Rapport de stage

Classification et reconstruction de fragments de fichiers binaires pour l'investigation numérique

Master 2 Cybersécurité – Université de Caen Normandie / ENSICAEN

JAFJAF Nora
Encadrants de stage : Emmanuel Giguet, Tristan Benoit
Professeur encadrant : Patrick Lacharme
2025-2026

Remerciements

Je tiens tout d'abord à remercier Emmanuel Giguet et Tristan Benoit pour leur encadrement tout au long de ce stage. Leurs conseils, leur disponibilité et les échanges réguliers que nous avons eus m'ont permis de faire progresser mon travail et d'orienter mes expérimentations. La complémentarité de leurs compétences a été particulièrement enrichissante pour ce sujet.

Je remercie également Patrick Lacharme pour son suivi universitaire, ainsi que les membres de l'équipe SAFE pour leur accueil au sein du laboratoire GREYC. Ce stage m'a permis de découvrir le fonctionnement d'un laboratoire de recherche et d'approfondir mes connaissances dans le domaine de l'investigation numérique.

Enfin, je remercie l'Université de Caen Normandie et l'ENSICAEN pour l'accompagnement proposé au cours de cette dernière année de master, ainsi que mes proches pour leur soutien et leurs encouragements.

Table des matières

1 Introduction

Dans le cadre de ma dernière année de master Cybersécurité, j'ai pu effectuer mon stage au sein du laboratoire GREYC (Groupe de Recherche en Informatique, Image et Instrumentation) à Caen.

Ce stage, d'une durée de près de 20 semaines, s'est déroulé du 13 avril 2026 au 31 août 2026.
Ce stage représentait pour moi une double opportunité : d'une part, clôturer ma dernière année de master, synonyme de fin de mon parcours académique, et, d'autre part, acquérir une première expérience concrète dans le monde de la recherche. Ce deuxième aspect est d'autant plus intéressant car j'ai déjà pu effectuer un stage de près de 6 mois au sein d'une entreprise, et une expérience longue au sein d'un laboratoire de recherche me permettait ainsi de découvrir une approche différente et ainsi m'aider dans mes choix post-master.

Mon travail portait sur l'analyse de fragments de fichiers binaires dans un contexte d'investigation numérique. Il a progressivement évolué au cours du stage : une première partie a été consacrée à la classification de fragments et à l'étude des limites des différentes approches testées. Ces résultats ont ensuite amené à approfondir le cas des formats composites, notamment PDF et DJVU, avant de s'intéresser à la reconstruction de documents PDF à partir de fragments mélangés. Ce rapport présente ainsi ces différentes étapes, les expérimentations menées et les principaux résultats obtenus.

2 Présentation du GREYC

Comme dit précédemment, mon stage s'est déroulé au GREYC. Ce laboratoire de recherche est reconnu pour ses contributions originales, ses réalisations matérielles et logicielles mais aussi pour sa collaboration pluridisciplinaire dans le domaine des sciences humaines et sociales et le domaine des interactions de l'informatique avec les mathématiques et les sciences de l'ingénieur.

Ce laboratoire est né d'un regroupement de plusieurs enseignants-chercheurs d'informatique et d'électronique de Caen dans les années 80. Au fil des années, et toujours dans cette même volonté d'unité, l'ensemble des équipes de recherche universitaires de Caen de ces domaines se sont réunies donnant lieu à la création du GREYC en 1995. Depuis 26 ans, il est associé au CNRS, à l'université de Caen Normandie (UNICAEN) ainsi qu'à l'École nationale supérieure d'ingénieurs de Caen (ENSICAEN).

Avec plus de 180 membres, le laboratoire réunit des enseignants-chercheurs de plusieurs sites : Cherbourg, Vire, Lisieux ou encore Caen. Leurs travaux de recherche s'articulent principalement autour de trois axes : science des données, capteurs et instruments, algorithmes et intelligence artificielle.

Le laboratoire est ainsi composé de 6 équipes :

— AMACC : Algorithmes, Modèle de calcul, Aléa, Combinatoire, Cryptographie, Complexité
— CODAG : Contraintes, Ontologies, Données, Annotations, Graphes
— ÉLECTRONIQUE : études des capteurs à haute et faible sensibilité
— IMAGE : traitement et analyse de signaux, images et vidéos
— MAD : Modèles, Agents, Décision
— SAFE : Sécurité, Architecture, Forensique, biomEtrie

3 Équipe d'accueil et encadrement

3.1 L'équipe SAFE

Pour mon stage, j'ai été intégrée à l'équipe SAFE dont le responsable est Patrick Lacharme.
Leur thématique de recherche porte sur la sécurité informatique et s'articule autour de trois axes :

— Biométrie : définition et évaluation de systèmes biométriques, sécurisation des systèmes et des données biométriques.

— Architectures et modèles de sécurité : sécurité réseau, cryptographie appliquée, aléatoire et protection de l'information.

— Science de l'investigation (forensique) : traitement automatique du langage, plateforme forensique et protection de la vie privée.

Mon stage s'inscrit dans le cadre de ce troisième axe, consacré à l'investigation numérique.

3.2 Encadrement du stage

Tout au long du stage, j'ai pu effectuer des points hebdomadaires avec mes superviseurs : Emmanuel Giguet et Tristan Benoit. Ces points permettaient d'échanger sur l'avancement de mon travail et d'assurer le bon déroulement de mon stage.

4 Missions

4.1 Contexte

Ce stage s'inscrit dans le contexte de l'analyse et de la reconstruction de fragments de données numériques. Le sujet du stage prend son point de départ d'une difficulté classique en investigation numérique.

En effet, en investigation numérique, il arrive fréquemment qu'un analyste soit confronté à des données dégradées : fichiers effacés, zones mémoire partiellement réécrites, zones corrompues. Ainsi, dans ces situations, les éléments disponibles se présentent souvent sous la forme de fragments binaires incomplets, parfois très courts, et le plus souvent dépourvus d'en-tête ou de toute indication de format. Le file carving consiste ainsi à identifier et reconstruire des fichiers à partir de ces blocs, indépendamment de toute table d'allocation.

Les méthodes d'identification anciennes et classiques s'appuient sur les en-têtes, les *magic bytes* ou l'extension du fichier. Or, un fragment isolé, prélevé au milieu d'un fichier, ne contient presque jamais ces marqueurs. Il faut donc raisonner à partir de son contenu brut, c'est-à-dire à partir d'une séquence d'octets sans information explicite sur son format d'origine.

D'autres approches reposent sur des descripteurs statistiques calculés directement sur le fragment. Les fréquences de n-grammes permettent par exemple de représenter les successions d'octets, tandis que l'entropie de Shannon mesure le degré de désordre de leur distribution. Ces méthodes améliorent partiellement le problème, mais montrent des limites importantes dès que les fragments sont courts, hétérogènes ou inhabituels.

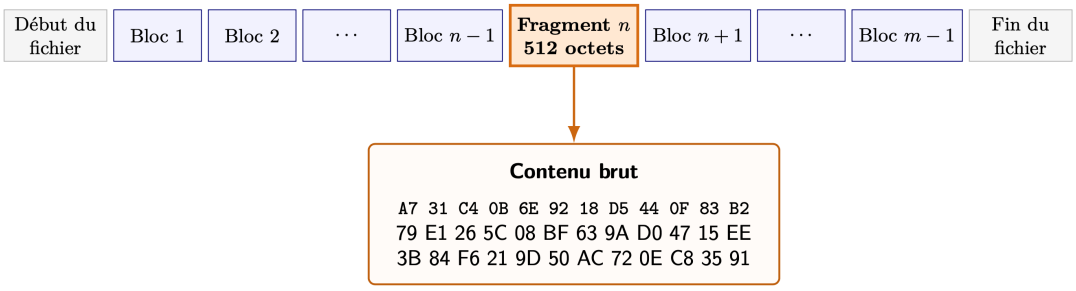

FIGURE 1 – Fichier source découpé en blocs de 512 octets

4.2 Objectifs du stage

L'objectif scientifique de ce stage est d'étudier une approche inspirée des modèles de langage, appliquée non pas à du texte, mais à des séquences binaires brutes. Il s'agit d'apprendre les régularités

présentes dans ces séquences afin de les exploiter dans un contexte d'investigation numérique.

La question centrale du stage est la suivante : **dans quelle mesure un fragment binaire court permet-il d'identifier le type de fichier dont il provient et d'aider à la reconstruction du fichier d'origine ?**

Cette problématique est étudiée à travers trois questions :

1. Quelles caractéristiques d'un fragment sont les plus utiles pour identifier son type de fichier ?
2. Quels formats restent difficiles à reconnaître lorsque la taille des fragments diminue, et quelles propriétés peuvent expliquer ces difficultés ?
3. Les relations entre fragments peuvent-elles être exploitées pour retrouver leur ordre et contribuer à la reconstruction d'un document ?

5 État de l'art

Le problème de la classification de fragments de fichiers (*File Fragment Classification*, FFC) est étudié depuis une vingtaine d'années en investigation numérique. Les travaux menés dans ce domaine reposent aussi bien sur des méthodes statistiques classiques que sur des architectures plus récentes d'apprentissage profond.

Cette partie présente, de manière non exhaustive, les principaux travaux existants. Elle s'intéresse aux représentations utilisées pour décrire les fragments, aux jeux de données employés, aux résultats obtenus et aux limites rencontrées. L'objectif est de repérer les éléments qui pourraient être utilisés pour ce travail.

5.1 Méthodes classiques : signatures et statistiques

Comme vu plus haut, l'identification par signature repose sur des marqueurs caractéristiques situés au début ou à la fin du fichier. Les *magic bytes* constituent l'un de ces marqueurs. Par exemple, un fichier PDF commence généralement par la séquence hexadécimale `25 50 44 46 2D`, qui correspond à `%PDF-`. Cette approche est stable et fiable sur un fichier complet, mais devient inopérante sur un fragment extrait au milieu du fichier, qui a peu de chances de contenir le marqueur recherché. Pour pallier cette limite, d'autres approches décrivent le contenu du fragment à partir de caractéristiques statistiques. L'entropie mesure le degré de dispersion des octets, tandis que les fréquences d'octets et les n-grammes décrivent leur distribution et leurs successions locales. D'autres mesures peuvent également être utilisées, comme le poids de Hamming. Ces caractéristiques sont ensuite fournies à des classifieurs comme les SVM, les Random Forest ou XGBoost. Sceadan, proposé par Beebe et al. [1], combine ainsi des unigrammes, des bigrammes, l'entropie et la longueur de la plus longue séquence d'octets identiques. Ces caractéristiques sont utilisées par un SVM à noyau linéaire pour prédire le type du fragment. Ces méthodes fonctionnent bien lorsque les formats présentent des distributions d'octets distinctes, mais leurs performances diminuent lorsque les statistiques des différentes classes se recouvrent. C'est notamment le cas des fragments compressés, chiffrés ou présentant une forte entropie. Elles montrent néanmoins que des représentations statistiques simples peuvent déjà contenir une information discriminante.

5.2 Apprentissage profond sur octets bruts

En 2020, Mittal et al. [2] proposent FiFTy, une contribution majeure à la classification de fragments de fichiers. Il s'agit d'un réseau neuronal convolutif (*Convolutional Neural Network*, CNN) 1D entraîné directement sur les octets bruts, sans extraction manuelle de caractéristiques.

Ils publient aussi FFT-75, le jeu de données utilisé plus tard dans ce stage. Il contient 75 types de fichiers et 102 400 blocs par type, pour des tailles de 512 et 4 096 octets. Il est décliné en six scénarios, du plus général au plus spécialisé. Le scénario #1 consiste à distinguer directement les 75 classes.

FiFTy y atteint 65,60 % d'exactitude à 512 octets, un score qui chute beaucoup par rapport aux 77,5 % obtenus à 4 096 octets, avec des erreurs concentrées sur les archives, les conteneurs et les formats composites.

5.3 Approches fondées sur l'image et information intra-octet

Chen et al. (2018) [3] convertissent un fragment binaire en image en niveaux de gris. Chaque octet devient un pixel, puis l'image est classée par un réseau convolutif.

L'intérêt est de montrer que certains formats produisent des textures visuelles régulières et exploitables, que les CNN peuvent apprendre à reconnaître.

L'approche est testée sur une sélection de 16 types de fichiers provenant de GovDocs, un corpus public constitué de documents collectés sur des sites gouvernementaux américains. Elle atteint cependant ses limites sur les fichiers à haute entropie. Un fragment compressé ou chiffré produit une image proche du bruit et peu discriminante.

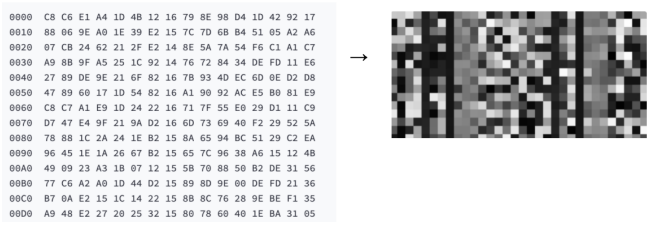

FIGURE 2 – Passage d'un fragment en octet bruts (tronqué) à une image en niveaux de gris

ByteNet [4], proposé par Liu et al., part d'un constat différent : une séquence d'octets traitée uniquement en 1D néglige l'information intra-byte, au niveau du bit.

La transformation Byte2Image applique des décalages de bits répétés pour révéler des motifs sous-octet, empilés en matrice 2D. Le modèle combine alors une branche 1D (relations entre octets) et une branche 2D (structures issues de Byte2Image). Deux variantes sont proposées : ByteResNet et ByteFormer, selon l'architecture utilisée dans la branche 2D.

L'intérêt annoncé porte notamment sur les formats utilisant un codage à longueur variable, comme JPEG, pour lesquels les relations pertinentes peuvent se situer à l'intérieur des octets.

Les résultats semblent confirmer l'intérêt de cette information complémentaire : sur le scénario le plus complexe de FFT-75, ByteFormer atteint 73,2 % d'exactitude à 512 octets, contre 65,60 % pour FiFTy. À 4 096 octets, les scores atteignent respectivement 81,9 % contre 77,5 %. Pour JPEG, ByteFormer obtient 96,0 % contre 83,5 % à 512 octets, et 95,1 % contre 86,3 % à 4 096 octets.

5.4 Approches Transformer

Un Transformer est une architecture fondée sur un mécanisme d'attention. Celui-ci permet au modèle de mettre en relation les différents éléments d'une séquence et d'identifier ceux qui sont les plus utiles pour réaliser la classification. Le Swin Transformer applique ce mécanisme dans des fenêtres locales décalées entre les couches, afin de combiner progressivement les informations locales et globales.

CarveFormer [5] adapte une architecture Swin Transformer V2 au traitement des fragments binaires. Le modèle travaille au niveau de l'octet : un fragment de 512 octets est représenté comme une séquence de 512 valeurs. Chaque octet est d'abord transformé en un vecteur de 96 dimensions par une couche d'*embedding*. La séquence obtenue est ensuite réorganisée sous une forme bidimensionnelle afin de pouvoir être traitée par le Swin Transformer.

CarveFormer obtient un taux de classification de 72,10 % sur le scénario #1 de FFT-75 à 512 octets. Ce résultat confirme l'intérêt des architectures attentionnelles pour modéliser les relations présentes dans une séquence d'octets.

5.5 Approches hiérarchiques

Dans une classification hiérarchique, les classes sont organisées sous la forme d'un arbre. La racine contient l'ensemble des classes, tandis que chaque branche les répartit progressivement en groupes plus restreints.

Un classifieur spécialisé peut ensuite être associé à chaque nœud ou à chaque feuille.

En 2020, Bhatt et al. [6] posent les bases de la classification hiérarchique appliquée aux fragments de fichiers. La hiérarchie est construite manuellement et un classifieur local est entraîné à chaque nœud. Ce travail montre aussi que les types de fichiers ne doivent pas nécessairement être traités comme 75 classes indépendantes et équivalentes.

En effet, certains formats partagent des propriétés communes et peuvent notamment contenir principalement du texte, des données binaires ou des données compressées.

Plus récemment, Zou et Liu [7] proposent une approche hiérarchique dans laquelle la hiérarchie n'est plus construite manuellement. Elle est dérivée automatiquement par clustering agglomératif, selon le critère de Ward, à partir de prototypes de classes qui correspondent aux vecteurs moyens des fragments de chaque classe.

Le clustering regroupe progressivement les classes qui présentent les prototypes les plus proches. Chaque feuille contient finalement entre 4 et 15 classes et dispose d'un classifieur spécialisé.

L'idée est de ne pas imposer à un seul modèle de séparer simultanément les 75 classes. Chaque expert travaille sur un sous-ensemble plus cohérent, ce qui peut faciliter la séparation de classes proches.

D'après l'analyse du code et du protocole, le routage vers les feuilles de la hiérarchie semble reposer sur une information issue du vrai label. Les auteurs filtrent les données de validation en vérifiant si leur étiquette réelle appartient aux classes de la feuille considérée en utilisant l'instruction `np.isin(y_val, leaf_labels)`. Chaque fragment est donc directement envoyé vers la feuille contenant sa classe réelle. Aucun classifieur intermédiaire n'est entraîné pour prédire la branche ou la feuille à partir du seul contenu du fragment.

Dans ces conditions, le taux de classification annoncé de 76,3 % sur le scénario #1 de FFT-75, avec des fragments de 512 octets, correspond plutôt à une borne supérieure sous l'hypothèse d'un routage parfait qu'à la performance d'une méthode de bout en bout dans laquelle un routeur devrait prédire automatiquement la bonne branche.

Dans l'ensemble, ces travaux montrent que la diminution de la taille des fragments réduit la quantité d'information disponible et dégrade les performances de classification. Les principales difficultés concernent les formats compressés, chiffrés ou composites, dont les fragments présentent peu de caractéristiques pour les distinguer ou peuvent contenir des données de natures différentes.

6 Caractérisation et classification des fragments sur le corpus maison

Cette première série d'expériences cherche principalement à répondre à deux questions. La première est de savoir quelles propriétés statistiques permettent de distinguer les types de fragments du corpus maison. La seconde est d'évaluer dans quelle mesure ces propriétés suffisent à les classer et comment les performances évoluent lorsque la taille des fragments diminue.

6.1 Corpus et protocole expérimental

Avant d'aborder un jeu de données de référence comme FFT-75, un premier corpus expérimental a été construit manuellement. L'objectif était de travailler dans un cadre contrôlé afin d'observer les distributions obtenues et de comparer le comportement de types de fichiers volontairement contrastés.

Le corpus a d'abord rassemblé quatre types de fichiers : TXT, DOC, JPG et PDF. Ils ont été choisis pour couvrir plusieurs situations, avec du texte lisible, des documents bureautiques, des images compressées et des formats documentaires composites.

Le corpus a ensuite été étendu à trois classes dérivées. Des fichiers chiffrés avec AES-256-CBC, encodés en Base64 et compressés au format ZIP ont été générés à partir des fichiers déjà présents dans les quatre premières classes. Cela permet de comparer un même contenu avant et après transformation, et donc d'évaluer l'effet propre du chiffrement, de l'encodage ou de la compression sur la structure statistique des fragments.

Les articles de l'état de l'art décrivent généralement le principe de fragmentation sans toujours détailler précisément le protocole employé. Un protocole propre a donc été défini pour ce premier cadre. Chaque fichier est lu en mode binaire, puis les 1 024 premiers et les 1 024 derniers octets sont retirés afin d'écarter les en-têtes, les marqueurs de fin de fichier et les signatures. Cette étape simule une situation proche de celle d'un fragment interne, dans laquelle les *magic bytes* ne sont pas disponibles. Elle évite également une classification triviale basée sur les premiers octets du fichier.

Le contenu restant est ensuite découpé en fragments de taille fixe. Une taille de 4 096 octets est d'abord utilisée, car elle correspond à une taille classique de bloc ou de cluster. Les expériences sont ensuite répétées avec des fragments de 512, 256 et 128 octets afin de mesurer l'effet de la réduction du contexte sur la difficulté de classification.

Le jeu de données final contient entre 102 600 fragments pour une taille de 4 096 octets et 104 820 fragments pour une taille de 128 octets, répartis entre les sept classes. La constitution et la répartition détaillées de ce corpus sont présentées en annexe A.1.

6.2 Quelles propriétés distinguent les différents formats ?

Sur chaque fragment, plusieurs descripteurs ont été calculés. L'entropie de Shannon mesure le degré de désordre de la distribution des octets. Une version locale, calculée sur des sous-blocs, permet d'estimer l'hétérogénéité interne des formats composites comme PDF. Les unigrammes correspondent à l'histogramme des 256 valeurs d'octets, tandis que les bigrammes et les trigrammes décrivent les régularités locales entre les octets successifs. En raison de leur dimension élevée, ces derniers ont été testés sous une forme hashée.

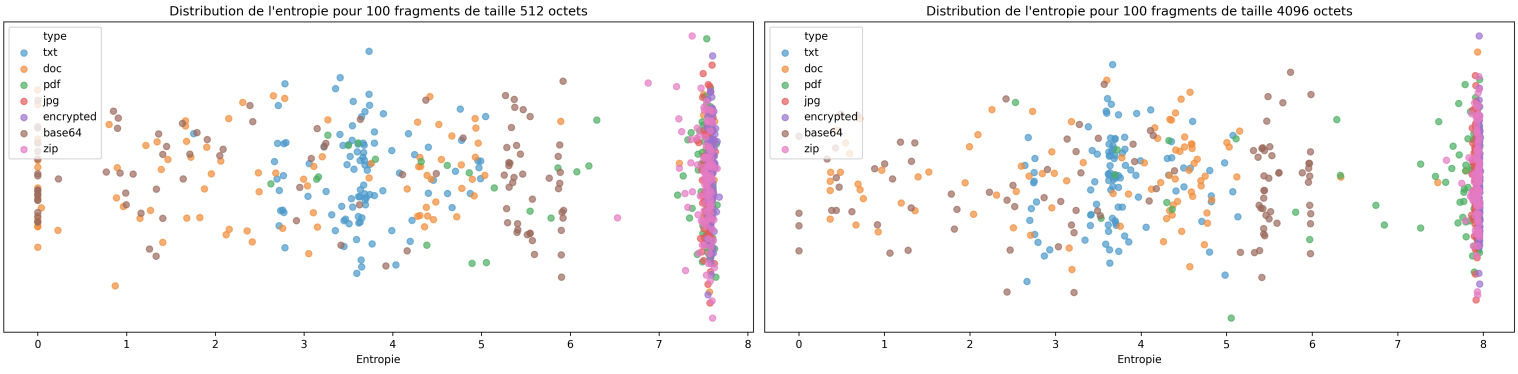

FIGURE 3 – Distribution de l'entropie de 100 fragments par type de fichier, pour des tailles de 512 et 4 096 octets.

Les fragments TXT présentent une distribution concentrée sur les caractères ASCII imprimables, notamment les espaces, les lettres minuscules, la ponctuation et les retours à la ligne. Leur entropie est faible à modérée et leur profil reste relativement stable d'un fragment à l'autre.

Les fichiers DOC occupent une position intermédiaire. Leur structure binaire associe du texte, des métadonnées et des informations de mise en forme. Ils sont donc plus structurés qu'un flux compressé, mais moins directement lisibles qu'un texte brut.

Le format Base64 se distingue par son alphabet restreint, composé principalement de lettres et de chiffres. Son entropie moyenne atteint environ 5,19 bits par octet, ce qui rend ses fragments assez

facilement reconnaissables.

Les fragments JPG présentent une distribution beaucoup plus uniforme en raison de la compression avec perte. Leur entropie moyenne est proche de 7,69 bits par octet. Le format PDF se caractérise par un profil plus hétérogène. Il alterne des zones lisibles ou semi-structurées, contenant par exemple les marqueurs `obj`, `stream` et `/Type`, avec des flux compressés plus denses.

Le PDF illustre ainsi le fait que le type global du fichier ne correspond pas toujours au type de données localement observable dans un fragment.

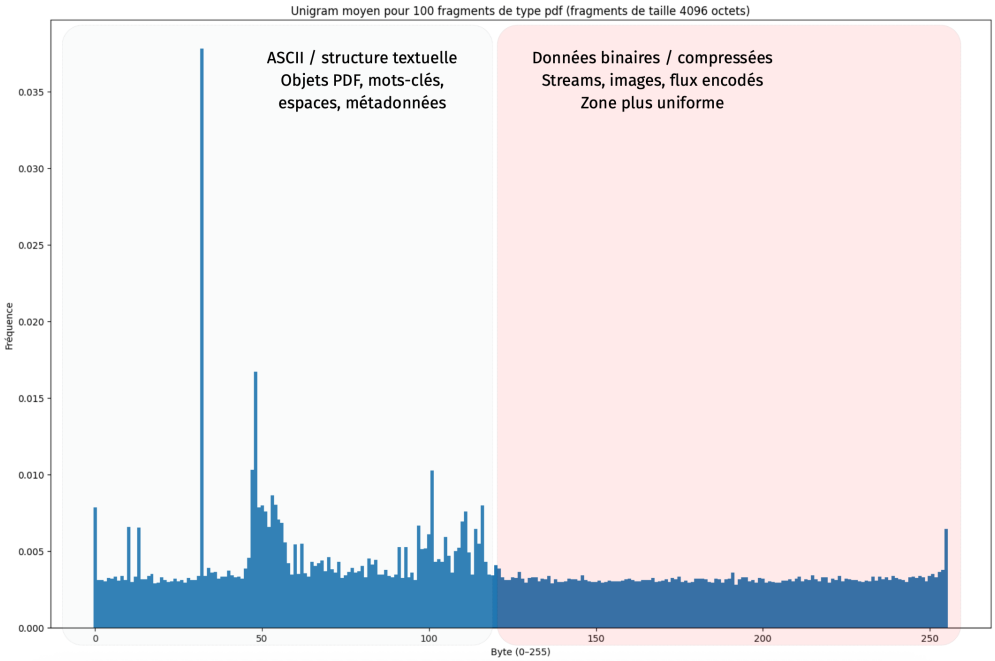

FIGURE 4 – Histogramme des unigrammes moyens des fragments PDF

Les entropies moyennes obtenues après chiffrement sont très proches du maximum théorique de 8 bits par octet, quel que soit le format source. Elles atteignent environ 7,89 bits par octet pour TXT, 7,99 pour PDF, 7,98 pour JPG et 7,99 pour DOC. Avant chiffrement, elles étaient respectivement d'environ 4,11, 7,35, 7,69 et 3,92 bits par octet.

Le chiffrement uniformise ainsi très fortement la distribution des octets. Les différences initiales entre les types de fichiers disparaissent presque entièrement. Un texte, un PDF et une image chiffrés deviennent statistiquement très proches, ce qui efface, à l'échelle du fragment, la plupart des indices permettant d'identifier le type du fichier source.

L'encodage Base64 produit un comportement différent. La distribution reste concentrée sur l'alphabet restreint utilisé par l'encodage, avec une entropie globale moyenne d'environ 5,19 bits par octet, ce qui la rend facilement reconnaissable.

La compression ZIP se rapproche au contraire du chiffrement. En réduisant la redondance du fichier source, elle uniformise la distribution des octets et produit une entropie moyenne de 7,94 bits par octet, contre 7,96 pour les fichiers chiffrés.

Ces observations montrent que certaines propriétés statistiques permettent de distinguer les formats structurés ou fondés sur un alphabet restreint. Elles deviennent cependant moins discriminantes pour les fragments compressés ou chiffrés, dont les distributions sont très proche.

Les unigrammes moyens obtenus pour chacun de ces types sont disponibles en annexe A.2.

6.3 Ces propriétés suffisent-elles à classer les fragments ?

Plusieurs classifieurs classiques ont été entraînés sur le corpus maison. Les modèles testés comprennent une forêt d'arbres de décision ou Random Forest, les k plus proches voisins, un perceptron multicouche (MLP), un Gradient Boosting et une machine à vecteurs de support (SVM). Chacun a été

évalué avec plusieurs représentations : séquence d'octets brute, unigrammes, bigrammes, trigrammes hashés et bigrammes réduits par TruncatedSVD. L'apport de l'entropie de Shannon a également été testé.

Les expériences ont été réalisées sur les quatre tailles de fragments. L'objectif était d'évaluer dans quelle mesure des représentations statistiques simples suffisent à discriminer les types de fragments.

Pour chaque configuration, les données ont été séparées en un ensemble d'entraînement de 80 % et un ensemble de test de 20 %, avec une graine fixée à 42. Le découpage a été effectué au niveau des fichiers sources. Ainsi, tous les fragments provenant d'un même fichier sont placés dans le même ensemble. Un même fichier ne peut donc pas se retrouver à la fois dans l'ensemble d'entraînement et dans l'ensemble de test.

Table 1 – Meilleure configuration observée par taille de fragment

Taille du fragment	**Meilleure configuration**	**Exactitude**
4 096 octets	Gradient Boosting – unigrammes + entropie	89,68 %
512 octets	MLP – unigrammes	81,10 %
256 octets	MLP – unigrammes	76,58 %
128 octets	SVM – trigrammes hashés + entropie	73,67 %

L'ensemble des modèles utilisés et des résultats obtenus est disponible en annexe A.3.

Les performances diminuent lorsque la taille du fragment se réduit. Elles passent de 89,68 % à 4 096 octets à 73,67 % à 128 octets. Ce résultat confirme que moins un fragment est long, moins il contient d'information exploitable.

Les représentations statistiques simples se montrent déjà compétitives. Les unigrammes donnent de bons résultats sur plusieurs tailles, notamment avec le MLP, ce qui montre que la distribution globale des octets porte une part importante du signal discriminant. L'entropie seule ne suffit cependant pas à séparer les classes. Deux fragments peuvent avoir une entropie proche tout en appartenant à des types différents.

Les classes TXT, DOC et Base64 ressortent systématiquement comme les plus faciles à reconnaître, grâce à des distributions structurées et stables. Les classes chiffrées, compressées et certaines images compressées restent plus difficiles.

En complément, un premier CNN 1D a été testé directement sur les séquences d'octets brutes. L'objectif était de vérifier si un réseau pouvait apprendre automatiquement des motifs locaux sans représentation statistique explicite. Le taux de classification obtenu est proche de 81 %. Le modèle exploite donc bien un signal, mais sans dépasser clairement les méthodes statistiques et pour un coût d'entraînement plus élevé. La limite apparaît surtout sur les fragments chiffrés ou compressés, dont les distributions sont presque uniformes.

6.4 Bilan

Cette expérience montre que les caractéristiques statistiques simples permettent de distinguer une partie des formats du corpus maison. Cependant, leur efficacité dépend fortement de la taille du fragment et du type de contenu. Les formats structurés restent relativement reconnaissables, tandis que le chiffrement et la compression réduisent fortement les différences observables entre les classes.

Ces premiers résultats ont conduit à poursuivre l'étude sur FFT-75, un corpus plus large et plus diversifié, afin de vérifier si les mêmes difficultés apparaissent lorsque le nombre de types de fichiers augmente.

7 Classification de fragments sur FFT-75

Après les expériences menées sur le corpus maison, le travail s'est déplacé vers FFT-75 [9], un jeu de données de référence plus difficile et comparable à ceux utilisés dans la littérature. FFT-75 impose

un changement d'échelle, avec 75 classes, plusieurs familles de formats et des confusions beaucoup plus proches de celles rencontrées dans les travaux récents.

Ces nouvelles expériences cherchent à répondre à de nouvelles questions. Les observations réalisées sur le corpus maison se retrouvent-elles sur un jeu de données plus large ? Quelle représentation est la plus adaptée à la classification de fragments courts ? Enfin, une décomposition hiérarchique du problème permet-elle de mieux traiter les groupes de formats présentant des niveaux de difficulté différents ?

7.1 Présentation et analyse du jeu de données

Plusieurs raisons ont motivé le choix de FFT-75. La première est la diversité des classes. Avec les 75 types de fichiers couverts par le jeu de données, il inclut des formats textuels, des images ou encore des exécutables. La deuxième est l'équilibre du jeu de données, puisque chaque type de fichier est représenté par un nombre comparable de fragments. Enfin, la plupart des travaux récents en classification de fragments de fichiers rapportent leurs résultats sur FFT-75, en particulier sur le scénario #1. Travailler sur ce jeu de données permet donc de situer les résultats obtenus par rapport aux méthodes existantes.

FFT-75 est organisé en plusieurs scénarios à granularité variable. Dans le scénario #1, chaque type de fichier est considéré comme une classe distincte. Le modèle doit donc prédire directement parmi les 75 types. Des scénarios plus simples regroupent certains types en familles et peuvent produire des taux d'exactitude plus élevés, mais ils ne répondent pas au même problème. Le scénario #1 teste directement la capacité d'un modèle à identifier le type du fichier source à partir d'un fragment isolé.

La répartition complète des 75 classes par grande catégorie est présentée en annexe A.4.

Parmi les deux tailles proposées par FFT-75, les fragments de 512 octets ont été utilisés pour la suite du travail. Il s'agit du cas dans lequel la quantité d'information disponible est la plus faible. Ces fragments peuvent aussi correspondre à des zones qui ne sont pas représentatives du type global du fichier source, par exemple un flux compressé, une image intégrée ou une zone textuelle.

FFT-75 présente cependant certaines limites. Les fragments sont considérés indépendamment les uns des autres. Le modèle ne dispose ni des secteurs voisins, ni de l'ordre des fragments, ni d'information sur leur position dans le fichier source.

Le label correspond par ailleurs au type du fichier source. Or, comme le montre déjà le corpus maison, un fragment ne contient pas forcément des données représentatives du type global du fichier. Un fragment issu d'un format conteneur peut ainsi ressembler localement à un autre format.

Enfin, les performances d'un modèle ne peuvent pas être interprétées indépendamment des données sur lesquelles elles sont mesurées. Avant d'attribuer une erreur à l'architecture, à la représentation ou à l'entraînement, il faut donc comprendre ce que contient réellement le jeu de fragments. L'unicité des fragments, la présence de doublons et les cas où un même contenu binaire apparaît avec plusieurs labels ont été analysés.

Chaque fragment est considéré comme une séquence exacte de 512 octets. Deux fragments sont donc identiques si leurs 512 octets sont strictement les mêmes. À l'échelle du jeu complet, on observe environ 111 317 doublons intra-classe. Les résultats détaillés pour les classes présentant les plus faibles taux d'unicité sont présentés en annexe A.6.

Leur présence n'est pas nécessairement problématique et peut refléter des zones structurelles communes, comme des motifs de remplissage ou d'alignement.

Les doublons inter-classes sont plus problématiques, puisqu'ils correspondent à des fragments strictement identiques apparaissant avec plusieurs labels. Dans ce cas, le contenu binaire seul ne permet pas de déterminer avec certitude la classe correcte.

Certains doublons correspondent également à des fragments constitués d'un seul octet répété, principalement `0x00` ou `0xFF`. Ils restent peu nombreux. Leur retrait ne concerne que 0,69 % des fragments du test et améliore faiblement l'exactitude du MLP statistique présenté ensuite, de 65,68 % à 65,90 %, soit un gain de 0,22 point. Ils peuvent cependant avoir davantage d'effet sur certaines classes, notamment MACH-O, ELF et DLL.

Ces observations montrent que certains fragments ne sont pas simplement difficiles à classer, mais sont peu, voire pas informatifs. Le fragment entièrement composé de `0x00` l'illustre. Ses occurrences se répartissent notamment entre DLL à 47 %, MACH-O à 37 % et ELF à 15 %. Dans ce cas, un label unique ne reflète pas nécessairement l'information réellement contenue dans le fragment.

7.2 Protocole expérimental et métriques

Le scénario #1 de FFT-75 à 512 octets comporte 75 classes. Le jeu de données d'origine contient 102 400 fragments par type, soit un corpus équilibré.

Le découpage suit le protocole défini dans FiFTy. Il attribue 80 % des fragments à l'entraînement, 10 % à la validation et 10 % au test.

L'évaluation des performances ne se limite pas au taux de classification globale. Celui-ci correspond à la proportion totale de fragments correctement classés. Il donne une première mesure de performance, mais masque les différences entre les classes.

Le rappel permet d'observer séparément la proportion de prédictions correctes pour chaque type de fichier. La précision indique, parmi les fragments prédits dans une classe, la proportion qui appartient réellement à cette classe. Enfin, les matrices de confusion indiquent vers quelles classes sont dirigées les erreurs.

Ces mesures sont importantes, car FFT-75 contient des formats très hétérogènes. Un taux de classification globale correct peut masquer des performances très faibles sur certaines classes et, à l'inverse, des classes très faciles à reconnaître.

7.3 Méthodes évaluées

7.3.1 Méthodes de référence

Trois méthodes de référence sont testées afin de mesurer la difficulté du problème avant d'introduire des architectures plus complexes.

La première repose sur un MLP entraîné directement sur la séquence des 512 octets, sans extraction de caractéristiques. La deuxième utilise un CNN simple sur les fragments bruts afin d'exploiter les motifs locaux présents dans la séquence d'octets.

La troisième méthode associe un MLP à une représentation statistique. Le modèle reçoit une description synthétique, un peu comme une signature, du fragment composée de l'histogramme des unigrammes, de l'entropie et de la variance. Des bigrammes hashés ont également été testés. Cette approche reprend les observations du corpus maison, où la distribution des octets s'était montrée discriminante.

7.3.2 Modèle global multibranche

Après ces méthodes de référence, un modèle plus complet est évalué sur le même scénario. Contrairement aux méthodes précédentes, qui reposent soit sur les octets bruts, soit sur des statistiques calculées à la main, ce modèle cherche à combiner plusieurs vues complémentaires du même fragment.

L'hypothèse est qu'un fragment peut contenir plusieurs types d'indices. Certains sont visibles au niveau de l'octet, par exemple des caractères ASCII. D'autres peuvent se situer au niveau du bit, notamment dans les formats compressés. Enfin, certaines propriétés globales, comme la distribution des octets ou l'entropie, peuvent aider à caractériser le type de contenu.

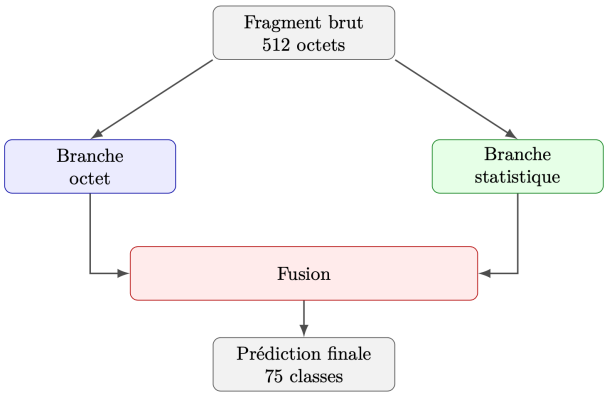

FIGURE 5 – Architecture dual-stream

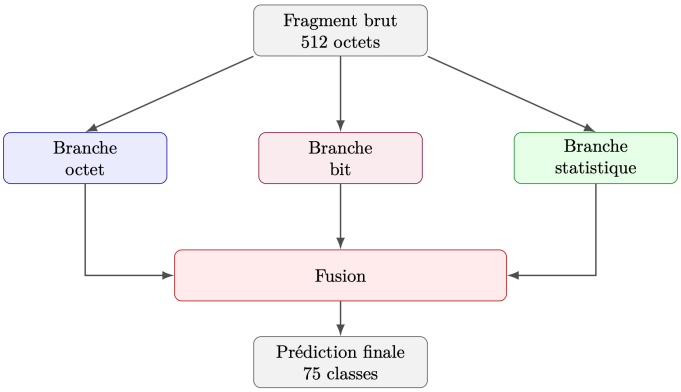

FIGURE 6 – Architecture tri-stream

La première branche est une branche octet. Elle reçoit la séquence des 512 octets, ensuite traitée par un réseau temporel convolutif, ou TCN. Cette branche vise à apprendre des motifs locaux et semi-locaux dans la séquence d'octets.

La deuxième est une branche bit. Elle transforme chaque octet en une séquence de bits, ce qui produit une représentation de 4 096 bits pour un fragment de 512 octets. L'objectif est de capturer une information intra-octet potentiellement perdue lorsque l'octet est traité comme une unité à part entière. Cette branche est inspirée des approches ByteNet et Byte2Image présentées dans l'état de l'art.

La troisième est une branche statistique. Elle reçoit un vecteur de dimension 260, composé des 256 fréquences d'octets, de l'entropie de Shannon, de la variance de l'entropie locale, de la moyenne et de l'écart-type des octets.

Les trois branches sont fusionnées par un mécanisme de *gating softmax* à trois voies. Ce mécanisme apprend à pondérer les branches selon le fragment. Une variante dual-stream a aussi été testée. Elle conserve uniquement les branches octet et statistique.

Une étude d'ablation est menée afin de mesurer séparément l'apport de chaque branche.

7.3.3 Approche hiérarchique

Demander à un seul modèle de séparer simultanément 75 classes revient à mélanger des problèmes de natures très différentes. Certaines classes sont facilement distinguables, alors que d'autres sont beaucoup plus difficiles. Les formats ne partagent pas tous la même structure et les confusions concernent surtout certains groupes, notamment les exécutables, les formats de présentation, les archives et les conteneurs documentaires.

L'hypothèse est alors de séparer d'abord les fragments selon des propriétés globales observables, puis d'orienter chaque groupe vers un modèle expert spécialisé. Le problème principal vient du routeur. Si un fragment est envoyé vers le mauvais expert, l'erreur devient difficile à corriger. La hiérarchie n'est donc pertinente que si le découpage initial est suffisamment fiable.

Plusieurs approches ont été testées pour découper les familles. Celle retenue repose sur trois bandes d'entropie, nommées LowEntropy, MediumEntropy et HighEntropy. Ce découpage vient des observations du corpus maison et des premières analyses sur FFT-75, où l'entropie apparaît comme un indicateur de difficulté. Les seuils de 5 et 7 ont été choisis en observant la répartition des entropies moyennes de chaque classe. La répartition complète des classes entre les trois bandes d'entropie est détaillée en annexe A.5.

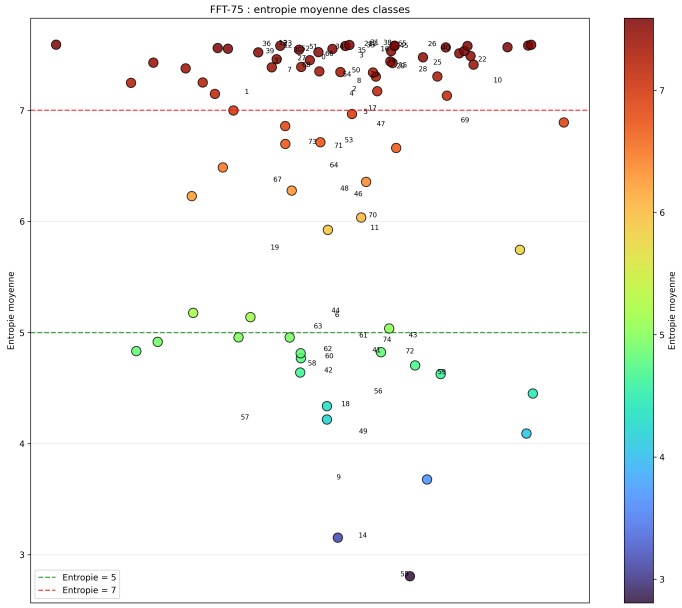

FIGURE 7 – Entropie moyenne des différentes classes de FFT-75

Le groupe LowEntropy contient 17 classes. Elles partagent une entropie moyenne faible et correspondent principalement à des formats textuels ou structurés, avec des motifs facilement exploitables au niveau de l'octet. Les modèles testés comprennent des MLP statistiques et des CNN entraînés sur les octets bruts.

Le groupe MediumEntropy contient également 17 classes. Il se situe entre deux régimes. Les fragments ne sont plus aussi structurés que ceux de LowEntropy, mais ne sont pas tous assimilables à du bruit. Des représentations locales plus riches sont donc évaluées avec des MLP et des CNN au niveau de l'octet ou du bit.

Le groupe HighEntropy rassemble les 41 classes restantes et constitue le cœur du problème. Ses fragments présentent des distributions proches du bruit, avec des entropies élevées et peu de redondance exploitable. Des CNN, un MLP utilisant des bigrammes et un TCN sont comparés sur ce groupe.

Deux conditions de routage sont testées. La première repose sur un routage oracle. Chaque fragment est envoyé vers l'expert correspondant à sa vraie famille d'entropie. Cette évaluation ne correspond pas à un système directement utilisable, mais permet de mesurer le potentiel maximal de la combinaison d'experts lorsque l'erreur de routage est neutralisée.

Dans la seconde condition, un routeur automatique prédit la famille d'entropie à partir de caractéristiques statistiques extraites des 512 octets. Ces caractéristiques comprennent l'histogramme des octets, l'entropie globale et des entropies calculées par blocs.

7.4 Résultats

7.4.1 Résultats des méthodes de référence

TABLE 2 – Résultats des méthodes de référence sur FFT-75, scénario #1, 512 octets

Modèle et représentation	**Exactitude**
Hasard sur 75 classes	1,33 %
MLP sur octets bruts	36,79 %
CNN sur octets bruts	54,13 %
MLP sur représentation statistique	**65,68 %**

Le MLP entraîné directement sur les 512 octets atteint 36,79 %, un résultat bien supérieur à une prédiction au hasard parmi 75 classes qui obtiendrait 1,33 %. Les octets contiennent donc déjà une information exploitable.

Le CNN atteint 54,13 % d'exactitude, soit une amélioration par rapport au MLP brut. Le MLP sur représentations statistiques obtient 65,68 %, ce qui montre que ces caractéristiques restent très compétitives. Ce résultat est déjà comparable aux 65,60 % rapportés par Mittal et al. pour FiFTy sur le même scénario et avec des fragments de même taille.

Ces résultats confirment sur FFT-75 les observations du corpus maison. Les fragments structurés sont plus facilement discriminables que les fragments compressés ou à haute entropie.

7.4.2 Résultats du modèle global multibranche

Table 3 – Exactitude selon la configuration sur FFT-75, scénario #1, 512 octets

Configuration	**Exactitude**
Branche octet seule	70,89 %
Branche bit seule	67,03 %
Branche statistique seule	58,98 %
Dual-stream octet + statistique	71,00 %
Tri-stream octet + bit + statistique	**71,12 %**

Le modèle dual-stream atteint 71,00 % d'exactitude et le modèle tri-stream atteint 71,12 %. L'étude d'ablation permet de situer ce résultat. Le tri-stream ne gagne que 0,23 point par rapport à la branche octet seule.

Les poids moyens le confirment. La branche octet domine largement la décision, tandis que les branches bit et statistique restent secondaires. La branche bit reçoit parfois un poids légèrement plus élevé sur certains formats compressés ou certaines archives.

Replacée dans l'état de l'art, l'exactitude de 71,12 % obtenue par le modèle de bout en bout dépasse FiFTy dans des conditions comparables, mais reste légèrement inférieure à celles de Carve-Former et ByteFormer qui obtenaient respectivement 72,10 % et 73,2 % d'exactitude.

Ces résultats montrent que la branche octet porte l'essentiel de la performance. Le niveau bit, pourtant théoriquement intéressant, apporte moins que prévu dans une fusion globale. La branche statistique seule reste limitée, même si elle fournit un signal utile lorsqu'elle est combinée à la branche octet.

7.4.3 Résultats de l'approche hiérarchique

Table 4 – Résultats de l'expert LowEntropy

Modèle	**Représentation**	**Exactitude**
MLP statistique	1g + entropie + variance	98,04 %
MLP enrichi	1g + entropie + variance + 2g hashés	**98,72 %**
CNN sans lissage des labels	octets bruts	98,11 %
CNN avec lissage des labels à 0,03	octets bruts	98,15 %
MLP avec lissage des labels à 0,03	1g + entropie + variance + 2g hashés	98,71 %

Expert LowEntropy. Les scores dépassent 98 % et les erreurs restantes sont localisées, notamment entre MACH-O, ELF et DLL, ainsi qu'entre certains formats textuels proches. Le groupe LowEntropy est donc largement résolu.

TABLE 5 – Résultats de l'expert MediumEntropy

Modèle	Représentation	Exactitude
MLP au niveau octet	représentation simple des octets	83,14 %
CNN au niveau octet	octets bruts	83,61 %
MLP au niveau bit	bits	82,97 %
MLP enrichi	1g + 2g hashés	86,51 %
CNN au niveau bit	bits	**87,95 %**

Expert MediumEntropy. L'ajout des bigrammes améliore le MLP à 86,51 %, tandis que le CNN au niveau bit obtient le meilleur score avec 87,95 %. Les confusions se concentrent principalement autour de AI, PCAP, PPT, PPTX, KEY et MOBI, des formats qui peuvent partager des zones structurelles ou compressées. Le groupe MediumEntropy est donc déjà affecté par le problème des formats composites.

TABLE 6 – Résultats de l'expert HighEntropy

Modèle	Représentation	Exactitude
CNN simple	niveau octet	45,75 %
MLP à bigrammes	1g + 2g hashés	48,69 %
CNN au niveau bit	niveau bit	51,21 %
TCN au niveau octet	séquence d'octets	**55,56 %**

Expert HighEntropy. Le TCN au niveau octet obtient le meilleur score avec 55,56 %, mais reste loin des performances de LowEntropy et MediumEntropy. HighEntropy reste donc le principal problème de cette décomposition.

Routage oracle et routage automatique. Dans la condition oracle, les experts obtiennent 98,74 % pour LowEntropy, 85,94 % pour MediumEntropy et 51,81 % pour HighEntropy. En pondérant ces scores par le nombre de fragments de chaque famille, l'exactitude globale oracle atteint 70,16 %. Même avec un routage parfait, cette configuration n'améliore donc pas le modèle global tri-stream à 71,12 %. La principale limite vient de l'expert HighEntropy.

Sur le jeu de test, le routeur automatique atteint 93,62 % d'exactitude. Les familles LowEntropy et HighEntropy sont correctement identifiées, avec des rappels respectifs de 98,84 % et 97,05 %. MediumEntropy est plus difficile à router, avec un rappel de 80,14 %. Une bonne partie de ses fragments est redirigée vers HighEntropy.

Dans un système non-oracle, une erreur de routage est problématique. Si un fragment est envoyé vers un expert qui ne contient pas sa vraie classe, la prédiction finale ne peut généralement plus être correcte. La performance dépend donc à la fois du routeur et des experts spécialisés.

Bien que ce chaînage complet n'ait pas été exécuté directement dans le temps imparti, il est possible d'en estimer la performance en supposant qu'un fragment mal routé entraîne systématiquement une prédiction erronée. L'exactitude globale estimée peut alors être modélisée par la somme pondérée des performances conditionnelles de chaque groupe :

$$\mathrm{Acc}_{\text{estimée}} = \sum_{g \in \{\mathrm{Low},\,\mathrm{Med},\,\mathrm{High}\}} w_g \times \mathrm{Rappel}_{\text{routeur}}(g) \times \mathrm{Acc}_{\text{expert}}(g). \tag{1}$$

Dans cette expression, w_g représente le poids de chaque groupe dans le jeu de test. FFT-75 étant quasiment équilibré entre les classes, ces poids sont approximés par la proportion de classes appartenant à chaque groupe. On utilise ainsi $w_{\mathrm{Low}} = 17/75$, $w_{\mathrm{Med}} = 17/75$ et $w_{\mathrm{High}} = 41/75$. En combinant les rappels du routeur avec les scores retenus pour les experts, on obtient :

$$\begin{aligned}\mathrm{Acc}_{\text{estimée}} \approx &\left(\frac{17}{75} \times 0{,}9884 \times 0{,}9872\right) + \left(\frac{17}{75} \times 0{,}8014 \times 0{,}8651\right) \\ &+ \left(\frac{41}{75} \times 0{,}9705 \times 0{,}5556\right) \approx 67{,}32\,\%.\end{aligned} \tag{2}$$

On observe ainsi la baisse de performance induite par la cascade d'erreurs. L'exactitude estimée atteint 67,32 %, contre 70,16 % sous routage oracle et 71,12 % pour le modèle global tri-stream.

Ces résultats montrent que la décomposition par entropie permet d'isoler des sous-problèmes de difficulté très différente, mais qu'elle ne suffit pas, dans sa forme actuelle, à améliorer la performance globale.

7.5 Analyse des erreurs et des confusions

L'objectif de cette partie est de qualifier les erreurs plutôt que de seulement les quantifier, afin de distinguer ce qui relève du modèle, de la représentation ou de l'information réellement disponible dans le fragment.

La matrice de confusion permet d'identifier les couples de classes les plus souvent confondus, les erreurs les plus fréquentes et les classes « attractives », vers lesquelles tendent plusieurs fragments issus d'autres types. Des matrices de confusion dynamiques ont ainsi été développées. Un clic sur une cellule permet d'afficher les fragments concernés avec plusieurs vues complémentaires.

FIGURE 8 – Exemple d'une matrice de confusion dynamique

Cette analyse ne détaille pas de manière exhaustive l'ensemble des foyers d'erreur observés. Elle se concentre sur quelques groupes représentatifs afin d'illustrer la nature des principales confusions.

Foyer LowEntropy : MACH-O, ELF et DLL

Ces trois formats correspondent à des exécutables ou à des bibliothèques binaires. Ils partagent donc une logique structurelle comprenant des sections de code, des zones de données et des tables de chargement.

Leurs entropies moyennes sont également proches. Elles atteignent environ 4,82 pour MACH-O, 4,64 pour ELF et 4,96 pour DLL, ce qui reflète leur proximité réelle. Le contrôle précédent écarte l'hypothèse d'une fuite massive de données, puisque l'unicité atteint 97,31 % pour MACH-O, 96,02 % pour ELF et 96,90 % pour DLL.

En revanche, les fragments triviaux peuvent jouer un rôle plus important. Les zones entièrement nulles, fréquentes dans les exécutables, peuvent apparaître dans plusieurs classes.

Un classifieur entraîné uniquement sur ces trois classes atteint environ 96,21 % avec un MLP et 96,76 % avec un CNN au niveau octet. Ces résultats montrent que les formats sont séparables, mais pas parfaitement distincts. Les confusions semblent donc refléter une proximité structurelle réelle.

Foyer MediumEntropy : AI, PCAP, PPT, PPTX, KEY et MOBI

La classe AI se comporte comme une classe « puits », mêlée à PCAP, PPT, PPTX, KEY et MOBI. Ces confusions peuvent venir de leur structure. AI, proche de PostScript et de PDF, peut présenter des sections très hétérogènes. KEY, PPTX et PPT sont liés à des formats de présentation qui peuvent contenir des objets compressés ou des structures XML.

Les erreurs ne se répartissent donc pas au hasard entre les 75 classes. Elles concernent surtout certains groupes de formats proches, notamment les archives, les exécutables ou les formats textuels proches. L'analyse de HighEntropy montre également que toutes les classes difficiles ne doivent pas être traitées de la même manière. Ce qui compte n'est pas seulement le niveau d'entropie, mais aussi la présence ou non de régularités exploitables dans le fragment.

7.6 Bilan

Les expériences menées sur FFT-75 confirment d'abord une partie des observations réalisées sur le corpus maison. Les caractéristiques statistiques restent efficaces, puisque le MLP statistique atteint 65,68 %. Les fragments structurés sont plus faciles à distinguer que les fragments compressés ou à haute entropie.

La comparaison des représentations montre ensuite que le niveau octet reste le plus efficace dans les configurations testées. La branche octet atteint à elle seule 70,89 %, tandis que le modèle tri-stream obtient 71,12 %. L'ajout des branches bit et statistique apporte donc peu dans une fusion globale.

Enfin, la décomposition hiérarchique permet d'isoler des groupes de difficulté très différente. LowEntropy dépasse 98 %, alors que HighEntropy reste proche de 55 % avec le meilleur modèle testé. Cette séparation aide donc à localiser la difficulté, mais elle n'améliore pas la performance globale. Même avec un routage oracle, l'exactitude atteint 70,16 %, contre 71,12 % pour le modèle global. L'erreur du routeur réduit encore cette performance dans l'estimation non-oracle.

Ces résultats répondent aux questions posées au début de cette section. Les observations du corpus maison se retrouvent en partie sur FFT-75, le niveau octet apporte le plus d'information dans les expériences menées, et la hiérarchie permet surtout d'identifier les groupes difficiles sans améliorer, dans sa forme actuelle, la classification globale.

8 Analyse des profils internes des fragments PDF

Le travail précédent sur le corpus a conduit à approfondir le cas des formats composites, en particulier PDF et DJVU. Ces deux formats documentaires figurent parmi les classes difficiles du modèle. Ils peuvent contenir plusieurs natures de données, notamment des éléments de structure lisibles, des objets internes et des flux compressés.

Cette partie cherche à déterminer si les performances de classification des fragments PDF dépendent de leur profil interne. L'objectif est d'abord de caractériser de manière empirique le vocabulaire visible du format, indépendamment de FFT-75. Les fragments PDF de FFT-75 sont ensuite analysés à partir de profils internes, afin de comprendre lesquels sont réellement discriminables et lesquels restent difficiles à distinguer à partir de leur seul contenu.

La même démarche a été appliquée au format DJVU, mais seuls les résultats obtenus pour PDF sont détaillés dans cette partie. Le vocabulaire structurel obtenu pour le format DJVU est présenté en annexe A.8.

8.1 Méthode

8.1.1 Construction d'un lexique empirique

Pour caractériser le format PDF, un corpus externe de fichiers PDF a été constitué afin d'en extraire des indices sur la structure du format. Chaque fichier est lu en binaire, puis les chaînes ASCII lisibles sont extraites à l'aide d'expressions régulières.

Pour chaque token, deux mesures sont calculées. La première est sa fréquence totale dans le corpus. La seconde est sa fréquence documentaire, c'est-à-dire le nombre de fichiers dans lesquels il apparaît. Cette deuxième mesure est importante, car un token très fréquent dans un seul fichier n'a pas la même portée qu'un token présent dans une grande partie du corpus.

8.1.2 Descripteurs des fragments

Une fois le vocabulaire constitué, les fragments PDF de FFT-75 sont décrits à l'aide d'un ensemble de descripteurs combinant structure globale, lisibilité et hétérogénéité locale. Les fragments DJVU sont décrits selon la même grille.

TABLE 7 – Descripteurs utilisés pour caractériser les fragments PDF et DJVU

Descripteur	**Rôle dans l'analyse**
`entropy`	Mesure le désordre global du fragment.
`printable_ratio`	Proportion d'octets ASCII imprimables.
`longest_ascii_run`	Plus longue séquence ASCII lisible continue.
`{pdf/djvu}_token_count`	Nombre total de tokens du format détectés.
`{pdf/djvu}_token_presence`	Présence ou absence de vocabulaire du format.
`{pdf/djvu}_token_unique_count`	Nombre de tokens distincts détectés.
`slash_count` et `separator_count`	Densité des marqueurs syntaxiques comme `/`, `[`, `]`, « et ».
`block_entropy_std` et `block_printable_std`	Hétérogénéité locale calculée par blocs.
`zero_ratio` et `byte_std`	Proportion d'octets nuls et dispersion des valeurs.

8.1.3 Identification des profils internes

Pour vérifier si des profils émergent des données elles-mêmes, sans imposer a priori des catégories comme *compressed-like* ou *structural-PDF-like*, un modèle de mélange gaussien, ou GMM, est appliqué aux fragments PDF de l'ensemble d'entraînement.

Le GMM cherche à regrouper les fragments présentant des profils statistiques proches. Chaque groupe est modélisé par une distribution gaussienne et aucun label de profil n'est fourni au modèle. Le nombre de composantes a été fixé empiriquement à quatre afin d'obtenir un découpage suffisamment détaillé tout en conservant des groupes que l'on peut interpréter.

Les clusters obtenus sont ensuite interprétés à partir des valeurs moyennes de leurs descripteurs. Le modèle ajusté sur l'ensemble d'entraînement est enfin appliqué aux fragments de test afin de leur attribuer un profil.

Pour mesurer le lien entre ces profils et la classification, le rappel de la classe PDF obtenu par le modèle tri-stream est calculé séparément dans chaque cluster. Il correspond à la proportion de fragments PDF du cluster qui sont correctement reconnus comme PDF.

8.2 Résultats

8.2.1 Vocabulaire structurel du format PDF

Table 8 – 10 tokens les plus fréquents du vocabulaire structurel PDF

Rang	Token	Fréquence totale	Fréquence documentaire
1	`«`	2 951 793	1 869
2	`»`	2 951 638	1 869
3	`obj`	2 419 412	1 869
4	`endobj`	2 419 412	1 869
5	`[`	2 045 527	1 869
6	`]`	2 045 487	1 869
7	`/Type`	1 446 387	1 869
8	`/Subtype`	594 772	1 869
9	`endstream`	488 417	1 869
10	`stream`	488 313	1 869

Le classement étendu aux vingt tokens les plus fréquents est présenté en annexe A.7.

Le vocabulaire fait ressortir des tokens directement liés à l'organisation interne du PDF. On retrouve notamment `obj`, `endobj`, `stream`, `endstream`, `xref`, `trailer`, `/Type`, `/Length`, `/Filter` et des noms de ressources.

Cela confirme qu'un fragment PDF n'est pas nécessairement « du PDF visible ». Certains fragments contiennent des tokens caractéristiques et peuvent être interprétés comme structurels. D'autres ne contiennent aucun élément propre au PDF et correspondent plutôt à des flux compressés, des images ou des zones binaires.

8.2.2 Profils de fragments identifiés

Table 9 – Distribution et interprétation des clusters PDF

Cluster	Interprétation	Train	Test	Rappel PDF du tri-stream
0	Haute entropie sans marqueur PDF	58,70 %	59,47 %	24,04 %
1	Haute entropie, marqueurs PDF faibles	29,95 %	29,77 %	24,64 %
2	Structure PDF dense et lisible	4,37 %	4,33 %	97,29 %
3	Structure mixte ou transition	6,98 %	6,44 %	71,47 %

Le cluster 0 domine. Il présente une entropie très élevée, un faible ratio d'octets imprimables et aucun token PDF détecté. Il correspond donc principalement à du contenu compressé ou binaire.

Le cluster 1 lui est proche en entropie, mais contient quelques tokens PDF, avec une moyenne de 1,767 token par fragment, sans structure syntaxique importante.

Le cluster 2 est très lisible, avec un ratio d'octets imprimables égal à 1, une longue séquence ASCII ainsi que de nombreux tokens et séparateurs. Il s'agit du groupe dans lequel la signature PDF est la plus explicite.

Le cluster 3 est intermédiaire, avec une hétérogénéité locale plus marquée qui pourrait suggérer une alternance entre des zones structurelles et des zones moins lisibles.

8.2.3 Performances selon le profil

Les performances de classification diffèrent selon les clusters. Les clusters 0 et 1, dominés par la haute entropie, sont les moins bien reconnus, avec des rappels respectifs de 24,04 % et 24,64 %. Les clusters 2 et 3, plus structurés, atteignent respectivement 97,29 % et 71,47 %.

Table 10 – Regroupement des clusters PDF en deux régimes empiriques

Régime empirique	**Clusters**	**Fragments de test**	**Rappel de la classe PDF**
Haute entropie et faible structure exploitable	0 et 1	9 137	24,24 %
Structure PDF dense ou mixte	2 et 3	1 102	81,85 %

Les clusters 0 et 1, bien que très entropiques, avec des entropies moyennes respectives de 7,546 et 7,559 bits par octet, produisent tout de même 2 215 prédictions correctes parmi leurs 9 137 fragments de test. Le modèle exploite donc probablement des signaux faibles qui ne sont pas entièrement capturés par les descripteurs interprétables.

Ces résultats montrent que la haute entropie ne rend pas systématiquement un fragment impossible à classer. Le modèle classe encore correctement une partie des fragments à haute entropie, mais avec une fiabilité bien plus faible que sur les fragments structurés.

La classe PDF est donc hétérogène à l'échelle du fragment. Une petite partie présente une structure dense et très discriminante. Une autre partie, plus mixte, conserve assez d'indices pour être relativement bien reconnue. La majorité des fragments appartient cependant à des régimes de haute entropie dans lesquels la signature PDF est absente ou trop faible.

Sur l'ensemble des fragments PDF du test, le modèle ne classe correctement qu'environ 30 % des fragments. Les erreurs se concentrent vers des formats eux-mêmes compressés ou conteneurs, notamment BZ2, RAR, 7Z, HEIC et XZ. Ces formats partagent avec les fragments PDF concernés une forte entropie ou une faible lisibilité.

8.3 Bilan et transition vers la reconstruction

Ce que l'on peut retenir de ces expériences est que le taux de classification par classe est insuffisant pour comprendre les formats composites. Il semble plus intéressant d'analyser les fragments à partir de leurs profils internes. Cette analyse permet de mieux distinguer les erreurs dues à une proximité entre formats documentaires, celles qui sont liées aux flux compressés et les limites informationnelles du fragment.

L'étude des PDF et DJVU permet ainsi de mieux comprendre l'origine des confusions observées lors de la classification. Ces formats composites alternent des fragments très différents, avec des éléments de structure lisibles, des métadonnées, des flux compressés ou des objets binaires. Le lexique empirique et les descripteurs statistiques montrent que le label du fichier source ne résume pas à lui seul la nature d'un bloc de 512 octets. C'est plutôt le profil interne du fragment qui détermine à quel point il peut être correctement classé.

Si certains fragments contiennent des indices structurels clairs, les zones compressées restent très ambiguës à l'échelle d'un bloc isolé. La suite du travail consiste donc à ne plus considérer uniquement les fragments de manière isolée, mais à étudier comment ces différents profils s'organisent au sein d'un document et comment leurs relations peuvent être exploitées pour tenter de le reconstruire.

9 Reconstruction de documents PDF fragmentés

Dans la continuité de l'analyse des formats composites, la suite du travail a consisté à étudier la reconstruction de documents PDF à partir de fragments isolés. L'objectif n'est plus de classer chaque fragment selon son type de fichier source, mais de retrouver les relations entre les fragments afin de contribuer à leur réassemblage. On parlera dans la suite de « puzzles PDF ».

Cette partie cherche à répondre à plusieurs questions. Les fragments voisins présentent-ils un signal d'adjacence mesurable ? Les informations situées à la frontière sont-elles plus utiles que les caractéristiques globales des fragments ? Les modèles entraînés sur un document peuvent-ils être généralisés à d'autres PDF ? Enfin, les informations structurelles propres au format PDF peuvent-elles compléter les approches d'apprentissage ?

9.1 Corpus et caractérisation des fragments

9.1.1 Construction du corpus

Cette étude a nécessité la constitution d'un corpus spécifique de documents PDF qui peuvent être découpés en pièces afin d'étudier leur réassemblage. Le corpus a été construit à partir de deux sources, arXiv et HAL. Cent documents ont été sélectionnés sur chacune des plateformes, puis téléchargés à l'aide de leurs API. Au total, 194 documents PDF ont été retenus, dont 98 provenant d'arXiv et 96 de HAL. Ils représentent environ 748 Mo de données.

Les documents sont découpés séquentiellement en fragments fixes de 512 octets. Le dernier fragment est complété par des octets nuls lorsqu'il est plus court. Cette étape produit un corpus de 1 532 362 fragments, soit une moyenne de 7 899 fragments par document. Chaque document constitue un puzzle indépendant dont l'ordre correct des pièces est connu.

9.1.2 Caractérisation et profils des fragments

Chaque fragment est annoté à l'aide du vocabulaire PDF créé précédemment, limité aux tokens les plus représentatifs. Des marqueurs spécifiques sont également recherchés, notamment `stream`, `endstream`, `obj`, `endobj`, `startxref` et `%%EOF`.

Les couples `stream`/`endstream` sont aussi localisés dans le document d'origine afin de distinguer les fragments situés à l'intérieur et à l'extérieur des flux compressés. L'analyse fait apparaître 53 645 flux compressés, soit une moyenne de 276,5 par document. Au total, 95,10 % des fragments sont situés à l'intérieur d'un flux compressé, contre 4,90 % à l'extérieur.

Plusieurs caractéristiques sont calculées pour chaque fragment. Elles comprennent l'entropie de Shannon, les ratios d'octets imprimables et non imprimables, la longueur de la plus longue séquence ASCII continue, la proportion d'octets nuls, la moyenne et l'écart-type des valeurs d'octets, ainsi que le nombre de séparateurs PDF et ASCII. L'hétérogénéité locale est également mesurée par l'écart-type de l'entropie et celui du ratio d'octets imprimables sur huit sous-blocs de 64 octets.

Sur l'ensemble du corpus, l'entropie moyenne est de 7,22 et le ratio moyen de caractères imprimables atteint 42,20 %. Cette moyenne cache cependant des différences. Les fragments situés dans un flux présentent une entropie moyenne de 7,39 et un ratio imprimable de 39,22 %, contre respectivement 3,95 et 99,998 % pour les fragments situés hors des flux.

À partir de ces observations, six profils sont définis. La majorité du corpus appartient aux profils `in_stream_high_entropy` et `mixed`, qui représentent respectivement 57,53 % et 30,33 % des fragments. À l'inverse, le profil `structural_pdf`, plus lisible et plus riche en marqueurs PDF, ne représente que 3,18 % du corpus. La répartition complète des profils est présentée en annexe A.9.1.

Ces profils confirment que la reconstruction ne peut pas être abordée de manière uniforme. La majorité des pièces appartient à des zones compressées et peu lisibles, tandis qu'une fraction plus réduite, potentiellement plus informative, contient des éléments syntaxiques explicites ou se situe au voisinage des transitions.

9.2 Protocole expérimental et métriques

Le problème est d'abord formulé comme une classification binaire. À partir de deux fragments A et B, le modèle doit déterminer si B est le véritable successeur de A dans le document d'origine.

Pour chaque fragment A_i, appelé *ancre*, une paire positive est construite avec son vrai successeur A_{i+1}. Trois paires négatives sont ensuite formées avec des fragments choisis aléatoirement dans le même document, en excluant ce successeur. Ce ratio permet de conserver une classe positive suffisamment représentée pour l'entraînement, tout en exposant le modèle à plusieurs successeurs incorrects pour chaque ancre.

L'exactitude, la précision et le rappel sont utilisés pour évaluer cette classification. L'exactitude mesure la proportion totale de paires correctement classées. Pour la classe positive, la précision indique la proportion de paires réellement voisines parmi celles prédites comme telles, tandis que le rappel mesure la proportion de vrais voisins retrouvés.

La reconstruction demande cependant de classer plusieurs successeurs candidats pour une même ancre. Le Top-1 mesure la proportion de cas dans lesquels le vrai successeur obtient le meilleur score. Le Top-5 mesure la proportion de cas dans lesquels il figure parmi les cinq premiers candidats. Le modèle est d'abord évalué parmi 100 candidats, composés du vrai voisin et de 99 fragments tirés aléatoirement. Il est ensuite évalué sur le pool complet du document, une condition plus difficile puisque le nombre de candidats est beaucoup plus élevé.

9.3 Méthodes évaluées

9.3.1 Continuité statistique et Random Forest

Avant d'entraîner un modèle, un score de continuité statistique est défini manuellement. Il combine de manière pondérée les différences d'entropie, de ratio imprimable, de moyenne et d'écart-type des octets, ainsi que la distance entre les histogrammes. Ces mesures sont calculées sur une fenêtre de 128 octets de part et d'autre de la jonction.

Un classifieur Random Forest est ensuite entraîné pour déterminer si deux fragments sont voisins. Chaque paire (A, B) est d'abord décrite par des caractéristiques globales calculées sur les deux fragments. Elles comprennent l'entropie, le ratio d'octets imprimables, la moyenne et l'écart-type des valeurs d'octets, le nombre de séparateurs et le nombre de tokens PDF. Pour chacune de ces caractéristiques, le modèle reçoit la valeur de A, celle de B et leur différence absolue.

Une seconde configuration ajoute des caractéristiques qui décrivent explicitement la frontière. Pour des fenêtres de 32, 64 et 128 octets, les derniers octets de A et les premiers octets de B sont analysés séparément, puis ensemble. Les 128 derniers octets de A et les 128 premiers octets de B sont aussi concaténés afin de rechercher des marqueurs PDF susceptibles d'apparaître ou d'être reconstitués à la jonction.

9.3.2 CNN concaténé et réseau siamois

Alors que la Random Forest reçoit des caractéristiques calculées à la main, l'idée du CNN est de donner directement les octets bruts au modèle afin qu'il apprenne lui-même les motifs utiles. Un premier CNN reçoit la paire concaténée $A + B$, soit 1 024 octets, traitée comme un seul bloc par un encodeur unique. Une seconde configuration restreint l'entrée à la seule jonction.

En concaténant directement A et B, le CNN produit une représentation globale de la paire sans construire explicitement une représentation propre à chaque fragment pour pouvoir ensuite les comparer. Ce constat conduit à tester une architecture siamoise.

Cette architecture repose sur deux branches identiques qui partagent les mêmes paramètres. Chaque fragment est encodé séparément, puis les deux représentations obtenues sont comparées afin de prédire si les fragments sont voisins. Deux configurations sont évaluées, l'une sur les fragments entiers et l'autre sur une fenêtre de 128 octets autour de la jonction.

9.3.3 Reconstruction déterministe à partir de la structure PDF

Avant d'attribuer à l'apprentissage automatique toute la charge de la reconstruction, une piste déterministe est également testée. Un fichier PDF classique contient un en-tête, des objets numérotés, puis une table `xref` qui associe chaque objet à son offset dans le fichier. Le bloc `trailer` contient une référence vers l'objet racine, tandis que `startxref` indique la position de la table `xref`.

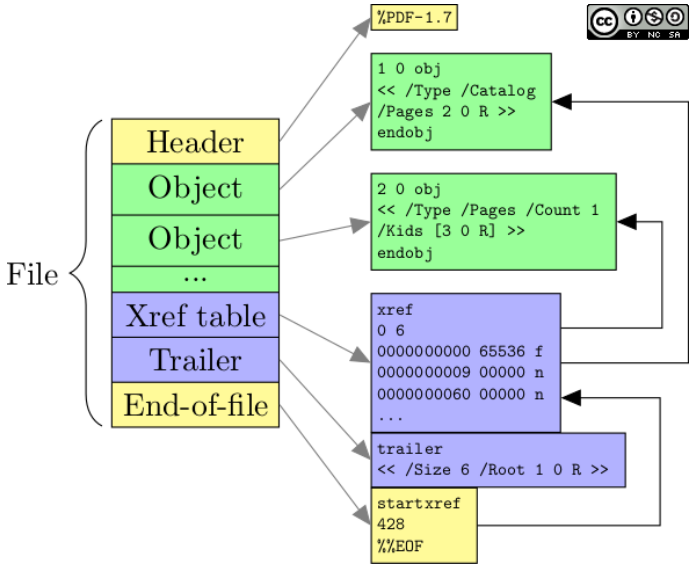

FIGURE 9 – Structure simplifiée d'un fichier PDF classique comportant une table `xref` [8].

Lorsqu'elle est intacte, cette structure peut fournir des placements précis sans apprentissage. Une reconstruction combinée est donc définie. Les en-têtes d'objets, le fragment contenant le `trailer` et la table `xref` sont repérés de manière déterministe. Les fragments restants sont ensuite traités par la Random Forest avec caractéristiques de frontière.

9.4 Résultats

9.4.1 Recherche d'adjacence sur un document

Le critère de continuité statistique sans apprentissage est évalué sur 500 fragments du document `hal_hal-05499643.pdf`. Le résultat est presque nul, avec un Top-1 de 0,4 % et un Top-5 de 1,8 %. La continuité statistique brute ne suffit donc pas.

La Random Forest fondée sur les caractéristiques globales obtient une exactitude de 63,95 %, avec une précision de 38,84 % et un rappel de 77,10 % pour la classe positive. Lors de la recherche du vrai successeur parmi 100 candidats, elle atteint 29,8 % en Top-1 et 46,6 % en Top-5.

L'ajout des caractéristiques de frontière améliore les performances. L'exactitude atteint 70,48 %, tandis que le Top-1 atteint 69,2 % et le Top-5 80,0 %. L'analyse de l'importance des variables montre que les mesures calculées aux frontières sont les plus discriminantes, notamment les différences d'entropie et d'écart-type entre les deux extrémités. Ce résultat rejoint l'intuition selon laquelle un puzzle s'assemble davantage par ses bords plutôt que par le motif de ses pièces.

Le CNN concaténé utilisant les fragments entiers obtient une exactitude de 72,70 %, mais un rappel de seulement 13,39 % et un Top-1 de 5,4 %. Le modèle rate donc la majorité des vrais voisins. Restreindre l'entrée à la seule jonction améliore l'exactitude et la précision, mais le Top-1 reste limité à 4,6 %.

Sur les fragments entiers, l'architecture siamoise atteint une exactitude de 76,45 %, un Top-1 de 47,6 % et un Top-5 de 73,0 %. À partir des mêmes octets bruts, le Top-1 passe ainsi de 5,4 % avec le CNN concaténé à 47,6 % avec le réseau siamois. Restreindre chaque entrée à une fenêtre autour de la jonction apporte un gain plus faible, avec un Top-1 de 50,0 % et un Top-5 de 76,2 %.

TABLE 11 – Comparaison des approches supervisées sur un document

Approche	Exactitude	Top-1	Top-5
RF avec caractéristiques de frontière	70,48 %	69,2 %	80,0 %
CNN concaténé, fragment entier	72,70 %	5,4 %	25,6 %
CNN concaténé, jonction	75,55 %	4,6 %	19,0 %
Réseau siamois, fragment entier	76,45 %	47,6 %	73,0 %
Réseau siamois, jonction	76,11 %	50,0 %	76,2 %

La Random Forest avec caractéristiques de frontière reste devant en Top-1. Lorsque la recherche est étendue au pool complet du document et plus à 100 candidats, elle atteint 44,6 % en Top-1 et 58,6 % en Top-5.

9.4.2 Généralisation à d'autres documents

Toutes les expériences précédentes reposent sur un protocole appliqué à un seul document. Les données d'entraînement et de test proviennent donc du même PDF. Pour vérifier si le signal obtenu est généralisable ou s'il relève d'un sur-apprentissage des particularités d'un seul fichier, deux protocoles sont comparés.

Dans le premier, un modèle distinct est entraîné et évalué sur chacun de six documents, puis une moyenne est calculée. Dans le second, un seul modèle est entraîné sur un pool de 81 documents, puis évalué sur des documents jamais vus pendant l'entraînement. Dans les deux cas, le vrai successeur doit être retrouvé dans le pool complet du document.

TABLE 12 – Comparaison des protocoles sur un document et entre documents

Approche	Top-1, un document	Top-1, nouveaux documents	Écart
RF sur statistiques globales	32,2 %	3,0 %	+29,2 points
RF avec caractéristiques de frontière	40,6 %	5,9 %	+34,7 points
CNN sur octets bruts	0,6 %	1,5 %	-0,9 point
CNN sur la jonction	0,6 %	1,9 %	-1,3 point
Réseau siamois sur octets bruts	1,7 %	2,2 %	-0,5 point
Réseau siamois sur la jonction	1,7 %	1,3 %	+0,4 point

Les performances diminuent pour les Random Forest. La RF basée sur les statistiques globales passe de 32,2 % de Top-1 à 3,0 %, tandis que la RF avec caractéristiques de frontière passe de 40,6 % à 5,9 %. Le modèle dédié à un seul document semble donc apprendre une partie importante de sa structure interne.

Pour les CNN, l'écart entre les deux protocoles est faible, voire inversé selon les configurations. Dans les configurations testées, ces résultats pourraient surtout indiquer que les CNN ont plus de difficultés à apprendre les relations d'adjacence entre les fragments. Le sur-apprentissage observé avec les Random Forest n'apparaît donc pas de la même manière.

9.4.3 Reconstruction déterministe et approche combinée

Sur quatre documents testés, la table `xref`, analysée avec `pypdf` à partir du fichier original, permet de retrouver l'ensemble des objets associés à un offset direct. En moyenne, 57,5 % des objets tiennent entièrement dans un seul fragment de 512 octets. Ils peuvent donc être placés exactement sans apprentissage. Pour les objets restants, les bornes exactes restent connues grâce à la table. Le bloc `trailer` tient systématiquement dans un seul fragment sur les documents testés.

Une reconstruction combinée est ensuite testée sur un document de 1 057 fragments, mélangés pour simuler un pool de carving.

Table 13 – Reconstruction combinée sur un document

Mesure	Valeur
Fragments placés par la méthode déterministe	175 / 1 057, soit 16,6 %
Exactitude du placement déterministe	174 / 175, soit 99,4 %
Top-1 de la RF sur le reste	36,5 %
Top-5 de la RF sur le reste	56,0 %
Proportion de fragments `in_stream` dans le reste	968 / 1 057, soit 91,6 %

La partie déterministe est très fiable, avec 99,4 % d'exactitude, mais ne couvre qu'une faible part des fragments, soit 16,6 %. La Random Forest atteint ensuite 36,5 % en Top-1 sur les fragments qu'il reste à placer.

Cette baisse est cohérente avec la difficulté observée pour les flux compressés, qui constituent 91,6 % des fragments du document. Le profil `stream_start` est correctement classé à 71,4 %, contre 33 à 36 % pour `in_stream_high_entropy` et `mixed`. La principale difficulté semble donc se concentrer sur les flux compressés plutôt que sur la structure elle-même.

Une expérience complémentaire compare la table `xref` extraite avec `pypdf` et celle reconstruite depuis les fragments. La première contient 405 objets, contre 403 pour la seconde. Cette différence de deux objets n'a pas d'effet sur l'expérience. Dans les deux conditions, la méthode déterministe place 159 fragments avec 100 % d'exactitude. La Random Forest obtient ensuite un Top-1 de 70,4 % et un Top-5 de 74,9 % sur le pool restant.

Sur ce document, le remplacement de la table `xref` oracle par une table reconstruite uniquement depuis les fragments ne dégrade donc pas la reconstruction. Cette observation reste limitée à un document qui dispose d'une table `xref` classique et non corrompue. Le détail de cette comparaison est présenté en annexe A.9.2.

9.5 Effet de la représentation et de la taille de la fenêtre

Le modèle Random Forest avec caractéristiques de frontière combine plusieurs tailles de fenêtres : 32, 64 et 128 octets.

Pour isoler l'effet de chaque paramètre, une étude d'ablation fait varier séparément la taille de la fenêtre et l'unité de représentation.

Des fenêtres de 32, 64, 96 et 128 unités sont comparées au niveau de l'octet, du nibble ou du bit.

Les résultats détaillés des douze configurations sont présentés en annexe A.9.3. Parmi les configurations isolées, les fenêtres de 32 octets obtiennent les meilleurs résultats. Sur le pool complet, le nibble conserve un léger avantage sur l'octet, avec 33,6 % contre 32,2 % en Top-1. L'écart reste cependant modeste.

Table 14 – Performances des configurations à fenêtres combinées selon la représentation

Configuration	Top-1, 100 candidats	Top-5, 100 candidats	Top-1, pool	Top-5, pool
RF frontière octet, statistiques et marqueurs	72,0 %	80,6 %	44,6 %	58,6 %
RF frontière bit, statistiques et marqueurs	66,8 %	79,4 %	44,4 %	56,0 %
RF bit pur, sans statistiques ni marqueurs	63,6 %	77,6 %	42,6 %	52,0 %

Le modèle à fenêtres combinées reste supérieur à chaque configuration isolée. Sur le pool complet, il atteint 44,6 % en Top-1, contre 33,6 % pour la meilleure configuration isolée. Cela pourrait donc indiquer que la fenêtre optimale n'est pas forcément une seule échelle mais plutôt une combinaison de plusieurs échelles qui, combinées, apporteraient plus d'information. Le niveau bit, même combiné, reste légèrement en retrait par rapport à l'octet, ce qui rejoint les observations faites lors de la classification des fragments.

9.6 Bilan et limites

La continuité statistique définie manuellement ne suffit pas à retrouver le vrai successeur, avec seulement 0,4 % en Top-1. En revanche, les modèles supervisés montrent qu'un signal d'adjacence existe. La Random Forest utilisant les caractéristiques de frontière atteint 69,2 % en Top-1 parmi 100 candidats et 44,6 % sur le pool complet.

Les informations situées aux extrémités sont particulièrement utiles. L'ajout des caractéristiques de frontière améliore la Random Forest. L'architecture siamoise confirme également l'intérêt d'une comparaison explicite entre les fragments, même si elle reste moins performante que la Random Forest.

La généralisation à des documents jamais vus reste difficile. Les performances élevées obtenues avec un modèle dédié à un document diminuent lorsque le modèle est appliqué à de nouveaux PDF.

Enfin, la structure du format PDF apporte des placements très fiables lorsque la table `xref` est intacte, mais elle ne couvre qu'une faible partie des fragments. Le reste du problème se concentre principalement sur les flux compressés, qui représentent la majorité du corpus et contiennent peu de marqueurs directement exploitables.

Ces résultats restent limités par le nombre réduit de documents utilisés dans les expériences. Les méthodes testées recherchent par ailleurs le successeur local d'un fragment et ne produisent pas encore une reconstruction complète et cohérente de l'ensemble du document.

10 Conclusion

10.1 Bilan et perspectives

Les expériences réalisées ont permis de répondre aux trois questions posées au début de ce rapport.

Tout d'abord, les résultats montrent que les caractéristiques les plus utiles dépendent de la nature du fragment. Les informations présentes au niveau de l'octet et les propriétés statistiques permettent de distinguer certains formats, tandis que les informations situées aux frontières sont particulièrement utiles pour rechercher des relations d'adjacence. La difficulté ne vient donc pas uniquement du choix du modèle, mais aussi de la nature et de la quantité d'information disponible dans chaque fragment.

Ensuite, les formats compressés ou composites restent les plus difficiles à reconnaître, notamment lorsque la taille des fragments diminue. Le focus mené sur le PDF et le DJVU a montré qu'une classe de fichier ne peut pas être traitée comme un bloc homogène. Un même format peut contenir plusieurs profils de fragments, structurels, mixtes ou compressés, avec des niveaux de classification très différents. La contrainte de 512 octets renforce cette difficulté en réduisant le contexte et les informations exploitables.

Enfin, les expériences de reconstruction de PDF ont montré qu'un signal d'adjacence existe entre les fragments, mais qu'il reste difficile à généraliser à des documents jamais vus. Elles montrent également l'intérêt de ne pas faire reposer toute la reconstruction sur l'apprentissage. Une partie de la structure du PDF peut être retrouvée directement à partir des fragments, notamment grâce à la table `xref`. Sa reconstruction depuis les fragments seuls obtient des résultats très proches de la version oracle sur les documents compatibles et permet de placer certains fragments de manière déterministe avant de traiter les fragments restants par apprentissage.

Une piste pour la suite serait donc de combiner davantage les informations structurelles propres aux formats avec les approches d'apprentissage, plutôt que de traiter tous les fragments de la même manière. La reconstruction des zones compressées reste notamment un problème ouvert, d'autant plus que les expériences ont été menées ici sans corruption des fragments.

10.2 Évaluation de mon expérience

Au cours de ce stage, j'ai pu développer plusieurs compétences, aussi bien techniques que transversales. J'ai notamment appris à construire un état de l'art cohérent avec un sujet de recherche, puis à m'appuyer sur celui-ci pour situer mes propres expérimentations par rapport aux travaux existants.

J'ai également appris à mettre en œuvre différentes expériences en apprentissage automatique et en apprentissage profond. Cela m'a amenée à construire, préparer et analyser des jeux de données constitués de fragments binaires, ainsi qu'à utiliser différentes plateformes de calcul, comme Kaggle.

Ce stage m'a aussi permis de mieux comprendre la structure interne de différents formats de fichiers, en particulier celle des fichiers PDF. Cette compréhension a progressivement fait évoluer ma manière d'aborder le problème, notamment lors du passage de la classification de fragments à leur reconstruction.

Plus globalement, j'ai appris à suivre une démarche de recherche de manière autonome, tout en étant guidée par mes superviseurs : formuler des hypothèses, construire des expériences pour les tester, analyser les résultats et, lorsque cela était nécessaire, remettre en question les hypothèses de départ. Les échanges avec mes superviseurs, dont les compétences différentes étaient complémentaires pour ce sujet, ont été particulièrement utiles pour faire évoluer mon travail.

Enfin, ce stage m'a permis d'améliorer l'organisation et le suivi de mon travail sur plusieurs mois. J'ai notamment tenu un carnet de suivi dans lequel je notais l'avancement des expérimentations, les résultats, les difficultés rencontrées et les questions à aborder avec mes superviseurs. Cet outil m'a permis de garder une trace de l'évolution du travail.

11 Références

[1] N. L. Beebe, L. A. Maddox, L. Liu, and M. Sun, "Sceadan : Using concatenated N-gram vectors for improved file and data type classification," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 9, pp. 1519–1530, 2013. https://doi.org/10.1109/TIFS.2013.2274728

[2] G. Mittal, P. Korus, and N. Memon, "FiFTy : Large-scale file fragment type identification using convolutional neural networks," *arXiv preprint arXiv :1908.06148*, 2020. https://arxiv.org/abs/1908.06148

[3] B. Chen, L. Liao, J. Jiang, Y. Fang, S. M. Yiu, J. Xi, S. Li, L. Yi, H. Wang, L. Hui, C. Liu, and J. Zhang, "File fragment classification using grayscale image conversion and deep learning in digital forensics," in *IEEE Security and Privacy Workshops (SPW)*, pp. 241–247, 2018. https://www.computer.org/csdl/proceedings-article/spw/2018/634901a140/12UTFwBQQMw

[4] W. Liu, K. Wu, T. Liu, Y. Wang, K.-H. Yap, and L.-P. Chau, "ByteNet : Rethinking multimedia file fragment classification through visual perspectives," *arXiv preprint arXiv :2410.20855*, 2024. https://arxiv.org/abs/2410.20855

[5] A. Guzhov and C. T. Wirth, "Transformer-based file fragment type classification for file carving in digital forensics," in *Proceedings of the European Conference on Cyber Warfare and Security (ECCWS)*, 2025. https://papers.academic-conferences.org/index.php/eccws/article/view/3552/3277

[6] M. Bhatt, A. Mishra, M. W. U. Kabir, S. E. Blake-Gatto, R. Rajendra, M. T. Hoque, and I. Ahmed, "Hierarchy-based file fragment classification," *Machine Learning and Knowledge Extraction*, vol. 2, no. 3, pp. 216–229, 2020. https://doi.org/10.3390/make2030012

[7] B. Zou and H. Liu, "Hierarchical deep learning for file fragment classification," *Electronics*, vol. 15, no. 7, art. 1507, 2026. https://www.mdpi.com/2079-9292/15/7/1507

[8] G. Endignoux, "Introduction to PDF syntax," *Blog de Guillaume Endignoux*, 4 octobre 2016. https://gendignoux.com/blog/2016/10/04/pdf-basics.html

[9] IEEE DataPort, "File Fragment Type (FFT-75) Dataset," *IEEE DataPort*. https://ieee-dataport.org/open-access/file-fragment-type-fft-75-dataset

A Annexes

A.1 Constitution et répartition du corpus maison

Le corpus initial peut être résumé de la manière suivante :

Table 15 – Constitution du corpus maison

Type	**Nombre de fichiers**	**Constitution du corpus**	**Source**
TXT	500	500 plus gros fichiers du dossier TXT de GovDocs1	GovDocs1
DOC	500	500 fichiers DOC	GovDocs1
JPG	500	202 images de la classe cats, 202 dogs, 96 premiers horses	Kaggle images dataset
PDF	500	500 premiers fichiers PDF	Kaggle PDF dataset
ENC	500	125 fichiers issus de TXT, DOC, JPG et PDF, chiffrés	Généré à partir du corpus
B64	500	125 fichiers issus de TXT, DOC, JPG et PDF, encodés en Base64	Généré à partir du corpus
ZIP	500	125 fichiers issus de TXT, DOC, JPG et PDF, compressés en ZIP	Généré à partir du corpus

Le jeu de données final contient entre 102 600 (fragments de 4 096 octets) et 104 820 (fragments de 128 octets) fragments, répartis sur 7 classes. Le tableau ci-dessous résume la répartition selon la taille de fragment.

Table 16 – Répartition des fragments du corpus maison selon les classes et la taille

Classe	**4 096 octets**	**512 octets**	**256 octets**	**128 octets**
TXT	15 030 (14,65 %)	15 030 (14,36 %)	15 030 (14,34 %)	15 030 (14,34 %)
DOC	15 030 (14,65 %)	15 030 (14,36 %)	15 030 (14,34 %)	15 030 (14,34 %)
PDF	14 190 (13,83 %)	14 790 (14,13 %)	14 820 (14,14 %)	14 820 (14,14 %)
JPG	14 550 (14,18 %)	14 970 (14,31 %)	15 000 (14,31 %)	15 000 (14,31 %)
ENCRYPTED	14 790 (14,42 %)	14 970 (14,31 %)	15 000 (14,31 %)	15 000 (14,31 %)
BASE64	14 790 (14,42 %)	15 030 (14,36 %)	15 030 (14,34 %)	15 030 (14,34 %)
ZIP	14 220 (13,86 %)	14 820 (14,16 %)	14 880 (14,20 %)	14 910 (14,22 %)
Total	**102 600**	**104 640**	**104 790**	**104 820**

A.2 Distributions 1-gramme moyennes par type (corpus maison)

Les figures suivantes complètent la synthèse du tableau de la section 6.2. Elles montrent, pour chaque type de fichier du corpus maison, la distribution moyenne des 256 valeurs d'octets sur 100 fragments de 4 096 octets.

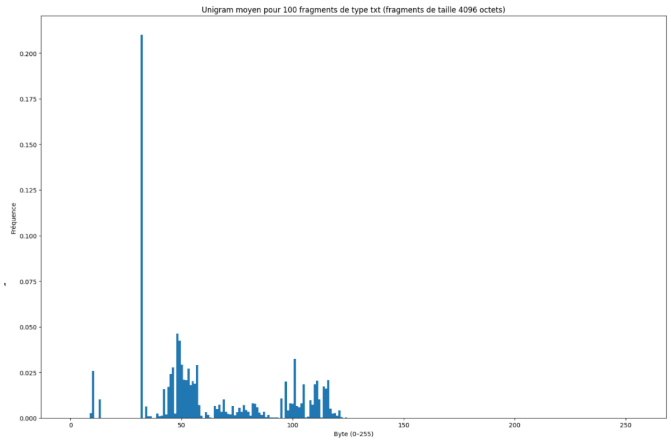

FIGURE 10 – Distribution 1-gramme moyenne des fragments TXT

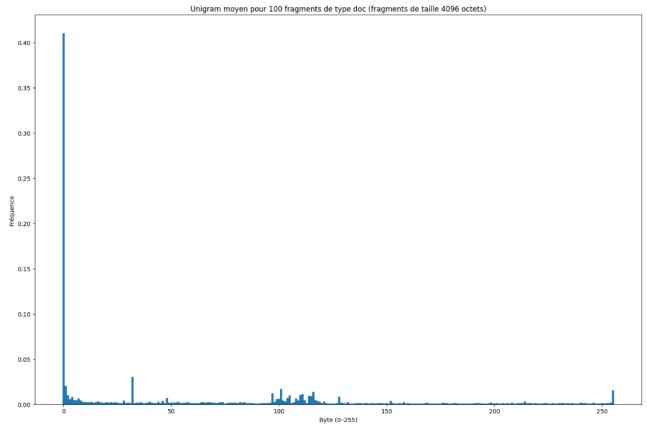

FIGURE 11 – Distribution 1-gramme moyenne des fragments DOC

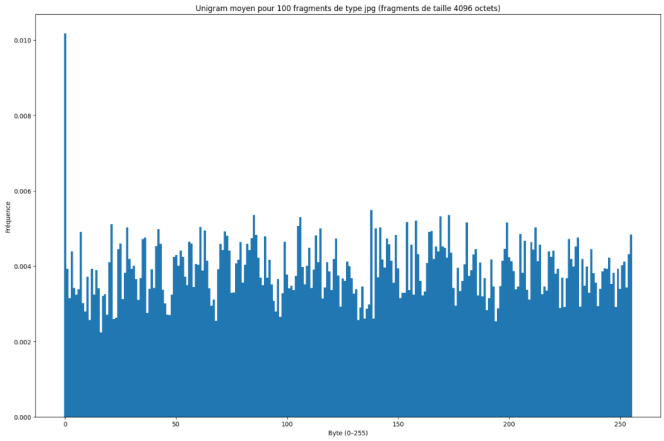

FIGURE 12 – Distribution 1-gramme moyenne des fragments JPG

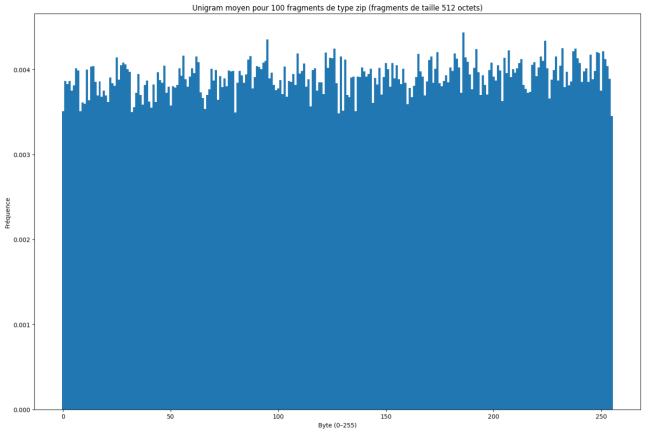

FIGURE 13 – Distribution 1-gramme moyenne des fragments ZIP

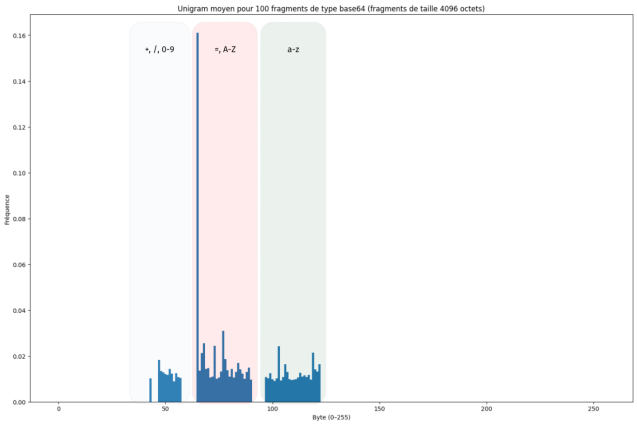

FIGURE 14 – Distribution 1-gramme moyenne des fragments Base64

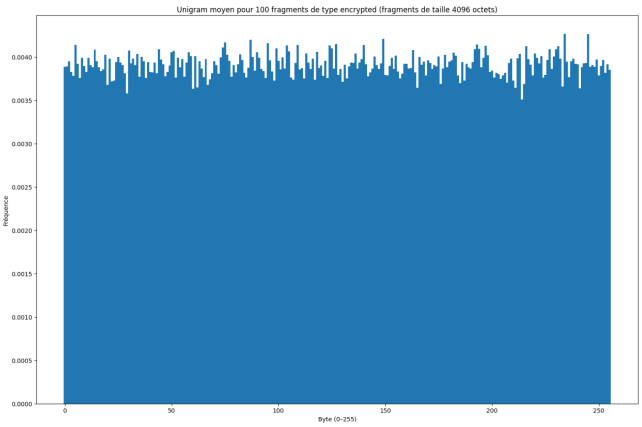

FIGURE 15 – Distribution 1-gramme moyenne des fragments chiffrés AES

A.3 Résultats complets des méthodes de référence de classification (corpus maison)

Trois configurations ont été évaluées sur le corpus maison, avec plusieurs classifieurs classiques, sur les quatre tailles de fragment (4 096, 512, 256, 128 octets).

(i) Sans entropie

	Caractéristiques utilisées															
	4 096				512				256				128			
Modèle	raw	1-g	2-g h.	3-g h.	raw	1-g	2-g h.	3-g h.	raw	1-g	2-g h.	3-g h.	raw	1-g	2-g h.	3-g h.
Random Forest	63,68	88,56	83,79	82,56	62,46	77,01	75,67	75,43	61,64	71,41	71,67	72,36	61,02	67,84	69,40	70,05
K-Nearest Neighbors	22,99	82,18	67,94	52,31	40,81	70,78	41,34	36,14	31,86	65,46	35,05	32,61	34,21	55,70	31,63	29,66
Support Vector Machine	32,94	84,66	81,00	79,98	29,96	74,46	77,16	77,19	28,63	71,50	74,94	74,64	27,63	67,37	72,33	**72,99**
Gradient Boosting	71,04	**89,29**	87,36	86,29	67,16	**79,14**	78,09	77,78	64,32	**75,32**	74,60	74,23	63,28	71,13	71,90	71,42
Logistic Regression	41,91	82,15	77,41	75,37	42,79	74,74	74,56	72,32	42,18	72,11	72,08	69,77	41,40	67,83	70,13	68,27
Naive Bayes	60,58	81,55	82,13	81,92	54,45	69,65	73,41	73,90	52,63	63,95	69,37	70,22	50,95	60,12	65,64	66,50
Decision Tree	56,60	79,52	77,62	77,47	56,12	68,38	67,39	67,10	55,08	65,18	62,85	63,00	55,05	62,74	59,03	60,65

TABLE 17 – Méthodes de référence sur le corpus maison, sans entropie (raw = fragment brut ; 1-g = unigramme, dim. 256 ; 2-g h. = bigramme hashé mod. 4 096 ; 3-g h. = trigramme hashé mod. 8192)

(ii) Avec ajout de la dimension entropie

	Caractéristiques utilisées															
	4 096				512				256				128			
Modèle	1-g	2-g h.	2-g svd	3-g	1-g	2-g h.	2-g svd	3-g	1-g	2-g h.	2-g svd	3-g	1-g	2-g h.	2-g svd	3-g
Random Forest	88,82 %	86,55 %	88,03 %	86,00 %	77,97 %	77,88 %	78,76 %	76,89 %	72,38 %	73,33 %	73,46 %	73,66 %	68,45 %	70,49 %	68,16 %	70,82 %
K-Nearest Neighbors	82,19 %	67,97 %	78,46 %	52,36 %	70,77 %	41,64 %	69,16 %	36,46 %	65,45 %	35,14 %	65,54 %	34,36 %	57,28 %	31,86 %	60,01 %	30,81 %
Support Vector Machine	83,99 %	81,22 %	73,26 %	81,08 %	77,52 %	79,02 %	70,85 %	79,05 %	73,16 %	**76,16 %**	68,54 %	75,94 %	68,94 %	73,48 %	65,37 %	**73,67 %**
Gradient Boosting	**89,68 %**	89,01 %	87,98 %	88,57 %	79,53 %	79,82 %	79,22 %	**79,98 %**	75,18 %	76,11 %	75,02 %	75,71 %	71,38 %	72,83 %	69,68 %	72,98 %
Logistic Regression	80,81 %	77,46 %	67,92 %	75,38 %	77,35 %	75,83 %	69,57 %	74,53 %	73,12 %	72,08 %	67,23 %	71,02 %	68,80 %	69,50 %	64,77 %	68,39 %
Naive Bayes	81,31 %	82,11 %	73,23 %	81,99 %	69,31 %	73,67 %	65,27 %	74,07 %	63,84 %	69,46 %	60,57 %	70,25 %	60,21 %	65,66 %	58,78 %	66,55 %
Decision Tree	81,21 %	80,88 %	81,86 %	81,92 %	69,55 %	70,84 %	70,74 %	71,11 %	65,97 %	66,46 %	66,79 %	67,28 %	62,81 %	64,72 %	62,82 %	64,48 %

TABLE 18 – Méthodes de référence sur le corpus maison, avec entropie (1-g = unigramme + entropie, dim. 257 ; 2-g h. = bigramme hashé mod. 4 096 + entropie ; 2-g svd = bigramme SVD dim. 128 + entropie ; 3-g = trigramme hashé mod. 8192 + entropie)

(iii) MLP

	Caractéristiques utilisées															
	4 096				512				256				128			
Modèle	raw	1-g	2-g h.	3-g h.	raw	1-g	2-g h.	3-g h.	raw	1-g	2-g h.	3-g h.	raw	1-g	2-g h.	3-g h.
Multi-Layer Perceptron	26,89	**88,47**	87,53	83,53	57,44	**81,10**	77,67	77,20	57,50	**76,58**	74,94	74,48	57,78	71,32	72,01	**72,28**

TABLE 19 – Méthodes de référence MLP sur le corpus maison (bigramme : hash mod. 4 096 ; trigramme : hash mod. 8192)

A.4 Répartition des classes FFT-75

Le tableau ci-dessous regroupe les 75 classes de FFT-75 par grande catégorie, avec le nombre total de fragments associés dans le jeu de données complet.

TABLE 20 – Répartition des classes FFT-75 par catégorie

Classes	Types	Nombre de fragments
Raw	ARW, CR2, DNG, GPR, NEF, NRW, ORF, PEF, RAF, RW2, 3FR	1 013 748
Bitmap	JPG, TIFF, HEIC, BMP, GIF, PNG	553 252
Vector	AI, EPS, PSD	276 509
Vidéo	MOV, MP4, 3GP, AVI, MKV, OGV, WEBM	644 913
Archives	APK, JAR, MSI, DMG, 7Z, BZ2, DEB, GZ, PKG, RAR, RPM, XZ, ZIP	1 197 506
Executables	EXE, MACH-O, ELF, DLL	368 995
Office	DOC, DOCX, KEY, PPT, PPTX, XLS, XLSX	645 029
Published	DJVU, EPUB, MOBI, PDF	368 684
Human-readable	MD, RTF, TXT, TEX, JSON, HTML, XML, LOG, CSV	829 825
Audio	AIFF, FLAC, M4A, MP3, OGG, WAV, WMA	644 912
Others	PCAP, TTF, DWG, SQLITE	368 627

A.5 Répartition des classes par bande d'entropie

Dans l'approche hiérarchique présentée en section 7.3.3, les 75 classes de FFT-75 sont réparties en trois groupes selon leur entropie moyenne : LowEntropy, MediumEntropy et HighEntropy. Les seuils utilisés sont 5 et 7.

TABLE 21 – Composition des groupes LowEntropy, MediumEntropy et HighEntropy

Groupe	Classes
LowEntropy (17)	MD, BMP, RW2, XLS, TXT, EPS, RTF, JSON, ELF, TEX, HTML, TTF, MACH-O, LOG, SQLITE, XML, DLL
MediumEntropy (17)	CSV, ORF, DOC, PSD, JPG, WMA, KEY, PPTX, MP3, AIFF, PCAP, DWG, MOBI, PPT, WAV, NRW, AI
HighEntropy (41)	NEF, CR2, DNG, RAF, 3FR, PDF, MOV, XLSX, PEF, JAR, MSI, GIF, OGG, OGV, RPM, MKV, 3GP, APK, ARW, GPR, M4A, ZIP, PKG, DMG, PNG, EPUB, EXE, DJVU, GZ, DOCX, TIFF, DEB, WEBM, RAR, MP4, BZ2, FLAC, AVI, HEIC, XZ, 7Z

A.6 Analyse des doublons intra-classe (FFT-75)

Le tableau suivant détaille, pour les classes présentant les plus faibles taux d'unicité intra-classe, le nombre de fragments, le nombre de fragments uniques, le taux d'unicité et le nombre de doublons intra-classe (cf. section 7.1).

TABLE 22 – Classes présentant les plus faibles taux d'unicité intra-classe

Classe	Fragments	Uniques	Unicité (%)	Doublons intra-classe
BMP	81 850	60 329	73,71	21 521
DOC	81 844	62 045	75,81	19 799
MD	81 195	63 035	76,88	18 960
KEY	81 702	74 180	90,79	7 522
PPTX	81 791	75 663	92,51	6 128
EPS	82 076	78 417	95,54	3 659
ELF	82 049	78 781	96,02	3 268
DLL	81 814	79 274	96,90	2 540
PCAP	81 867	79 496	97,10	2 371
MOV	81 970	79 691	97,22	2 279
MACH-O	82 099	79 887	97,31	2 212
MOBI	81 902	79 819	97,46	2 083
PPT	82 079	80 047	97,52	2 032
JAR	81 817	80 051	97,84	1 766
TTF	81 739	80 315	98,26	1 424

A.7 Vocabulaire structurel PDF – top 20

Le tableau ci-dessous complète le top 10 présenté en section 8.2.1, en détaillant les tokens.

TABLE 23 – Top 20 du vocabulaire structurel PDF (complément du top 10, section 8.2.1)

Rang	Token	Fréq. totale	Fréq. documentaire
1	«	2 951 793	1 869
2	»	2 951 638	1 869
3	`obj`	2 419 412	1 869
4	`endobj`	2 419 412	1 869
5	`[`	2 045 527	1 869
6	`]`	2 045 487	1 869
7	`/Type`	1 446 387	1 869
8	`/Subtype`	594 772	1 869
9	`endstream`	488 417	1 869
10	`stream`	488 313	1 869
11	`/Length`	488 310	1 869
12	`/Filter`	481 358	1 869
13	`/FlateDecode`	456 938	1 869
14	`/Size`	3 698	1 869
15	`/Root`	1 913	1 869
16	`startxref`	1 895	1 869
17	`%%EOF`	1 895	1 869
18	`/Font`	116 740	1 868
19	`/Pages`	8 106	1 868
20	`/Info`	2 036	1 868

Le vocabulaire structurel fait généralement ressortir des tokens directement liés à l'organisation interne du PDF : `obj`, `endobj`, `stream`, `endstream`, `xref`, `trailer`, `/Type`, `/Length`, `/Filter`, ou encore des noms de ressources. Ces éléments sont importants, car ils correspondent à des indices explicitement observables dans certains fragments.

A.8 Vocabulaire structurel DJVU – top 20

Le tableau ci-dessous détaille le vocabulaire structurel empirique du format DJVU, classé par fréquence documentaire.

Table 24 – Top 20 du vocabulaire structurel DJVU

Rang	Token	Fréq. totale	Fréq. documentaire
1	`[`	19 831 556	234
2	`]`	19 824 741	234
3	`FORM`	64 953	234
4	`DJVUINFO`	52 246	234
5	`X3`	48 582	234
6	`fl`	46 968	234
7	`g_`	46 154	234
8	`lw`	46 082	234
9	`By`	45 786	234
10	`l/`	45 479	234
11	`WO`	45 402	234
12	`xk`	45 308	234
13	`Ph`	45 302	234
14	`h3`	45 283	234
15	`Fo`	45 243	234
16	`Mm`	45 088	234
17	`A4`	44 973	234
18	`l6`	44 971	234
19	`Uy`	44 970	234
20	`ou`	44 949	234

Le vocabulaire empirique DJVU reste très différent de celui obtenu avec le PDF. Quelques tokens correspondent à des marqueurs plausibles du format, comme `FORM` ou `DJVUINFO`.

A.9 Résultats complémentaires sur la reconstruction des PDF

A.9.1 Répartition complète des profils de fragments

Table 25 – Répartition des profils de fragments PDF

Profil	Nombre de fragments	Proportion
`in_stream_high_entropy`	881 600	57,53 %
`mixed`	464 768	30,33 %
`stream_start`	72 528	4,73 %
`structural_pdf`	48 713	3,18 %
`other`	47 425	3,09 %
`stream_end`	17 328	1,13 %

Le profil `in_stream_high_entropy` correspond à des fragments situés dans des flux compressés. Leur entropie moyenne atteint 7,52 et leur nombre moyen de tokens PDF est faible, avec 1,73 token par fragment. Il s'agit des fragments les moins informatifs et potentiellement les plus difficiles à reconstruire.

Le profil `structural_pdf` se trouve exclusivement hors des flux. Sa lisibilité ASCII est presque totale et son nombre moyen de tokens PDF atteint 30,95. Le profil `stream_start` est fortement

associé au début des flux et présente un nombre moyen élevé de tokens PDF, avec 19,66 tokens par fragment.

A.9.2 Comparaison détaillée des tables xref

TABLE 26 – Reconstruction à partir de la table xref oracle et de la table reconstruite

Mesure	**Oracle avec pypdf**	**Fragments seuls**
Objets connus dans la table `xref`	405	403
Correspondance entre un objet et un fragment unique	395	393
Fragments placés par la méthode déterministe	159 / 1 057, soit 15,0 %	159 / 1 057, soit 15,0 %
Exactitude du placement déterministe	100,0 %	100,0 %
Top-1 de la RF sur le reste	70,4 %	70,4 %
Top-5 de la RF sur le reste	74,9 %	74,9 %

A.9.3 Étude détaillée de la taille et de l'unité des fenêtres

TABLE 27 – Résultats selon la taille de fenêtre et l'unité de représentation

Configuration	**Exactitude**	**Top-1**	**Top-5**
$k = 32$/octet	68,2 %	60,5 %	74,5 %
$k = 32$/nibble	68,0 %	60,5 %	71,0 %
$k = 128$/nibble	70,0 %	60,5 %	77,0 %
$k = 64$/nibble	69,3 %	58,0 %	70,5 %
$k = 128$/octet	68,9 %	57,5 %	70,0 %
$k = 96$/nibble	69,4 %	57,5 %	74,0 %
$k = 96$/octet	67,7 %	57,0 %	73,5 %
$k = 64$/octet	68,9 %	55,5 %	72,5 %
$k = 128$/bit	69,9 %	55,5 %	69,5 %
$k = 32$/bit	71,0 %	51,5 %	70,0 %
$k = 64$/bit	70,6 %	50,5 %	69,0 %
$k = 96$/bit	70,0 %	48,0 %	69,0 %

À l'issue de ce premier test, les configurations $k = 32$/octet et $k = 32$/nibble sont retenues. Les petites fenêtres obtiennent les meilleurs scores en Top-1. Le niveau bit reste en retrait, quelle que soit la taille de la fenêtre, malgré un taux de classification globale parfois plus élevé.

TABLE 28 – Confirmation des deux meilleures configurations sur 500 ancres

Configuration	**Top-1, 100 candidats**	**Top-5, 100 candidats**	**Top-1, pool**	**Top-5, pool**
$k = 32$/octet	59,6 %	77,6 %	32,2 %	45,6 %
$k = 32$/nibble	61,2 %	76,8 %	33,6 %	47,8 %