RAPPORT DE PROJET DE MASTER

R.A.I.D.

RAM Artificial Inspection Dump

14 mai 2026

HENRY Amand — ROUSÉE Tom — SICOT Théo

Supervisé par :
GIGUET Emmanuel — BENOIT Tristan

Résumé

Ce projet présente le développement de R.A.I.D. (RAM Artificial Inspection Dump), une solution d'analyse forensique visant à classifier les données issues de la mémoire vive (RAM). L'objectif central est de segmenter des dumps mémoire bruts pour distinguer les données chiffrées des informations non chiffrées (textes, images, PDF) sans connaissance préalable de leur structure. L'approche repose sur l'utilisation d'un modèle d'apprentissage profond de type *Transformer*, capable d'évaluer la prédictibilité des séquences d'octets. La méthodologie comprend la génération d'un jeu de données synthétique, l'entraînement d'un classifieur d'octets avec mécanisme d'attention, et une validation par vote pondéré.

Les résultats démontrent que l'approche s'affranchit des vulnérabilités des méthodes traditionnelles basées sur des signatures statiques, prouvant la viabilité des architectures Transformer pour l'interprétation sémantique de flux binaires bruts. Grâce à la stratégie de vote pondéré, le système parvient à transformer des prédictions locales incertaines en une segmentation globale robuste. Si le modèle isole avec succès les zones chiffrées des structures conventionnelles, la distinction mathématique entre le chiffrement et la forte compression à haute entropie demeure un défi complexe.

Table des matières

1 Introduction

1.1 Contexte

L'analyse forensique de la mémoire vive (RAM) constitue un pilier fondamental de l'informatique légale. Contrairement au stockage persistant, la RAM contient des artéfacts volatils critiques : clés de chiffrement, fragments de documents, ou résidus de textes souvent invisibilisés dans de grandes quantités de données déstructurées.

Ceci, combiné à l'augmentation constante des capacités de mémoire, et à la multitude de formats de données rend l'inspection manuelle ou basée sur des signatures classiques de plus en plus laborieuse.

1.2 Problématique

Dans ce contexte, le groupe de travail sur la forensique de l'équipe SAFE du laboratoire GREYC a initié des recherches visant à automatiser l'interprétation de dumps mémoire.

Si des travaux antérieurs, notamment dans le cadre de thèses sur les cartes à puce, ont permis de distinguer les données chiffrées des informations textuelles sans connaissance préalable du format, le défi actuel réside dans le passage à l'échelle. Il s'agit désormais de traiter des fragments de mémoire arbitraires et des structures complexes comme les fichiers PDF ou les images binaires.

1.3 Objectifs

Le projet R.A.I.D. (RAM Artificial Inspection Dump) s'inscrit dans cette dynamique en proposant une approche basée sur l'apprentissage profond. L'objectif central est de développer un système capable de classifier les octets d'un dump de RAM pour déterminer leur nature : chiffrée ou non, et ce de manière autonome.

L'implémentation repose sur l'utilisation de l'architecture *Transformer* et en particulier l'encodeur ainsi que son mécanisme de *self-attention* associé.

L'hypothèse de travail est qu'un modèle de langage, entraîné à classer des séquences d'octets présentera plus de perplexité sur des données chiffrées et compressés (par nature à forte entropie) tout en identifiant facilement les structures de données plus conventionnelles et prévisibles, telles que du texte brut.

Cette différenciation doit permettre à terme de segmenter précisément la mémoire et d'identifier les différents champs de données sans supervision humaine, la difficulté réside dans la similitude d'une donnée chiffrée et compressée sur le plan mathématique.

2 Méthodologie

2.1 Collecte de données

Pour constituer notre corpus initial, nous avons agrégé des données issues de trois sources : des documents PDF via l'API d'arXiv, des images via l'API de Lorem Picsum et des textes provenant du projet Gutenberg. Afin de *Transformer* cet échantillon en un jeu de données réaliste et exploitable, ces fichiers bruts sont soumis à un pipeline de transformations reproduisant la diversité des données en mémoire.

Concrètement, des fragments de ces données sont traités selon différentes méthodes (chiffrement AES-256-CBC, compression zlib ou gzip, encodage Base64, extraction de pixels bruts) avant d'être intégrés dans une simulation de capture mémoire (RAM dump) incluant du bruit système. L'outil permet de contrôler la répartition par type de données afin de s'adapter selon les besoins.

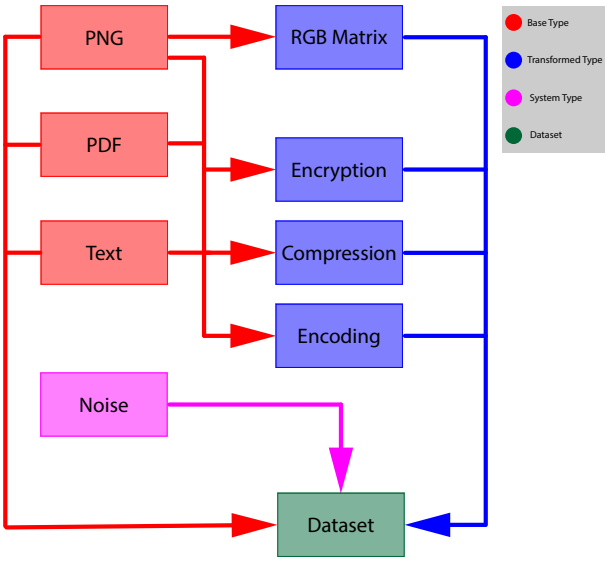

Figure 2.1 – Arbre représentatif des transformations

Nos jeux de données reposent sur une classification binaire distinguant les données chiffrées des données non chiffrées. Afin de simuler des dumps mémoire aussi réalistes et diversifiés que possible pour l'entraînement du modèle, ces deux classes principales englobent une multitude de sous-types. D'une part, la classe des données non chiffrées comprend du texte brut, des documents PDF, des images, des matrices de pixels, ainsi que des données

ayant subi des transformations courantes (encodage Base64, algorithmes de compression à taux variables). À cela s'ajoutent du bruit aléatoire (noise) et des simulations de traces système. D'autre part, la classe chiffrée est exclusivement constituée de données traitées via l'algorithme AES, appliquées de manière aléatoire sur les formats sources précédemment cités (PDF, images, textes).

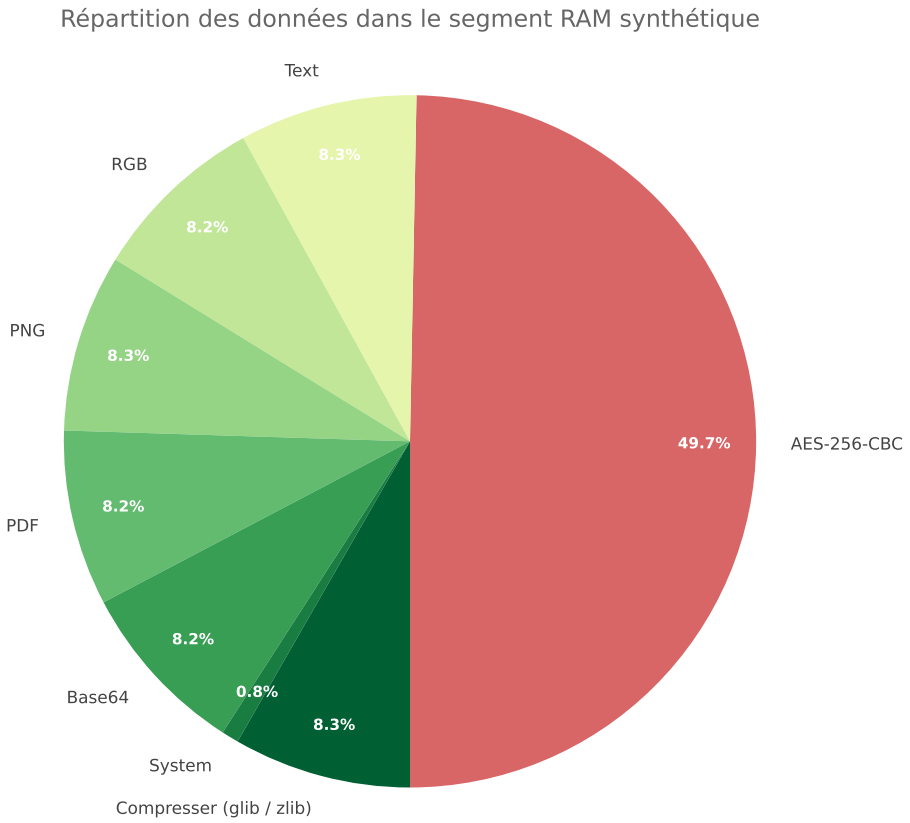

FIGURE 2.2 – Répartition des données dans le dump mémoire synthétique

Afin de mener à bien l'apprentissage et l'évaluation du modèle, nous avons généré trois jeux de données dédiés à l'entraînement, à la validation et au test. L'ensemble d'entraînement possède une taille de 1.2 Go dont 20% sont réservés exclusivement à la validation, tandis que l'ensemble de test mesure 500 Mo. Pour garantir une évaluation non biaisée, ces trois corpus partagent une distribution de classes strictement équilibrée, composée à 50% de données chiffrées et à 50% de données en non chiffrées.

2.2 CRÉATION DU SEGMENT RAM SYNTHÉTIQUE

Une fois transformées, les données sont injectées dans un espace mémoire contigu (le dump synthétique). Les fichiers ajoutés à cet espace peuvent se retrouver fragmentés et placés à divers endroits, ils ne sont pas simplement ajoutés à la suite. Le générateur détermine la taille et l'emplacement des fragments de manière aléatoire en respectant néanmoins des limites de taille configurées. Cette opération permet de se rapprocher du fonctionnement réel de la ram puisqu'il est rare d'avoir une données stockées en un seul bloc. L'intégralité de l'aléatoire du projet est seedé, donc reproductible.

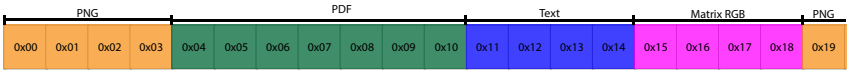

FIGURE 2.3 – Illustration d'un dump RAM synthétique

2.3 MODÈLE DE CLASSIFICATION

2.3.1 Introduction au modèle

Transformers

L'architecture *Transformer* représente un changement de paradigme majeur dans le domaine de l'apprentissage profond (Deep Learning). Introduit par *Google* en 2017 à travers l'article *« Attention is All you Need »* [1], ce modèle s'est imposé comme le standard pour le traitement de données séquentielles, notamment dans le traitement automatique des langues (NLP).

Contrairement aux architectures récurrentes conventionnelles (RNN et LSTM), qui opèrent par une analyse chronologique et itérative de l'information, le *Transformer* repose sur un principe de parallélisation massive des flux de données.

Architecture Encodeur

Un encodeur traite et modélise l'information d'entrée en convertissant des données brutes en représentations vectorielles de haute dimension. Lorsqu'il est utilisé indépendamment d'un décodeur, ce module est particulièrement efficace pour les tâches de classification ou de compréhension sémantique, où la génération de texte n'est pas requise.

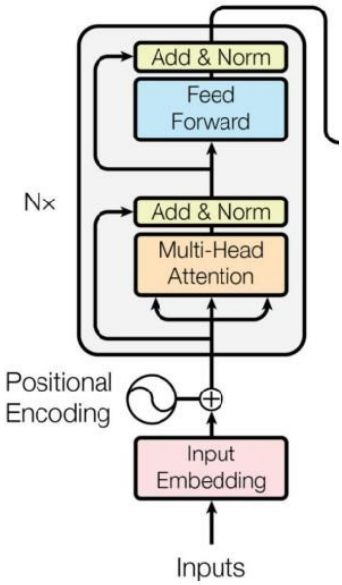

FIGURE 2.4 – Illustration d'un encodeur du papier *« Attention is All you Need »* [1]

Input Embedding

L'input embedding transforme les données brutes en représentations vectorielles de dimension d_{model}, rendant l'information traitable mathématiquement par le modèle.

Une fois projetés dans l'espace vectoriel, ces vecteurs sont multipliés par un facteur d'échelle $\sqrt{d_{model}}$ afin de stabiliser les gradients lors de l'entraînement.

Contrairement aux réseaux récurrents (RNN), l'achitecture *Transformer* traite les elements d'une séquence simultanément. Pour compenser cette absence de notion d'odre intrinsèque, un encodage de position est ajouté à chaque vecteur d'entrée. Cette information permet au modèle de distinguer la position relative de chaque element dans la séquence.

Cet encodage repose sur des fonctions sinusoïdales de fréquences différentes, définies par les formules suivantes :

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.1}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.2}$$

Mécanisme d'attention

Le mécanisme de *self-attention* permet au modèle de pondérer l'importance relative de chaque élément d'une séquence par rapport à tous les autres au sein de la même entrée.

En projetant chaque token dans trois espaces distincts(Q, K, V), le modèle calcule une distribution de scores par produit scalaire. Cette opération permet d'extraire les dépendances contextuelles, qu'elles soient locales ou à longue distance, en focalisant le traitement sur les parties les plus informatives de la séquence.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$

$$Q = \text{matrice de requêtes}$$
$$K = \text{matrice de clés}$$
$$V = \text{matrice de valeurs}$$
$$d_k = \text{dimension des clés}$$

L'attention multi-têtes, c'est l'application de plusieurs mécanismes de *self-attention* en parallèle, dans le but de spécifier et d'extraire plusieurs métriques (types, syntaxe ...) afin de ne pas écarter de sens exploitable. Leurs résultats sont donc fusionnés pour extraire un unique vecteur portant toutes les informations sur le sens.

Sortie du modèle

A la sortie du modèle on trouve des logits (score une valeur entre $-\infty$ et $+\infty$), nous y appliquons une fonction sigmoïde pour *Transformer* entre 0 et 1, entre les deux classes.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

2.4 BytesTransformerClassifier

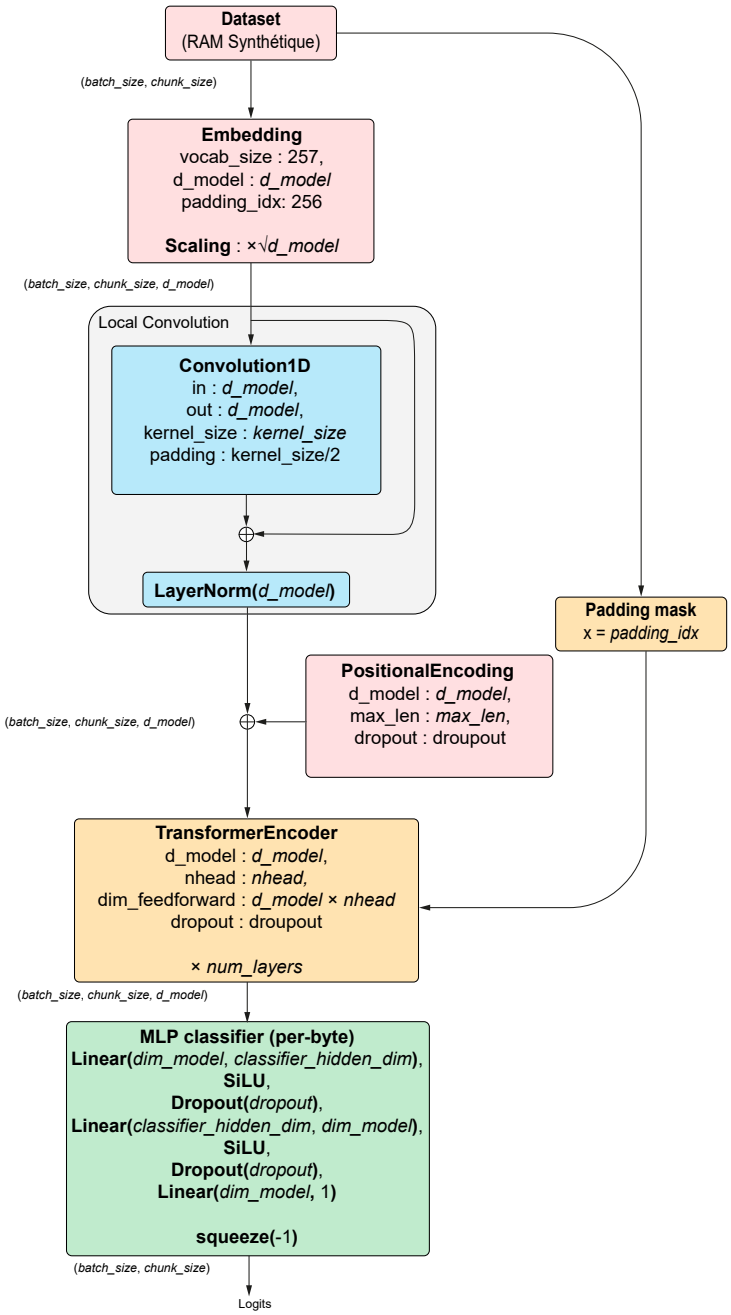

FIGURE 2.5 – Pipeline de classification du `BytesTransformerClassifier`

Paramètre	Valeur
chunk_size	1024
d_model	256
kernel_size	3
num_heads	8
num_layers	4
dropout	0.2
classifier_hidden_dim	256

TABLE 2.1 – Paramètres du classifieur RAID

Nous avons opté pour une architecture hybride combinant une couche de Convolution unidimensionnelle (*Local Convolution* dans la fig 2.5) en amont du bloc d'encodeur. Cette configuration permet d'extraire des motifs locaux au sein des séquences d'octets, tout en bénéficiant de la capacité du *Transformer* à modéliser les dépendances à longue distance.

2.5 ENTRAÎNEMENT ET VALIDATION

FIGURE 2.6 – Pipeline d'entraînement du modèle

Nous utilisons `BinaryCrossEntropy`. Ce choix est justfié car nous classifions chaque octet entre deux classes (chiffré ou non chiffré).

Les paramètres du modèle sont optimisés via AdamW, configuré avec un *weight decay* de $1e^{-2}$ ainsi que le *learning rate* de $1e^{-4}$ afin de régulariser l'apprentissage sur les 1 Go de données synthétiques. Cet optimiseur permet de naviguer efficacement dans l'espace de haute dimension généré par le mécanisme de self-attention.

2.6 PRINCIPE DE *l'overlapping* (RECOUVREMENT)

Le modèle traite le dump mémoire par fenêtres glissantes de `N` octets. Au lieu de découper la mémoire en blocs contigus, nous utilisons un pas de glissement (stride) inférieur à la

taille de la fenêtre. Par conséquent, chaque octet de la RAM est analysé plusieurs fois par le Transformer, au sein de contextes différents.

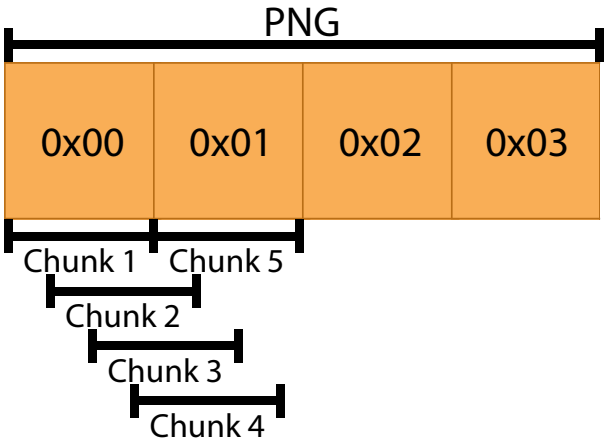

FIGURE 2.7 – Illustration du principe de l'overlapping, ici avec un stride de $\frac{1}{4}$ de chunk

2.7 PIPELINE DE VOTE PONDÉRÉ

Après avoir transformé les logits en probabilités via une fonction sigmoïde, les prédictions locales sont agrégées. Cette étape repose sur une pondération de chaque résultat par le score de confiance associé.

$$\text{Confiance}_x = 2 \times |p_x - 0.5| \tag{2.5}$$

$$\text{P\_Global}_x = \frac{\sum_{i=1}^{n} \text{Confiance}_i \times p_i}{\sum_{i=1}^{n} \text{Confiance}_i} \tag{2.6}$$

$$\text{avec } n = \frac{\text{chunk\_size}}{\text{stride}}$$

2.8 OPTIMISATION

Nous sommes arrivés à un stade où nous avions besoin d'augmenter la taille de nos datasets, nous avons donc dû passer par une très grosse phase d'optimisation. Nous avons d'abord repensé la gestion de la mémoire lors du chargement des données en implémentant le *memory mapping* (`mmap`), ce qui permet d'accéder directement aux segments du dump sans charger l'intégralité du fichier en RAM. Pour éviter les goulots d'étranglement lors de la constitution des batchs, nous avons pré-calculé les masques de labels sur l'ensemble du da-

taset, garantissant un accès en complexité $O(1)$, le tout couplé à un DataLoader utilisant des persistent workers et un `prefetch_factor` élevé. Côté architecture matérielle, nous avons allégé l'empreinte VRAM et accéléré les calculs de notre Transformer en exploitant la précision mixte (bfloat16) et en activant la compilation du graphe PyTorch (`torch.compile`) via le backend inductor en mode max-autotune. Ce backend reposant sur le compilateur Triton (fortement dépendant de l'environnement), nous avons mis en place une image Docker dédiée afin de garantir la portabilité et la reproductibilité de ces optimisations.

2.9 OUTILS D'ANALYSE

Pour faciliter l'interprétation des erreurs de classification et valider spatialement les prédictions du modèle, nous avons développé une interface de visualisation dédiée en React. Cet outil permet de confronter dynamiquement le dump binaire initial avec les fichiers de logs générés lors de l'inférence. En superposant les prédictions du Transformer à la "vérité terrain" (ground truth), nous pouvons identifier visuellement si les erreurs se concentrent sur des zones spécifiques, comme les transitions entre deux types de données ou les structures internes de certains fichiers.

FIGURE 2.8 – Interface de visualisation des prédictions du modèle

3 Résultats

3.1 Courbes d'apprentissage

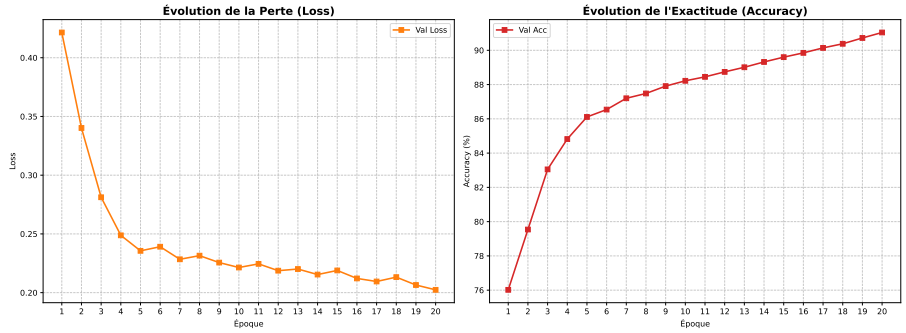

FIGURE 3.1 – Courbes d'apprentissage du modèle

3.2 Performance Globales

3.2.1 Métriques d'évaluation

Métrique finale	Score
Accuracy	88,79%
Precision	≈ 86%
Recall	85,21%
F1-Score	85,60%

TABLE 3.1 – Performances globales du classifieur RAID

3.2.2 Le vote pondéré

Accuracy		
Sans vote pondéré	Avec vote pondéré	Diminution de l'erreur
87,59%	88,79%	9.67%

TABLE 3.2 – Performances du vote pondéré

3.2.3 Matrices de confusion

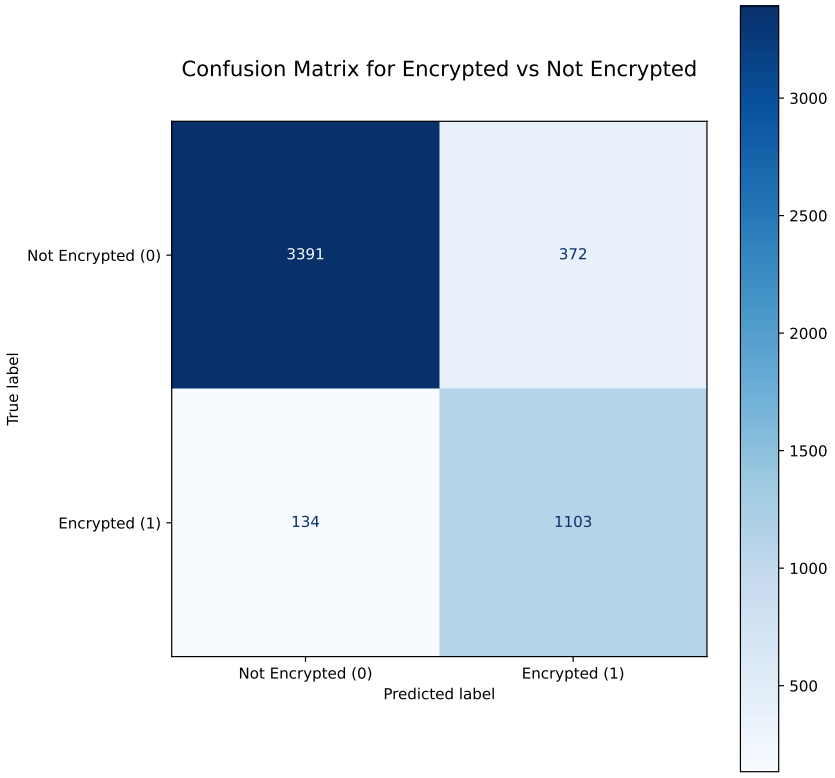

FIGURE 3.2 – Matrice de confusion du modèle

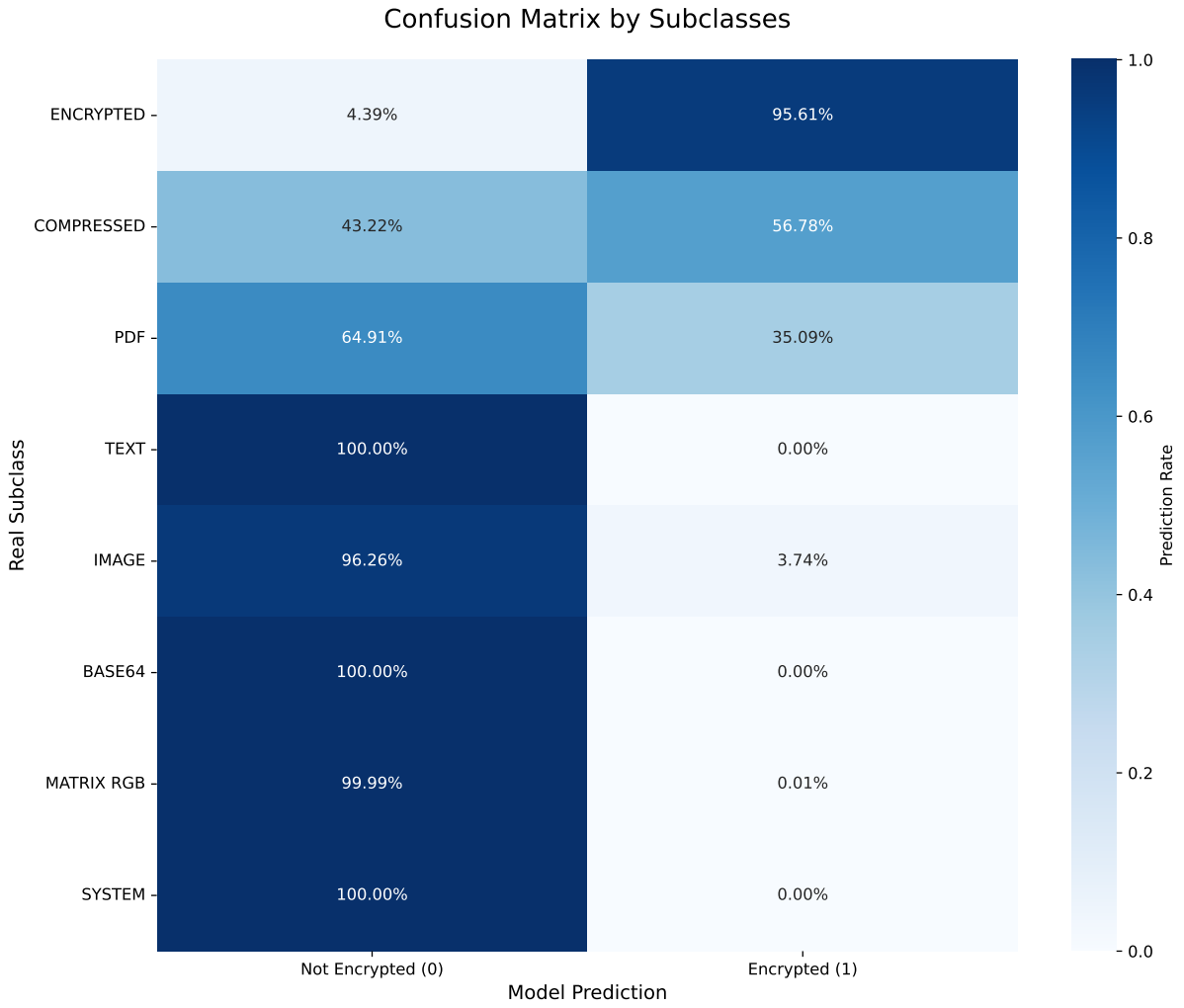

FIGURE 3.3 – Matrice de confusion du modèle

4 Discussion

4.1 Performance du vote pondéré

L'un des défis majeurs de l'analyse de dumps mémoire par un *Transformer* réside dans la volatilité des prédictions locales. Un segment de données peut présenter des caractéristiques ambiguës (par exemple, un en-tête de fichier compressé ressemblant à du texte structuré) qui induisent le modèle en erreur s'il est analysé de manière isolée.

Pour pallier cette limite, nous avons implémenté un mécanisme de vote pondéré. Cette approche permet de stabiliser les prédictions et d'augmenter la fiabilité de la classification des zones de mémoire.

En confrontant plusieurs prédictions pour un même octet via des fenêtres glissantes, nous avons observé une réduction des “artefacts de prédiction" (changements de classe abrupts et incohérents sur quelques octets). Cette approche agit comme un filtre passe-bas, éliminant le bruit haute fréquence dans les résultats pour ne conserver que les zones où le modèle maintient une confiance élevée et constante.

Il est important de noter que cette robustesse a un coût. L'utilisation d'un *stride* (pas) réduit pour multiplier les votes multiplie proportionnellement le temps d'inférence. Dans des grands volumes de données, ce compromis entre le temps d'analyse et la précision de la segmentation est un paramètre critique. Nos résultats suggèrent que le gain en fiabilité justifie cette latence, tout en ouvrant la porte à des optimisations futures sur le parallélisme de l'agrégation des votes.

4.2 Chiffrement et compression : l'entropie trompe le modèle

L'analyse des erreurs de classification révèle une confusion récurente du modèle entre les segments de données chiffrés et compressés. Cette difficulté s'explique par la nature mathématique même des algorithmes utilisés dans le projet, tels que l'AES-256-CBC pour le chiffrement et zLib ou Gzip pour la compression.

4.2.1 La convergence statistique des hautes entropies

Le modèle *Transformer* repose sur sa capacité à identifier des structures et des dépendances au sein des séquences d'octets. Or, l'objectif d'un algorithme de compression efficace est d'éliminer toute redondance, ce qui conduit à une distribution des octets proche d'un bruit blanc aléatoire, simulant ainsi une entropie maximale. Le chiffrement et la compression sont deux processus produisent des flux binaires dépourvus de motifs répétitifs exploitables par les mécanismes d'attention du modèle.

4.2.2 Impact sur la précision de la segmentation

Dans les résultats d'évaluation, cette confusion se traduit par des faux positifs où des flux compressés sont étiquetés comme `ENCRYPTED`. Les résultats montrent que même avec une validation par vote pondéré, le score de confiance reste souvent élevé pour ces deux catégories, car le modèle "est certain" de ne pas faire face à du texte ou des images structurées, mais ne possède pas de descripteurs sémantiques suffisants pour distinguer un flux compressé d'un flux chiffré.

4.2.3 Limites de l'approche sémantique pure

Cette problématique souligne une limite intrinsèque de l'apprentissage profond appliqué aux flux binaires bruts : sans la détection de signatures spécifiques que le projet cherche justement à éviter pour gagner en généricité, la distinction entre haute entropie "utile" (compression) et "intentionnelle" (chiffrement) demeure un défi ouvert. Pour les travaux futurs, l'intégration de métriques statistiques complémentaires, telles que des tests de distribution de fréquences bien plus fins au sein des couches du classifieur, pourrait offrir une piste pour séparer ces deux types de données

4.2.4 Sémantique binaire vs Identification rigide

Les méthodes classiques échouent lorsque les données sont fragmentées ou dépourvues de leurs en-têtes originaux, une situation fréquente au sein d'un dump RAM. Là où un outil statique verrait une suite d'octets incohérente, le modèle parvient à extraire une "signature statistique" du contenu. En apprenant la sémantique profonde des différents types de données (la structure d'un texte, la distribution d'une image compressée), le *Transformer* identifie la nature de l'information même en l'absence de métadonnées explicites.

4.2.5 Résilience face à l'obfuscation

Une limite majeure des signatures statiques réside dans leur vulnérabilité aux techniques d'obfuscation simples. Un changement mineur dans les premiers octets d'un fichier suffit à tromper un scanner traditionnel. À l'inverse, l'utilisation du mécanisme d'attention du *Transformer* permet au modèle de pondérer l'importance de chaque octet dans son contexte global. Cette capacité rend le système beaucoup plus résilient : la classification ne repose pas sur un point de passage unique, mais sur une compréhension diffuse du bloc de données de 1024octets.

4.2.6 Le coût de l'intelligence

Il est toutefois nécessaire de nuancer cette supériorité par le coût computationnel associé. Si les signatures statiques sont quasi instantanées et légères, l'inférence d'un modèle *Transformer* couplée à une stratégie de vote pondéré, exige des ressources matérielles significatives (GPU). Cette discussion souligne un arbitrage nécessaire : les méthodes classiques restent d'une utilité majeure en l'absence de matériel performant ou lors d'analyses ou l'on cherche la rapidité brute.

4.3 LIMITES

4.3.1 La généralisation face aux données réelles

Bien que les résultats soient prometteurs, il est important de souligner que le modèle a été entraîné et évalué sur un jeu de données synthétique. Bien que ce dernier s'efforce de reproduire la diversité d'un dump mémoire, il existe inévitablement des écarts avec la complexité d'une mémoire vive réelle, notamment au niveau de la fragmentation dynamique et des structures de données propriétaire liées aux OS.

On peut aussi considérer que dans la RAM, la présence de formats propriétaires non inclus dans le dataset pourraient faire l'objet d'erreurs de classification si leur format paraît trop éloigné de la signature statistique des données présentes dans le corpus d'entraînement. La non exhaustivité des types présents dans notre dataset peut donc influencer négativement les performances de la classification du modèle.

4.3.2 Le choix de l'architecture *Transformer*

Le passage d'une analyse par signatures à une analyse par apprentissage profond a nécessité le choix d'une architecture capable de saisir la complexité sémantique des flux binaires. Si les réseaux de neurones récurrents (RNN ou LSTM) ont longtemps été la norme pour les données séquentielles, le choix d'un *Transformer* pour le projet R.A.I.D. répond à des problématiques spécifiques à l'analyse de la mémoire vive.

Multi-Head Attention vs Dépendances séquentielles

La principale limite des RNN réside dans leur nature séquentielle : ils traitent les octets l'un après l'autre, ce qui rend difficile la capture de relations à longue distance au sein d'un bloc de données. À l'inverse, le mécanisme d'attention multi-têtes permet au modèle d'analyser simultanément toutes les relations entre les octets d'une fenêtre de 1024octets. Cette capacité est cruciale : un en-tête situé au début d'un segment peut avoir une influence sémantique directe sur des données situées bien plus loin. Là où un RNN aurait pu "oublier" le contexte initial, le Transformer maintient une vision globale et pondère l'importance de chaque zone du dump avec une précision accrue.

L'importance de l'encodage positionnel

Contrairement aux images ou au texte naturel, la structure d'un dump mémoire est strictement liée à l'adresse physique des données. L'absence de structure inhérente aux *Transformers* (qui sont, par nature, invariants par permutation) a été compensée par l'implémentation d'un Positional Encoding. Cette composante n'est pas qu'un détail technique : elle permet au modèle de conserver la notion d'ordre et de distance entre les octets. Dans notre contexte, savoir qu'un octet spécifique se situe à une position relative précise par rapport à un autre est souvent le seul moyen de distinguer un motif structuré d'un flux aléatoire.

Parallélisation et passage à l'échelle

Enfin, un argument pragmatique a dicté ce choix : la performance. L'architecture *Transformer* permet une parallélisation totale lors de l'entraînement et de l'inférence, contrairement aux RNN qui sont limités par leur récursivité. Étant donné la volumétrie des dumps RAM modernes (souvent supérieurs à 16 Go), l'utilisation des cœurs de calcul des GPU via une architecture *Transformer* est la seule voie viable pour obtenir des temps d'analyse convenables.

4.4 PERSPECTIVES

4.4.1 Passage en apprentissage non supervisé

Actuellement, le système repose sur un entraînement supervisé à l'aide d'un générateur de datasets qui définit arbitrairement les classes de données. Cependant, la réalité d'un dump RAM système est bien plus complexe et hétérogène que les échantillons produits synthétiquement. Un passage à l'apprentissage non supervisé permettrait au modèle d'apprendre directement les structures statistiques intrinsèques de la mémoire vive sans nécessiter d'étiquetage préalable, facilitant ainsi la détection d'artefacts encore inconnus.

L'architecture *Transformer* de R.A.I.D. est particulièrement adaptée à cette transition via des techniques de pré-entraînement similaires à celles utilisées dans les modèles de langage (type BERT). En masquant certains octets au sein d'une séquence et en demandant au modèle de les prédire (architecture decoder), le système développerait une compréhension profonde de la "syntaxe" binaire des différents types de données. Dans ce cadre, une erreur de prédiction élevée deviendrait un indicateur natif d'une zone à haute entropie (chiffrement), sans que le modèle ait besoin d'avoir "vu" des données chiffrées durant son entraînement.

L'objectif final de cette perspective est de permettre au modèle de s'adapter dynamiquement à n'importe quel système d'exploitation ou architecture matérielle. En passant au non supervisé, R.A.I.D. ne chercherait plus à valider des catégories pré-définies, mais à segmenter la mémoire en zones de cohérence. Cette approche permettrait d'isoler avec une plus grande précision les zones de données volatiles, posant ainsi les bases d'une assistance automatisée capable de s'auto-ajuster à la spécificité de chaque dump RAM analysé, bien que cela reste hypothétique.

4.4.2 Le défi de la fragmentation et de la reconstruction

Si le projet R.A.I.D. a validé la capacité des *Transformers* à classifier des segments isolés, une étape suivante pourrait consister en la gestion de la fragmentation et la reconstruction logicielle des fichiers. Un dump RAM n'est pas une suite linéaire de fichiers, mais un agglomérat de blocs mémoires souvent éparpillés.

De la segmentation locale à la vision globale

Le modèle actuel travaille sur des fenêtres de 1024octets (chunk_size), ce qui lui permet d'identifier avec précision la nature d'un fragment (image, PDF, texte). Le défi est de déter-

miner si deux segments identifiés comme `IMAGE` et situés à des adresses distantes appartiennent au même objet binaire. L'utilisation du mécanisme d'attention pourrait être étendue pour comparer les "signatures sémantiques" de différents blocs afin de détecter des continuités logiques malgré la fragmentation physique.

R.A.I.D. comme moteur de "File Carving" intelligent

Les outils de *file carving* traditionnels (Scalpel, Foremost ..) reposent sur la détection de structures rigides (en-têtes et pieds de fichiers). R.A.I.D. offre une perspective de "*carving* sémantique" via :

— L'isolation des zones d'intérêt au milieu du bruit ou des données chiffrées, le modèle réduira drastiquement l'espace de recherche pour les algorithmes de reconstruction.

— La reconstruction sans en-tête, car le modèle a prouvé sa capacité à reconnaître des structures (comme des pixels bruts via `DECODED`) même lorsque les métadonnées du fichier original ont disparu de la RAM. Cela permettrait de récupérer des documents partiels là où les méthodes classiques échouent.

Vers une assistance à la reconstruction automatique

À terme, l'intégration de R.A.I.D. dans une chaîne de traitement automatisée permettrait de proposer une prévisualisation des fichiers fragmentés. En couplant la classification par octets avec des techniques de réassemblage basées sur la probabilité de transition entre les types de données, le projet pose les bases d'un outil capable de reconstituer des preuves volatiles (fragments de conversations, images de sessions) qui sont aujourd'hui souvent considérées comme perdues car trop déstructurées.

5 Conclusion

Le projet R.A.I.D. (RAM Artificial Inspection Dump) a permis de démontrer l'efficacité des architectures *Transformer* pour l'analyse automatisée de la mémoire vive, un domaine où la volumétrie et la fragmentation des données rendent les méthodes traditionnelles de moins en moins opérantes. En nous affranchissant des signatures statiques au profit d'une approche basée sur la compréhension statistique des flux binaires, nous avons développé un outil capable de segmenter un dump mémoire avec une précision de 88,79%.

L'implémentation du modèle `BytesTransformerClassifier`, couplée à une couche de convolution locale et à un mécanisme de *self-attention*, a prouvé sa capacité à identifier la nature profonde des données (texte, images, documents) même en l'absence de métadonnées ou d'en-têtes explicites. L'introduction du pipeline de vote pondéré a constitué une étape clé de notre méthodologie, permettant de stabiliser les prédictions et de réduire significativement le bruit de classification inhérent à l'analyse par fenêtres glissantes.

Toutefois, ce projet a également mis en lumière des défis persistants, notamment la convergence statistique entre les données chiffrées et compressées. Cette limite souligne que si l'apprentissage profond excelle dans la reconnaissance de structures ordonnées, la distinction entre différentes formes de haute entropie nécessite encore l'intégration de métriques mathématiques plus fines ou de contextes d'analyse élargis.

En conclusion, R.A.I.D. pose les jalons d'une nouvelle génération d'outils de forensique assistée par intelligence artificielle. Les perspectives ouvertes par ce travail qu'il s'agisse du passage à l'apprentissage non supervisé pour s'adapter dynamiquement à de nouveaux systèmes, ou du développement d'un moteur de *file carving* sémantique suggèrent que l'IA peut transformer radicalement la manière dont les enquêteurs extraient des preuves volatiles d'un système. Ce projet démontre que, face à la complexité croissante des environnements numériques, la capacité des modèles de langage à "lire" le binaire brut constitue un atout majeur pour l'informatique légale de demain.

Bibliographie

[1] Ashish VASWANI et al. « Attention is All you Need ». In : *Advances in Neural Information Processing Systems (NIPS)*. 2017, p. 5998-6008. URL : https://arxiv.org/abs/1706.03762.