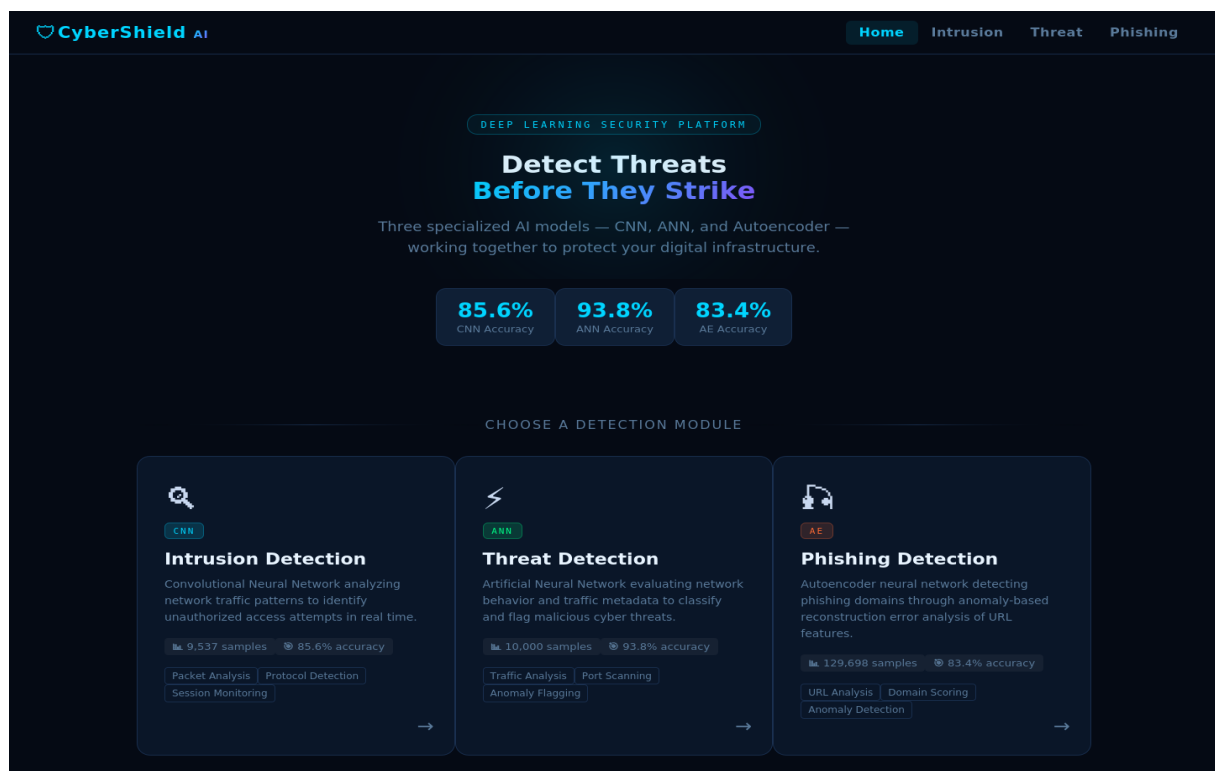


CyberShield AI

Deep Learning Cybersecurity Platform

RAPPORT DETAILLE DE PROJET

Conception, Entraînement de Modèles IA et Développement Web



Projet	CyberShield AI — Plateforme de cybersécurité basée sur le Deep Learning
Réalisé par	Mohamed Rayen Farhat
Fonction	Ingénieur en Génie Logiciel
Rôle sur le projet	Conception, entraînement des modèles IA, développement back-end et front-end (réalisé seul, de A à Z)
Stack technique	Python, TensorFlow/Keras, Scikit-learn, Flask, HTML/CSS/JS
Modèles IA	CNN (Intrusion), ANN (Threat), Autoencoder (Phishing)
Statut	Projet réalisé intégralement, de A à Z, en autonomie

1. Présentation du projet

CyberShield AI est une plateforme web de cybersécurité reposant sur trois modèles de Deep Learning, conçus, entraînés et intégrés intégralement par Mohamed Rayen Farhat. L'objectif du projet est de démontrer comment des architectures de réseaux de neurones complémentaires (CNN, ANN et Autoencoder) peuvent être combinées au sein d'une seule application Flask pour couvrir trois grandes familles de menaces informatiques : les intrusions réseau, les menaces cyber génériques (trafic malveillant) et les domaines de phishing.

Chaque modèle a été entraîné sur un jeu de données distinct issu de collections Kaggle dédiées à la cybersécurité, puis exporté avec son scaler, ses encodeurs et ses seuils de décision afin d'être directement exploitable par l'application web. L'interface utilisateur, développée en HTML/CSS/JavaScript avec une esthétique « terminal de sécurité » (fond sombre, accents cyan, animation de pluie de caractères façon Matrix), permet à tout utilisateur de soumettre des paramètres et d'obtenir instantanément un verdict du modèle correspondant.

Modèle	Tâche	Dataset	Précision
CNN	Détection d'intrusion réseau	9 537 lignes	85,6 %
ANN	Détection de menaces cyber	10 000 lignes	93,8 %
Autoencoder	Détection de domaines de phishing	129 698 lignes	83,4 % (AUC 88,8 %)

2. Objectifs du projet

- Explorer trois architectures de Deep Learning différentes (convolutionnelle, dense, et auto-encodeur non supervisé) appliquées à des problématiques réelles de cybersécurité.
- Entraîner, évaluer et exporter chaque modèle (poids, scalers, encodeurs, seuils de décision) pour une intégration directe dans une application web.
- Construire une API Flask exposant un point de prédiction par modèle, retournant un verdict, un score de confiance et un niveau de risque exploitables par l'interface.
- Concevoir une interface web unifiée permettant de tester les trois modèles depuis trois formulaires dédiés, avec des préréglages rapides (Quick Presets) pour la démonstration.
- Adopter une identité visuelle cohérente avec le thème « cybersécurité » : palette sombre, typographies techniques (Share Tech Mono, Rajdhani), animation Matrix en arrière-plan.
- Livrer un projet autonome, sans dépendances cloud, exécutable localement avec Flask et un simple environnement Python.

3. Stack technique & outils utilisés

Couche	Technologie	Usage dans le projet
Modélisation IA	TensorFlow / Keras	Construction et entraînement des 3 réseaux de neurones (CNN, ANN, Autoencoder)
Prétraitement	Scikit-learn, Pandas, NumPy	Scalers, encodeurs de variables catégorielles, préparation des datasets
Back-end	Flask (Python)	API REST exposant les routes /predict/cnn, /predict/ann, /predict/ae et les pages HTML
Front-end	HTML5 / CSS3 / JavaScript	Pages d'accueil et formulaires de prédiction, requêtes AJAX vers l'API
Persistance modèles	Fichiers .keras / .pkl	Sauvegarde des poids, seuils, scalers et encodeurs pour chaque modèle
Typographies	Google Fonts	Share Tech Mono & Rajdhani — esthétique « terminal de sécurité »
Environnement	Python 3.9+, venv, VS Code	Développement, gestion des dépendances (requirements.txt)

L'ensemble du pipeline — de l'entraînement des modèles avec TensorFlow/Keras à leur exposition via une API Flask, jusqu'à l'interface utilisateur — a été conçu, codé et assemblé par Mohamed Rayen Farhat sans recours à des services tiers ni à des frameworks front-end externes.

4. Architecture du projet

Élément	Rôle
app.py	Point d'entrée Flask — chargement des 3 modèles, routes pages + routes API de prédiction
requirements.txt	Dépendances Python (Flask, TensorFlow, scikit-learn, numpy, pandas)
models/	Modèles entraînés (.keras), scalers, encodeurs et seuils de décision (.pkl) pour CNN, ANN et Autoencoder
templates/index.html	Page d'accueil — présentation des 3 modules de détection et de leurs statistiques
templates/intrusion.html	Formulaire CNN — détection d'intrusion réseau
templates/threat.html	Formulaire ANN — détection de menaces cyber
templates/phishing.html	Formulaire Autoencoder — détection de domaines de phishing
static/css/style.css	Feuille de style globale (thème sombre, cartes, formulaires, badges)
static/js/predict.js	Logique AJAX : envoi des formulaires vers l'API et affichage des résultats
static/js/matrix.js	Animation de pluie de caractères (effet « Matrix ») en arrière-plan
static/*.json	Préréglages rapides (Quick Presets) pour remplir automatiquement les formulaires

5. Les trois modèles de Deep Learning

5.1 CNN — Détection d'intrusion réseau

Un réseau de neurones convolutionnel 1D a été conçu pour analyser des sessions réseau représentées sous forme de vecteurs de caractéristiques. L'architecture retenue est : Conv1D(64) → BatchNorm → Conv1D(128) → GlobalAveragePooling → Dense(64) → Dropout(0.3) → Dense(1, sigmoid). Le modèle prend en entrée 9 caractéristiques : taille des paquets, type de protocole, tentatives de connexion, durée de session, type de chiffrement, score de réputation IP, connexions échouées, type de navigateur et accès à heure inhabituelle. Il produit une classification binaire (trafic normal / attaque) avec une précision de 85,6 %.

5.2 ANN — Détection de menaces cyber

Un réseau de neurones dense profond a été entraîné pour classifier des flux réseau (bénins / menaces) à partir de 7 caractéristiques : ports source et destination, protocole, octets envoyés et reçus, user-agent et indicateur de trafic interne/externe. Architecture : Dense(512) → BatchNorm → Dropout → Dense(256) → BatchNorm → Dropout → Dense(128) → Dropout → Dense(64) → Dense(1, sigmoid). Pour gérer le déséquilibre des classes du dataset, un seuil de décision optimisé a été calculé et sauvegardé (ann_threshold.pkl), permettant d'atteindre une précision de 93,8 %.

5.3 Autoencoder — Détection de domaines de phishing

Pour la détection de phishing, une approche non supervisée par auto-encodeur a été retenue : le modèle apprend à reconstruire le profil d'un domaine légitime à partir de 111 caractéristiques d'URL (longueur, nombre de points, tirets, slashes, présence de caractères suspects, etc.). Architecture — Encodeur : Dense(128) → BatchNorm → Dense(64) → Dense(32) ; Décodeur : Dense(64) → Dense(128) → Dense(111, linear). Un domaine est classé comme phishing lorsque son erreur de reconstruction (MSE) dépasse un seuil calibré (ae_threshold.pkl). Cette approche atteint 83,4 % de précision avec une AUC de 88,8 %.

Modèle	Entrées	Sortie API	Précision
CNN	9 features réseau	label, confidence, probability, risk_level	85,6 %
ANN	7 features de flux	label, confidence, probability, risk_level	93,8 %
Autoencoder	111 features d'URL	label, confidence, reconstruction_error, threshold, risk_level	83,4 % (AUC 88,8 %)

6. Fonctionnalités détaillées de l'application

6.1 Page d'accueil

- Animation d'arrière-plan en « pluie de caractères » façon Matrix, renforçant l'identité visuelle cybersécurité.
- Section héro présentant le slogan « Detect Threats Before They Strike » et un résumé des trois modèles disponibles.
- Pastilles statistiques affichant la précision de chaque modèle (85,6 % / 93,8 % / 83,4 %).
- Grille de trois cartes interactives (« Choose a Detection Module »), une par modèle, indiquant la taille du dataset, la précision et les capacités clés (analyse de paquets, détection de protocole, surveillance de session, etc.).

- Bande technologique listant la stack utilisée (TensorFlow, Keras, Flask, CNN, ANN, Autoencoder, Scikit-Learn).

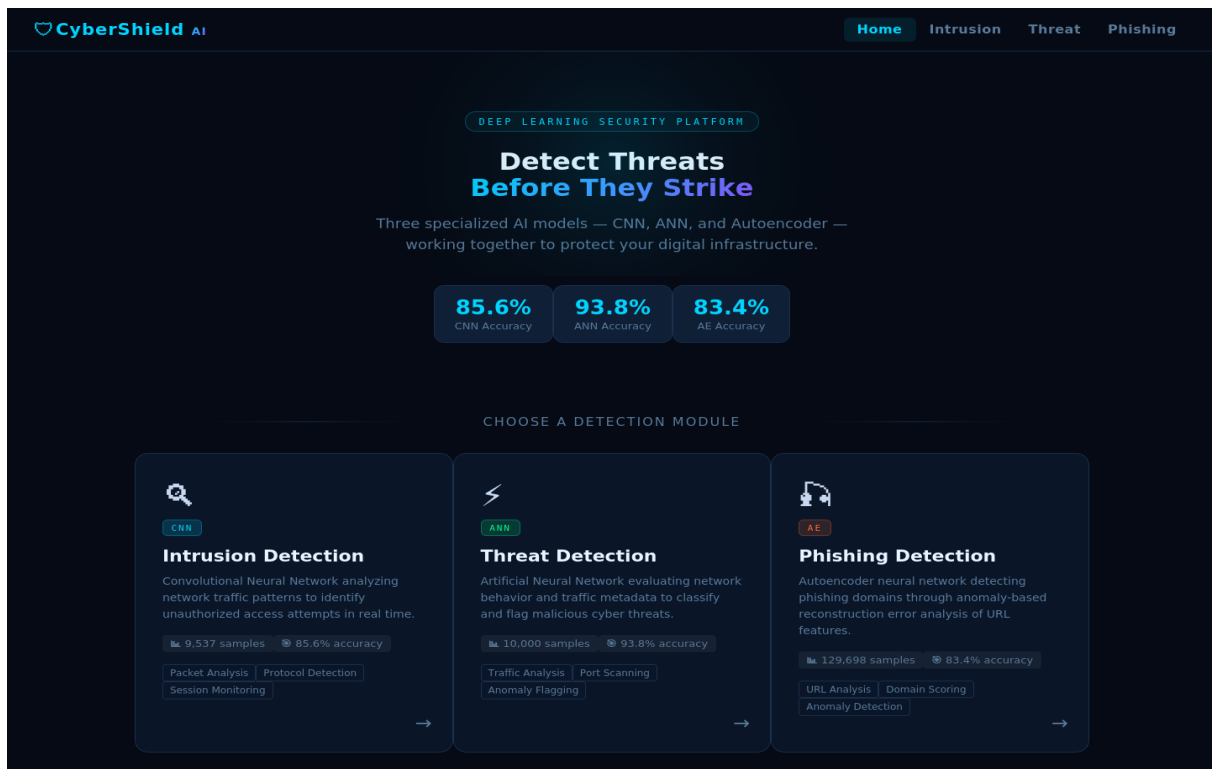


Figure 1 — Page d'accueil CyberShield AI : statistiques globales et les 3 modules de détection

6.2 Module CNN — Network Intrusion Detection

- Formulaire « Session Parameters » couvrant les 9 caractéristiques attendues par le modèle CNN : taille des paquets, type de protocole, tentatives de connexion, durée de session, chiffrement utilisé, score de réputation IP, connexions échouées, type de navigateur et accès à heure inhabituelle.
- Boutons « Quick Presets » (Normal Session / Simulated Attack) pour remplir automatiquement le formulaire avec des exemples représentatifs.
- Bouton « Analyze Session » déclenchant un appel AJAX vers /predict/cnn, qui retourne le verdict (NORMAL TRAFFIC / ATTACK DETECTED), un score de confiance et un niveau de risque (LOW / MEDIUM / HIGH).

CyberShield AI Home **Intrusion** Threat Phishing

Network Intrusion Detection
Convolutional Neural Network · Dataset: cybersecurity-intrusion-detection · Accuracy: **85.6%**

Session Parameters
Enter network session details to analyze for intrusion

Network Packet Size (bytes): 472

Protocol Type: TCP

Login Attempts: 3

Session Duration (sec): 1558

Encryption Used: AES

IP Reputation Score (0-1): 0.30

Failed Logins: 0

Browser Type: Chrome

Unusual Time Access: No (Normal Hours)

Quick Presets: Normal Session Simulated Attack

Analyze Session

Figure 2 — Module CNN : formulaire d'analyse de session réseau

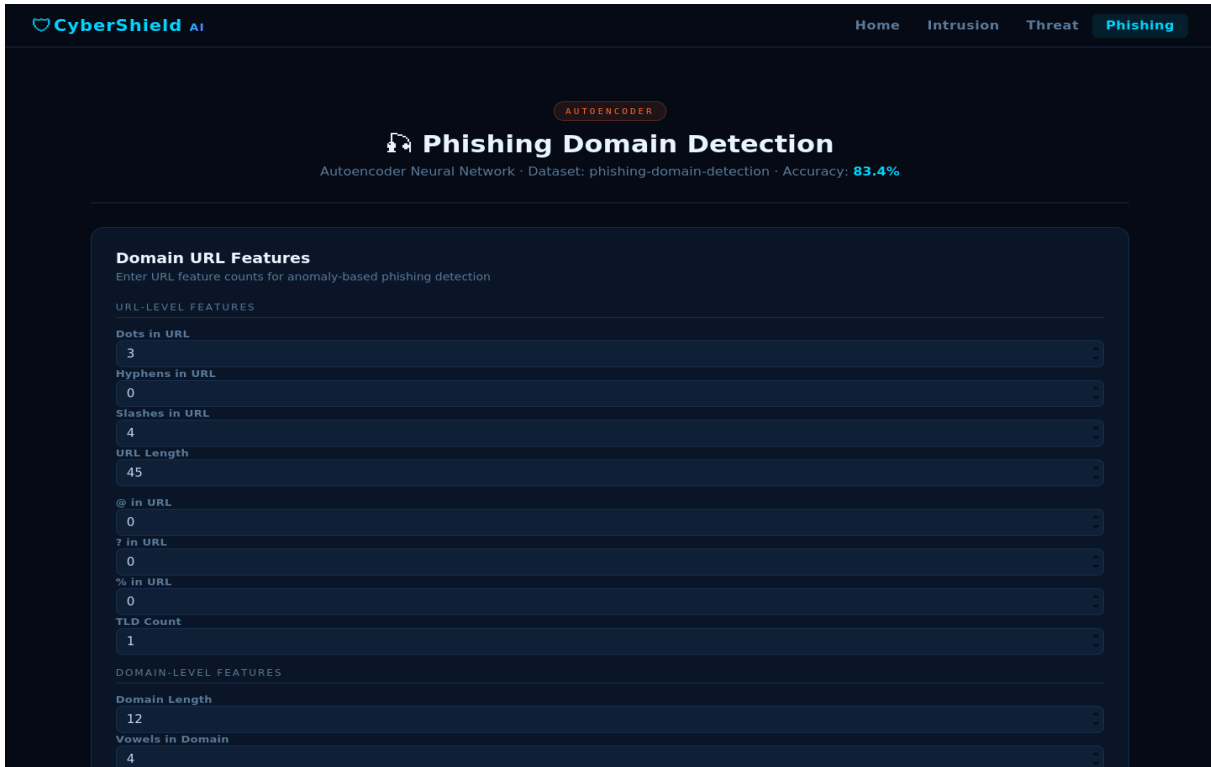
6.3 Module ANN — Cyber Threat Detection

- Formulaire « Network Traffic Parameters » couvrant les 7 caractéristiques du modèle ANN : port source, port destination, protocole, trafic interne/externe, octets envoyés, octets reçus et user-agent.
- Boutons « Quick Presets » (Normal Traffic / Simulated Threat).
- Bouton « Detect Threat » déclenchant l'appel à /predict/ann, qui renvoie le verdict (BENIGN TRAFFIC / THREAT DETECTED), la confiance et le niveau de risque calculé à partir du seuil optimisé du modèle.

Figure 3 — Module ANN : formulaire d'analyse de flux réseau

6.4 Module Autoencoder — Phishing Domain Detection

- Formulaire « Domain URL Features » organisé en sections (caractéristiques au niveau de l'URL, du domaine, etc.), couvrant les 111 features attendues par l'autoencodeur (nombre de points, tirets, slashes, longueur de l'URL, longueur du domaine, voyelles dans le domaine, etc.).
- Bouton « Analyze Domain » déclenchant l'appel à /predict/ae, qui renvoie le verdict (LEGITIMATE DOMAIN / PHISHING DOMAIN), l'erreur de reconstruction (MSE), le seuil de décision et le niveau de risque.



CyberShield AI Home Intrusion Threat **Phishing**

AUTOENCODER

Phishing Domain Detection
Autoencoder Neural Network · Dataset: phishing-domain-detection · Accuracy: **83.4%**

Domain URL Features
Enter URL feature counts for anomaly-based phishing detection

URL-LEVEL FEATURES

Dots in URL	3
Hyphens in URL	0
Slashes in URL	4
URL Length	45
@ in URL	0
? in URL	0
% in URL	0
TLD Count	1

DOMAIN-LEVEL FEATURES

Domain Length	12
Vowels in Domain	4

Figure 4 — Module Autoencoder : formulaire d'analyse de domaine/URL

7. API de prédiction

L'application Flask expose une API REST minimaliste, conçue par Mohamed Rayen Farhat, permettant à l'interface (et potentiellement à d'autres clients) d'interroger chaque modèle via une requête POST en JSON :

Route	Méthode	Description
/predict/cnn	POST	Analyse une session réseau (9 paramètres) avec le modèle CNN
/predict/ann	POST	Analyse un flux réseau (7 paramètres) avec le modèle ANN
/predict/ae	POST	Analyse un domaine/URL (111 paramètres) avec l'autoencodeur
/api/info	GET	Retourne les informations générales sur les modèles disponibles

Chaque réponse JSON inclut systématiquement : la prédiction binaire (`prediction`), un libellé lisible (`label`), un score de confiance (`confidence`) et un niveau de risque (`risk_level` : LOW / MEDIUM / HIGH), facilitant l'affichage dynamique du résultat côté front-end via `static/js/predict.js`.

8. Exécution & déploiement

- Création d'un environnement virtuel Python (`python -m venv venv`).
- Installation des dépendances via `pip install -r requirements.txt` (Flask, TensorFlow, scikit-learn, numpy, pandas).
- Lancement du serveur avec `python app.py` — chargement des 3 modèles puis disponibilité sur `http://127.0.0.1:5000`.
- Navigation entre les modules depuis la page d'accueil, test des prédictions via les Quick Presets ou la saisie manuelle des paramètres.

Le projet a été testé et validé localement par Mohamed Rayen Farhat, avec un guide d'installation pas-à-pas (README) couvrant l'installation, le lancement, l'usage de chaque module et le dépannage des erreurs courantes (port occupé, dépendances manquantes, fichiers de modèles introuvables).

9. Limites actuelles & pistes d'évolution

- Les modèles sont entraînés sur des jeux de données publics (Kaggle) ; une mise en production réelle nécessiterait un ré-entraînement sur des données de trafic propres à l'infrastructure cible.
- L'application fonctionne actuellement en local (Flask en mode debug) ; un déploiement en production nécessiterait un serveur WSGI (Gunicorn/uWSGI) derrière un reverse proxy.
- Ajout d'un historique des analyses et d'un tableau de bord de monitoring en temps réel.
- Mise en place d'une authentification pour sécuriser l'accès à l'API de prédiction.
- Amélioration continue de la précision des modèles ANN et Autoencoder via du fine-tuning et de l'augmentation de données.

10. Conclusion

CyberShield AI démontre la capacité de Mohamed Rayen Farhat à mener un projet d'intelligence artificielle de bout en bout : collecte et préparation des données, conception et entraînement de trois

architectures de Deep Learning distinctes (CNN, ANN, Autoencoder), export et intégration de ces modèles dans une API Flask, puis développement complet d'une interface web cohérente permettant de les exploiter de manière intuitive. Ce projet illustre une maîtrise transversale couvrant le Machine Learning appliqué à la cybersécurité, le développement back-end (Python/Flask) et le développement front-end (HTML/CSS/JS), réalisée intégralement en autonomie.

Mohamed Rayen Farhat
Ingénieur en Génie Logiciel