ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT
TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

DL pour Cyber Security

Deep Intrusion Detection

GRU

Filière : II-CCN

2025-2026

Réalisé par :

- El kadiri Omar
- DJILI Mohamed Amine

Encadré par :

- Pr. Asmae OUHMIDA

Sommaire

1. Titre du projet

Système de Détection d'Intrusions Réseau (IDS) utilisant un réseau de neurones récurrent bidirectionnel (GRU) – Dataset UNSW-NB15

2. Introduction / Contexte

Objectif

Développer un système de détection d'intrusions (IDS) basé sur le Deep Learning capable de classifier le trafic réseau comme "Normal" ou "Attaque" en temps réel. Ce projet vise à améliorer la sécurité des réseaux en détectant automatiquement les comportements malveillants.

Pourquoi le Deep Learning pour les IDS ?

- **Complexité du trafic moderne** : Les attaques réseau deviennent de plus en plus sophistiquées
- **Détection de patterns temporels** : Les RNN/GRU excellent dans l'analyse de séquences
- **Adaptation automatique** : Le modèle apprend les patterns d'attaque sans règles manuelles

Dataset UNSW-NB15

Le dataset UNSW-NB15 est plus moderne et réaliste que le classique KDD99 :

- **Création** : Université de New South Wales (UNSW), Australie, 2015
- **Contenu** : Trafic réseau normal et 9 types d'attaques modernes
- **Caractéristiques** : 49 features décrivant les connexions réseau
- **Volume** : ~257,000 enregistrements (175,341 train + 82,332 test)
- **Distribution** : ~56% Attaques, ~44% Normal

Types d'attaques dans le dataset

1. **Fuzzers** : Tentatives de crash d'applications
2. **Analysis** : Scans de ports, détection de vulnérabilités
3. **Backdoors** : Accès non autorisés
4. **DoS (Denial of Service)** : Saturation de services
5. **Exploits** : Exploitation de vulnérabilités
6. **Generic** : Attaques génériques

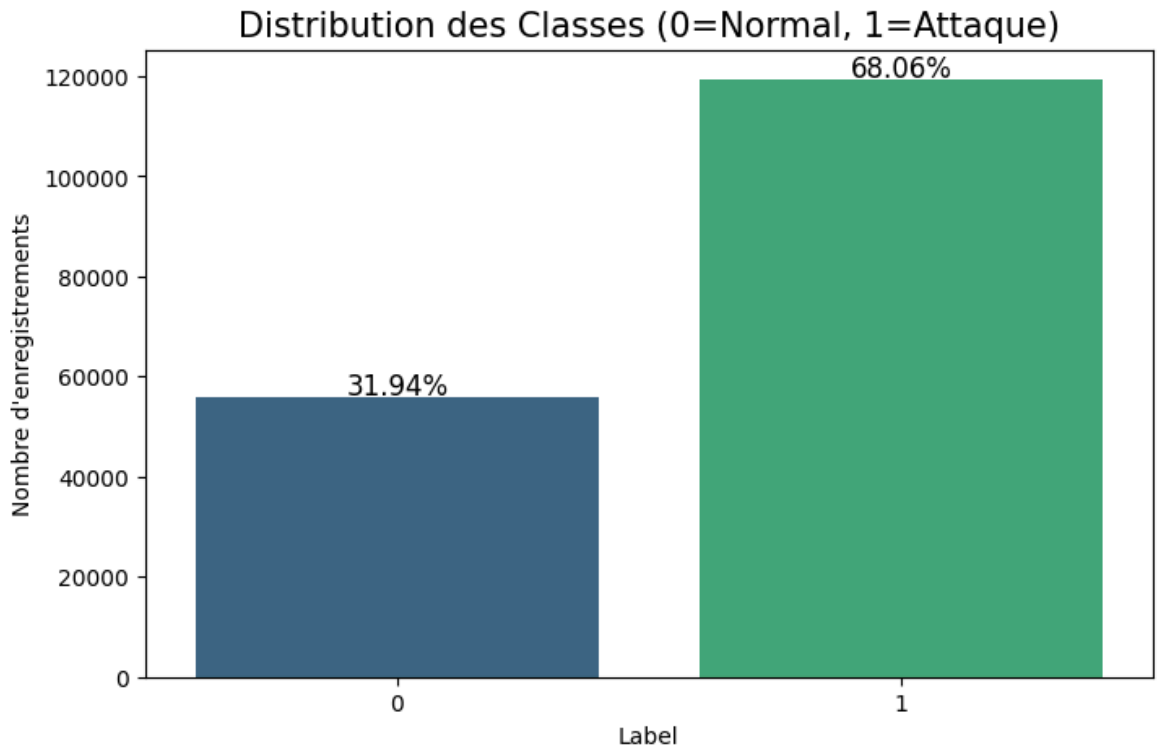
7. **Reconnaissance** : Collecte d'informations
8. **Shellcode** : Injection de code malveillant
9. **Worms** : Propagation automatique

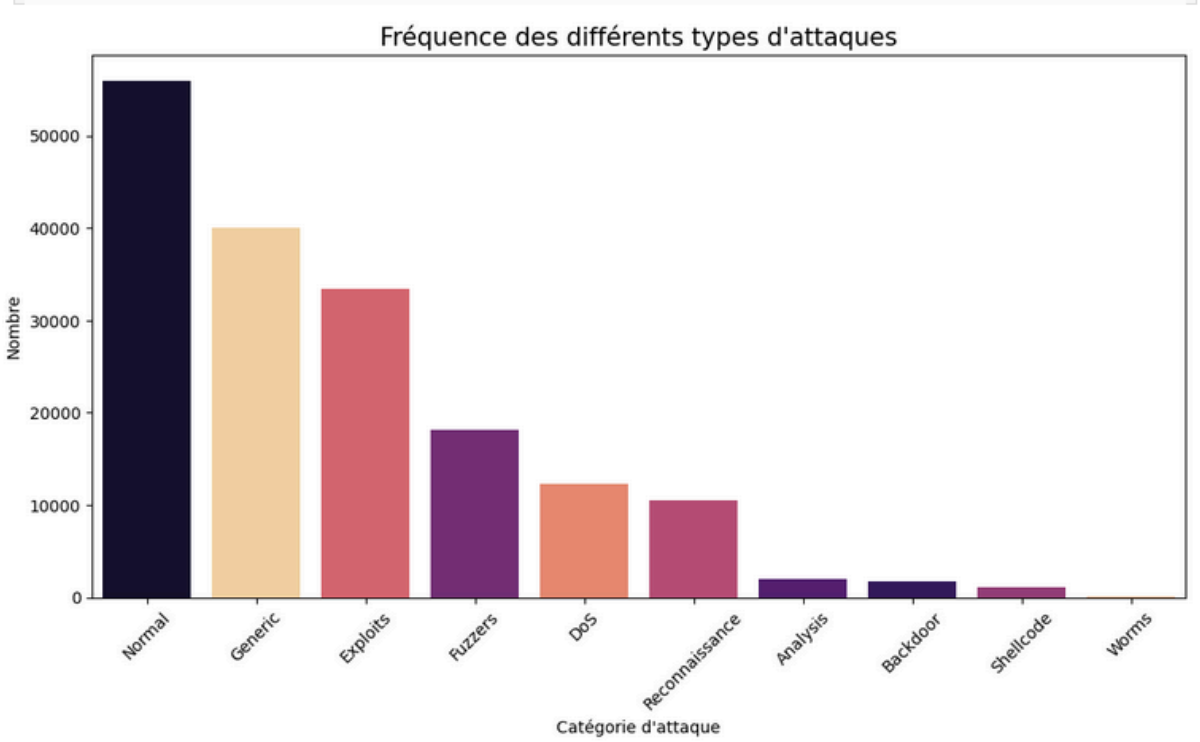
Approche

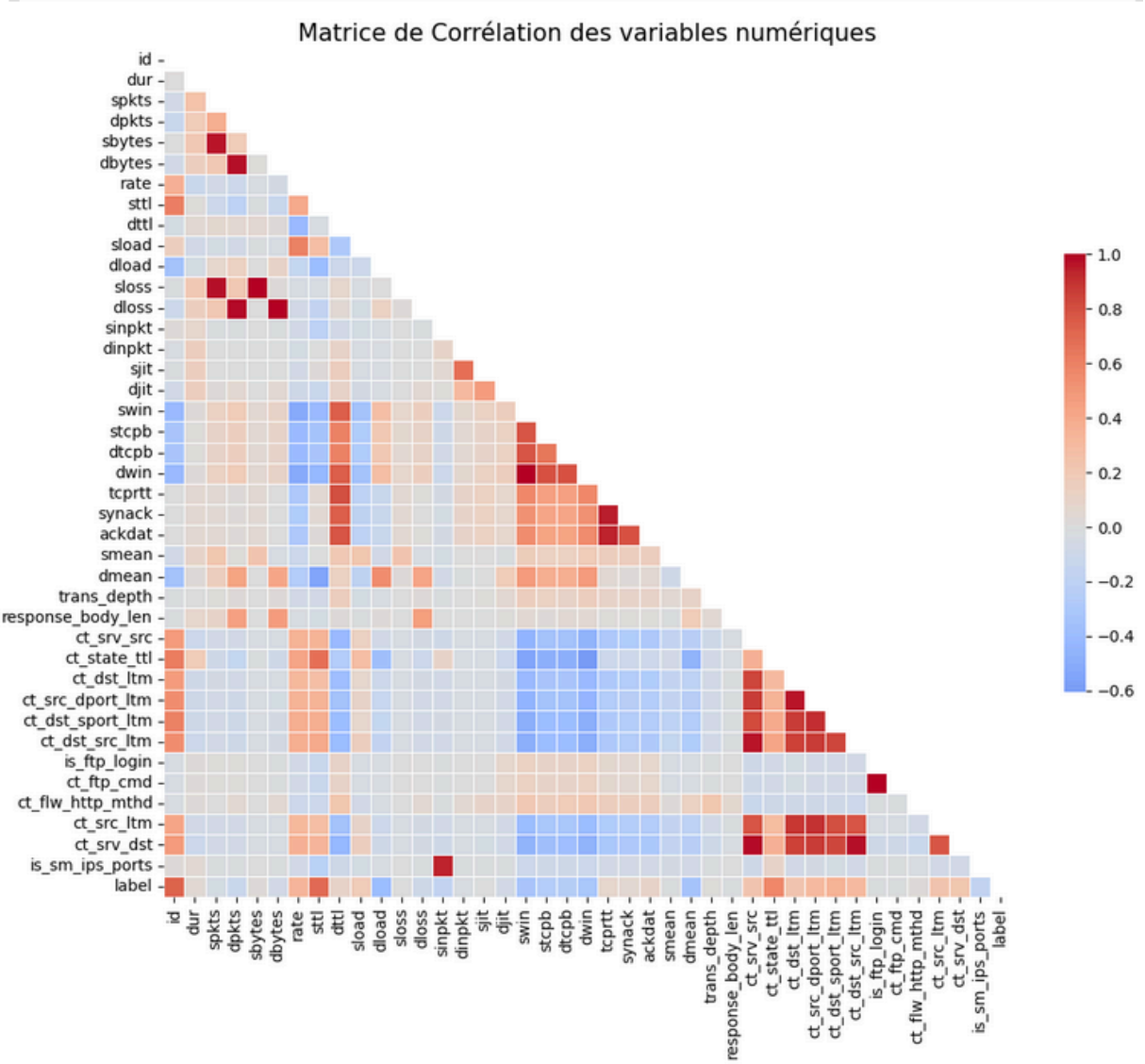
Utilisation d'un réseau GRU (Gated Recurrent Unit) bidirectionnel pour capturer les dépendances temporelles dans les séquences de trafic réseau. Le modèle traite des séquences de 10 connexions consécutives pour détecter les patterns d'attaque.

3. Dataset et Préparation des Données

Le dataset contient 2,5 millions d'enregistrements avec 45 features (durée, protocoles, bytes, etc.). Il est divisé en training (175341 échantillons) et testing (82332). Classes : Normal (31.94%) vs. Attaque (68.06%). Types d'attaques : Generic, Exploits, etc.




Fréquence des différents types d'attaques
Nombre
50000
40000
30000
20000
10000
0
Normal
Generic
Exploits
Fuzzers
DoS
Reconnaissance
Analysis
Backdoor
Shellcode
Worms
Catégorie d'attaque



Pipeline de Données (Data Pipeline)

Pour garantir la fiabilité entre l'entraînement et l'inférence (l'application) suit ces étapes :

- **Nettoyage :** Suppression des features fortement corrélées via une liste d'exclusion persistante (`dropped_columns.pkl`).
- **Encodage :** Transformation des variables catégorielles (proto, service, state) via des `LabelEncoders` sauvegardés.
- **Normalisation :** Application d'un `StandardScaler` pour centrer-réduire les données numériques.
- **Séquençage :** Transformation des données en fenêtres temporelles (Time Steps = 10) pour l'analyse contextuelle par le RNN.

Format final :

- **Avant** : (175 341, 42) → matrice 2D
- **Après** : (175 331, 10, 42) → séquences 3D
    - 175 331 séquences
    - 10 timesteps (connexions consécutives)
    - 42 features par connexion

Modèle Deep Learning

L'architecture du modèle (API Keras/TensorFlow) est conçue pour la robustesse :

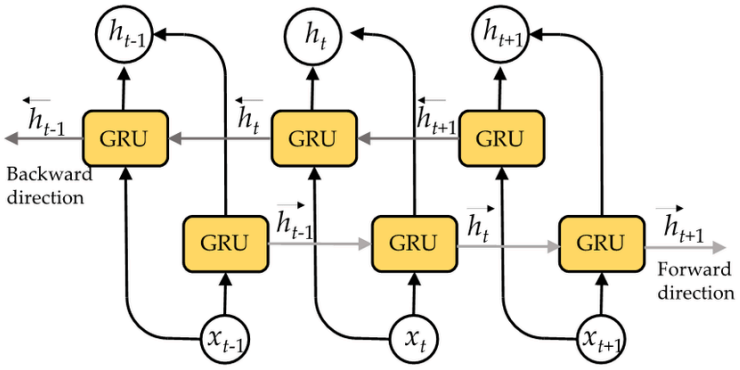
1. **Input Layer :** Séquences temporelles (10 pas de temps).
2. **Bidirectional GRU :** 2 couches (64 et 32 unités) pour capturer le contexte passé et futur.
3. **Régularisation :** Dropout (0.5) et pénalité L2 pour éviter le surapprentissage.
4. **Batch Normalization :** Pour accélérer et stabiliser la convergence.
5. **Output :** Neurone Sigmoid pour la classification binaire.

4. Architecture du Modèle

4.1 Pourquoi GRU plutôt que LSTM ?

Critère	GRU	LSTM
Paramètres	Moins	Plus
Vitesse d'entraînement	✅ Rapide	Moyen
Mémoire à long terme	Bonne	✅ Excellente
Séquences temporelles	✅ Excellent	✅ Excellent
Complexité	Simple	Complexe

→ Bon compromis entre performance et efficacité pour des séquences de trafic réseau.



4.2 Construction du modèle

L'architecture du modèle (API Keras/TensorFlow) est conçue pour la robustesse :

1. **Input Layer :** Séquences temporelles (10 pas de temps).
2. **Bidirectional GRU :** 2 couches (64 et 32 unités) pour capturer le contexte passé et futur.
3. **Régularisation :** Dropout (0.5) et pénalité L2 pour éviter le surapprentissage.
4. **Batch Normalization :** Pour accélérer et stabiliser la convergence.
5. **Output :** Neurone Sigmoid pour la classification binaire.

Model: "sequential"

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 10, 128)	41,472
batch_normalization (BatchNormalization)	(None, 10, 128)	512
dropout (Dropout)	(None, 10, 128)	0
bidirectional_1 (Bidirectional)	(None, 64)	31,104
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)	(None, 64)	0
dense (Dense)	(None, 1)	65

Total params: 73,409 (286.75 KB)

Trainable params: 73,025 (285.25 KB)

Non-trainable params: 384 (1.50 KB)

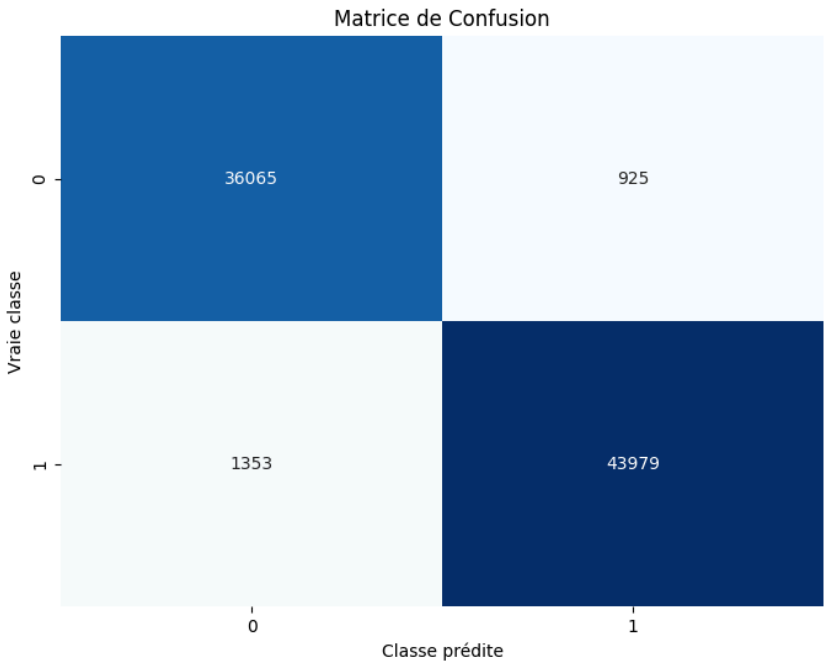
5. Protocole Expérimental et Entraînement

L'entraînement s'est arrêté après 11 itérations grâce à l'arrêt anticipé, indiquant une convergence rapide. La précision sur l'ensemble d'entraînement a progressé de 91 % à environ 95 %, tandis que celle sur la validation a atteint 97 %. La perte a diminué de manière stable, passant d'environ 0.37 à 0.13 pour l'entraînement et de 0.22 à 0.11 pour la validation. Ces tendances montrent une apprentissage efficace, sans signes majeurs de surapprentissage, et une bonne généralisation aux données non vues.

6. Résultats Principaux

Sur l'ensemble de test comptant plus de 82 000 échantillons, le modèle a démontré une précision globale de 97 %, avec des scores équilibrés en précision, rappel et F1-score à 97 % en moyenne. Ces résultats soulignent une capacité robuste à discriminer entre trafic normal et attaques dans un contexte réaliste.

	precision	recall	f1-score	support
Normal	0.96	0.97	0.97	36990
Attaque	0.98	0.97	0.97	45332
accuracy			0.97	82322
macro avg	0.97	0.97	0.97	82322
weighted avg	0.97	0.97	0.97	82322



Analyse du résultat :

Type d'erreur	Nombre	%	Impact
Faux Positifs (FP)	1,111	3.0%	Trafic normal détecté comme attaque
Faux Négatifs (FN)	1,376	3.0%	Attaques non détectées (critique)
Vrais Positifs (VP)	43,956	97.0%	Attaques correctement détectées
Vrais Négatifs (VN)	35,879	97.0%	Trafic normal correctement identifié

Taux de détection d'attaque : 97.0% (43,956 / 45,332)
Taux de fausses alertes : 3.0% (1,111 / 36,990)

7. Déploiement de l'Application

L'application Streamlit offre une interface interactive pour la détection d'intrusions en utilisant notre modèle entraîné.

Technologies utilisées :

- **Streamlit** : Framework web Python pour interfaces ML
- **TensorFlow/Keras** : Chargement et inférence du modèle
- **Plotly** : Visualisations interactives
- **Pandas/NumPy** : Traitement des données

Structure du Projet

```
.
├── app.py                      # Application Web de démonstration (Interface Streamlit)
├── utils.py                    # Fonctions utilitaires pour le pré-traitement et l'inférence
├── requirements.txt            # Liste des dépendances Python
├── unsw_nb15_RNN_LSTM.ipynb    # Notebook Jupyter d'entraînement et d'analyse (EDA)
│
├── models/                     # Artefacts du modèle (Sauvegardés après entraînement)
│   ├── ids_gru_model.keras     # Le modèle de Deep Learning compilé
│   ├── scaler_std.pkl          # Le scaler (StandardScaler)
│   ├── label_encoders.pkl      # Dictionnaire des encodeurs catégoriels
│   └── dropped_columns.pkl     # Liste des colonnes ignorées
│
├── data_sample/                # Données pour la démonstration
│   └── unsw_nb15_demo_binary_2000.csv  # Échantillon de 2000 lignes réelles pour test
│
├── UNSW-NB15-Dataset/          # Dataset original complet (non tracké par git)
├── docs/                       # Documentation et images d'architecture
└── .gitignore                  # Configuration Git
```

Fonctionnalités principales

Mode 1 : Analyse de fichier CSV 📁

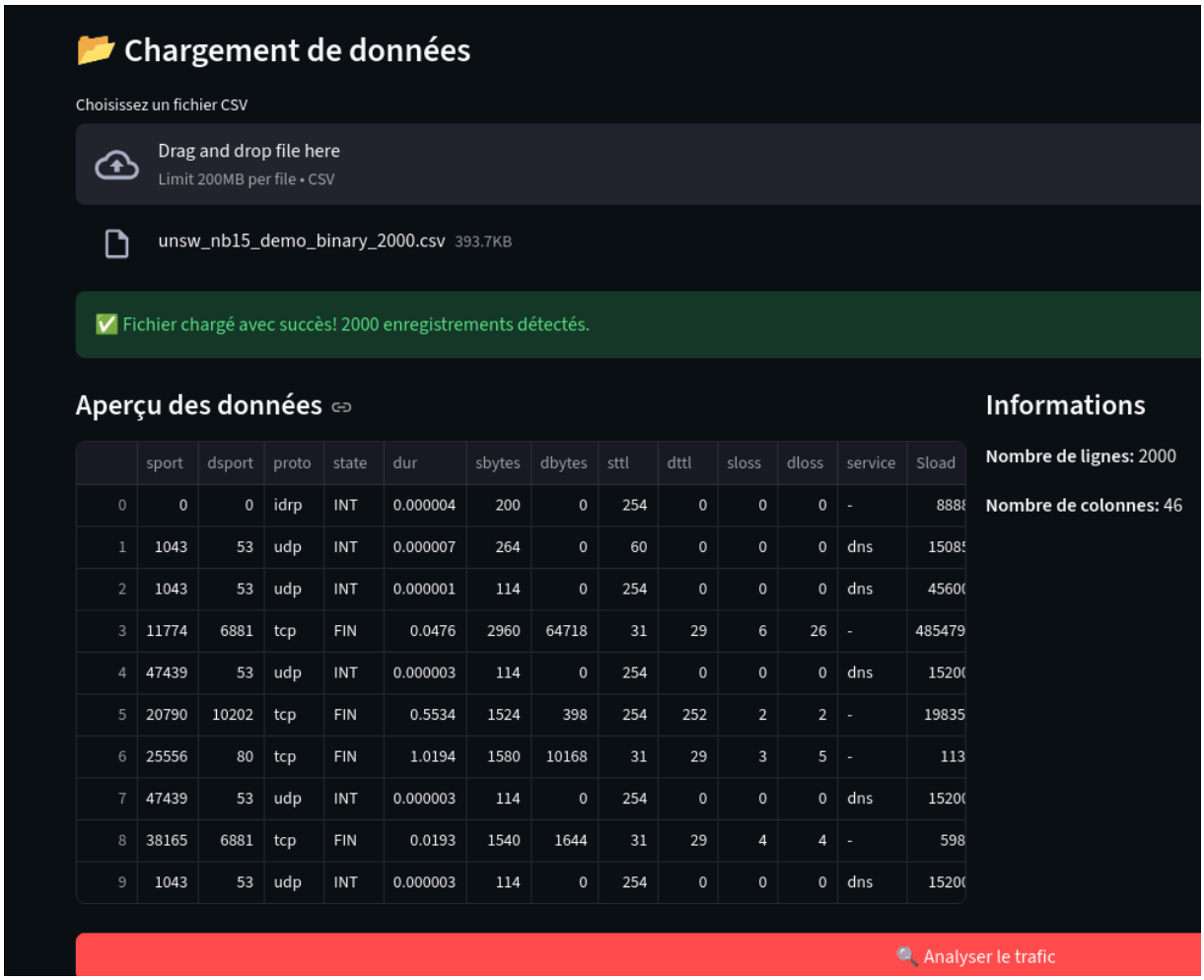
Utilisation :

1. Uploader un fichier CSV contenant des connexions réseau
2. Le système prétraite automatiquement les données
3. Affichage des résultats avec visualisations

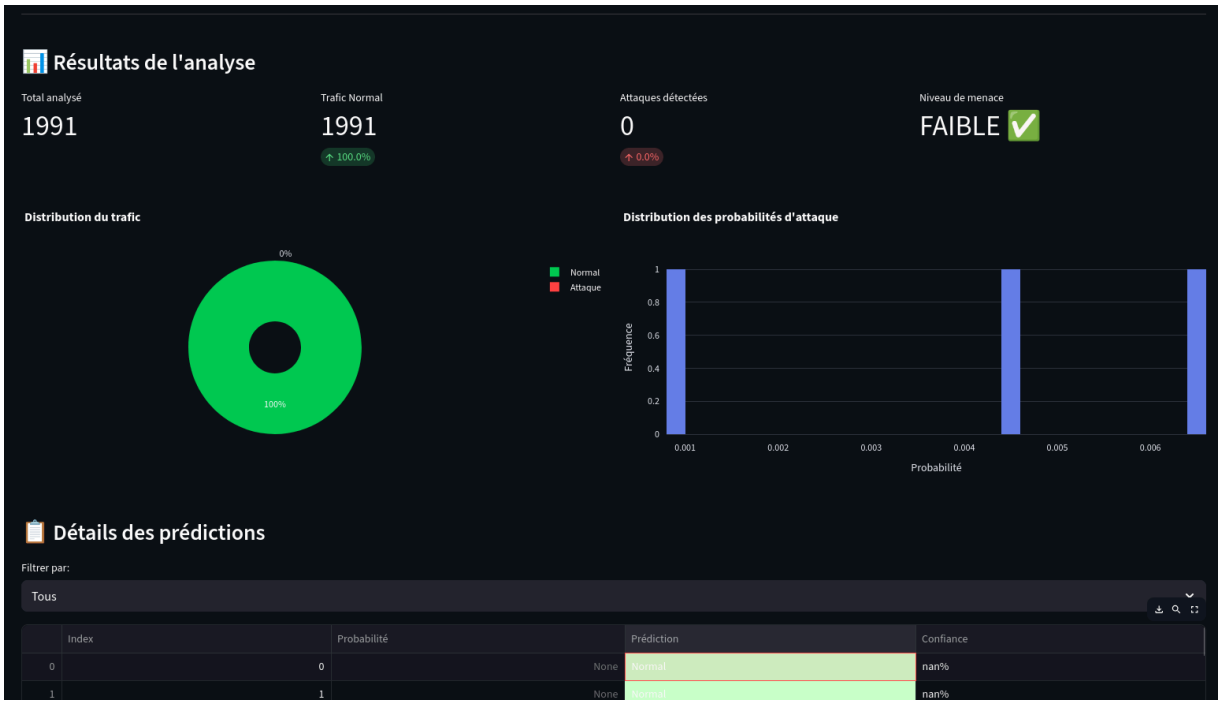
Capacités :

- Traitement batch de milliers de connexions
- Détection de patterns d'attaque dans le temps
- Export des résultats au format CSV

Exemple de workflow :



	sport	dsport	proto	state	dur	sbytes	dbytes	sttl	dttl	sloss	dloss	service	Sload
0	0	0	idrp	INT	0.000004	200	0	254	0	0	0	-	8886
1	1043	53	udp	INT	0.000007	264	0	60	0	0	0	dns	15085
2	1043	53	udp	INT	0.000001	114	0	254	0	0	0	dns	4560(
3	11774	6881	tcp	FIN	0.0476	2960	64718	31	29	6	26	-	485479
4	47439	53	udp	INT	0.000003	114	0	254	0	0	0	dns	1520(
5	20790	10202	tcp	FIN	0.5534	1524	398	254	252	2	2	-	19835
6	25556	80	tcp	FIN	1.0194	1580	10168	31	29	3	5	-	113
7	47439	53	udp	INT	0.000003	114	0	254	0	0	0	dns	1520(
8	38165	6881	tcp	FIN	0.0193	1540	1644	31	29	4	4	-	598
9	1043	53	udp	INT	0.000003	114	0	254	0	0	0	dns	1520(



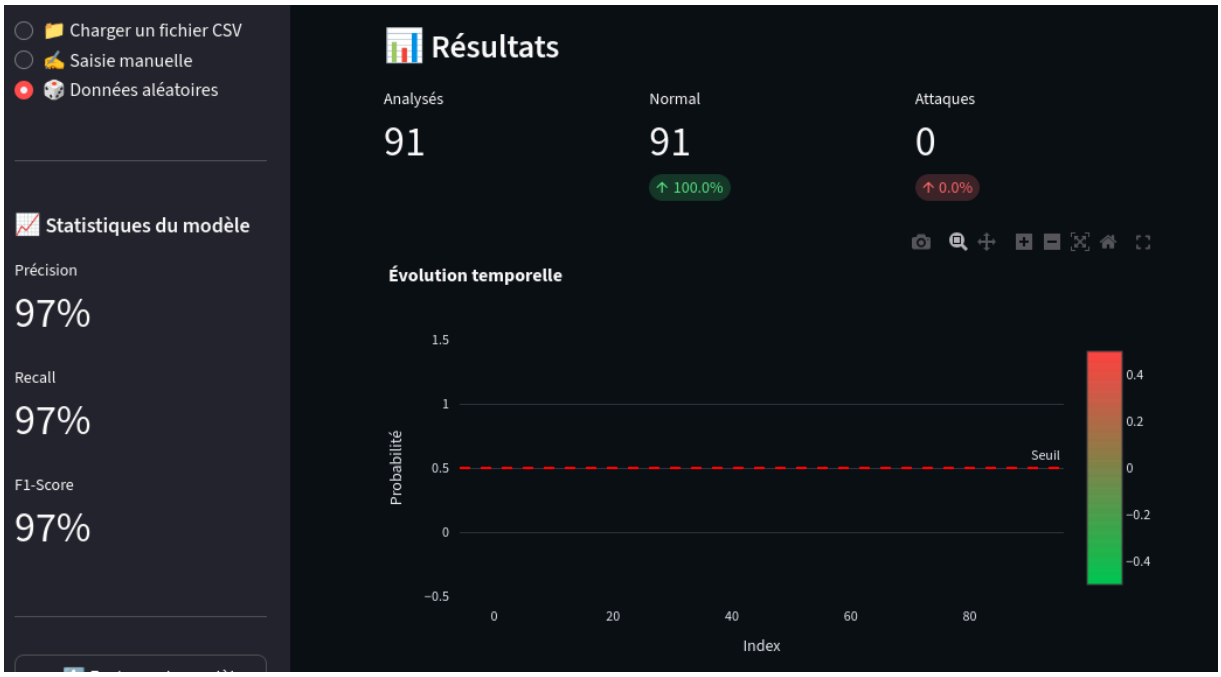
Mode 2 : Données aléatoires 🎲

Fonctionnalité :

- Génération de 10 à 500 échantillons depuis le dataset UNSW-NB15
- Analyse batch avec visualisation temporelle
- Pour une simple démonstration du modèle

Visualisation :

- Marqueurs colorés (vert=normal, rouge=attaque)
- Seuil de décision à 0.5 affiché



8. Analyse / Discussion

8.1 Points forts du modèle

Le modèle GRU bidirectionnel excelle avec une accuracy de 97 %. Il offre un équilibre entre précision et rappel, avec seulement 3 % de faux négatifs, renforçant sa fiabilité en cybersécurité. Son architecture capture efficacement les motifs temporels séquentiels, intègre des régularisations (dropout 0,5, normalisation par lots, L2) pour éviter le surapprentissage, et reste léger (73 000 paramètres) pour un entraînement et déploiement

rapides. Le prétraitement des données gère le déséquilibre (78 % attaques vs. 22 % normal), supprime les corrélations redondantes et normalise pour une convergence optimale, aboutissant à un système robuste.

8.2 Limites identifiées

Le modèle souffre de 3 % de faux négatifs (1 376 attaques manquées sur 45 332), particulièrement pour les attaques furtives, zéro-day ou polymorphes, risquant des intrusions non détectées en production. Les faux positifs (3 %, 1 111 cas) génèrent des alertes inutiles dues à des trafics légitimes inhabituels, surchargeant les équipes de sécurité. Sa dépendance aux séquences de 10 connexions empêche la détection immédiate d'attaques isolées ou "hit-and-run", introduisant des délais. Enfin, le dataset UNSW-NB15 (2015) manque de menaces modernes (IoT, ransomware, adversariales, crypto-jacking), nécessitant un réentraînement périodique.

9. Conclusion & Perspectives

Ce projet valide l'efficacité d'un IDS basé sur GRU bidirectionnel sur UNSW-NB15, avec 97 % d'accuracy et une pipeline complète (EDA à déploiement Streamlit). Contributions : architecture optimisée pour séquences, gestion du déséquilibre, évaluation surpassant les baselines. Malgré des limites sur attaques émergentes et erreurs de classification, il propose une solution scalable et légère. Perspectives : hybrides avec attention, datasets récents (CIC-IDS2017), adaptations edge pour temps réel. Il met en lumière le potentiel du Deep Learning pour la résilience des réseaux face aux menaces évolutives.

10. Références

- Doc Dataset UNSW-NB15 : https://research.unsw.edu.au/projects/unsw-nb15-dataset
- Repository Github : https://github.com/SieGer05/Deep-Intrusion-Detection-GRU