le texte extrait.

Nous avons ensuite évalué nos différentes solutions en utilisant des métriques de classification telles que le rappel, la précision, l'exactitude, la spécificité et le score F1. En raison de sa capacité à généraliser sur différentes langues, le modèle Attention ResNet s'est révélé être la meilleure solution pour le déploiement, offrant des performances optimales et un temps d'exécution efficace.

**Mots clés :** Deep Learning, Natural Language Processing (NLP), computer vision (CV), Document understanding, OCR.

## Abstract

During the implementation of our final year project, we focused on automating document classification to meet the needs of the company EdenAI. Our main objective was to create an API capable of taking a document as input and returning the corresponding type among the following categories : handwritten, id, invoice, receipt, resume, or table. To achieve this goal, we explored different solutions and approaches, relying on the latest technological advancements in the field, such as automatic document understanding.

We considered several approaches for our project, starting with a direct approach using the ResNet model for image classification. Then, we developed a more customized solution by implementing a classification pipeline. This pipeline combines two models : a vision model and a natural language processing (NLP) model. Lastly, we explored an end-to-end approach by training the Donut model. This model does not rely on OCR but uses a vision encoder to process the image and an NLP decoder to process the extracted text.

We then evaluated our different solutions using classification metrics such as recall, precision, accuracy, specificity, and F1 score. Due to its ability to generalize across different languages, the Attention ResNet model proved to be the best solution for deployment, offering optimal performance and efficient execution time.

**Key words :** Deep Learning, Natural Language Processing (NLP), computer vision (CV), Document understanding, OCR.

---