



Intel® Unnati
Data-Centric Labs in Emerging Technologies



A PROJECT REPORT ON
AI/ML for Network Security

Submitted By

M.SAI KOUSHIK

237Y1A66G2

V.LAKSHMI HARSHITHA

237Y1A66E2

A.SANDEEP

237Y1A66B0

K.ROHITH

237Y1A6642

COMPUTER SCIENCE AND ENGINEERING (AI/ML)

Marri Laxman Reddy Institute of Technology and Management

Under the Guidance of
B.Swapna, Assistant Professor

Submitted to

Intel® Unnati Program

Supported by

EduGate Technologies

In partnership with

Marri Laxman Reddy Institute of Technology and Management (MLRITM)



Date of Submission: July 2025

INDEX

| S.NO | TABLE OF CONTENTS | PAGE NO. |
|------|--------------------------------|----------|
| 1. | AGENDA | 1 |
| 2. | ABSTRACT | 2 |
| 3 | INTRODUCTION | 3 |
| 4 | ANALYSIS | 4 |
| | NETWORK ANALYSIS | 4 |
| | DEEP PACKET INSPECTION | 5 |
| | ROLE OF AI IN NETWORK ANALYSIS | 6-7 |
| 6 | TECHNOLOGIES USED | 8 |
| 7 | IMPLEMENTATION | 9-12 |
| 8 | OUTCOMES | 13-14 |
| 9 | REFERENCES | 15 |
| 10 | CONCLUSION | 16 |

AGENDA

- Automated Network Traffic Analysis using AI/ML models
- Improved Threat Detection & Security - identifying anomalies
- Reduced False Positives & False Negatives, enhancing the efficiency of network security operations
- Scalability & Performance Optimization, ensuring AI models can handle high-traffic environments with minimal latency

ABSTRACT

In an increasingly digital and connected world, networks form the core infrastructure enabling communication, commerce, education, governance, and countless other essential services. As the volume, variety, and velocity of data traversing these networks increase exponentially, so does the complexity and sophistication of the threats and performance bottlenecks they face. Traditional methods of network analysis, primarily rule-based or signature-based systems, are insufficient in handling modern cyberattacks and performance management challenges that evolve dynamically. Consequently, there is a pressing need for intelligent, adaptive, and scalable solutions that provide deep visibility into network traffic while also offering real-time threat detection and mitigation capabilities.

This project explores the integration of **Artificial Intelligence (AI)** and **Deep Packet Inspection (DPI)** as a next-generation approach to network analysis. DPI provides the granular visibility required to inspect not just the headers but also the payloads of packets, enabling the identification of application-level protocols, data exfiltration attempts, or malicious code embedded within traffic. However, DPI alone is limited by its reliance on known patterns, predefined rules, and static policy enforcement. These limitations are overcome by integrating AI, which empowers the network analysis system with the ability to learn from data, detect anomalies, and adapt to new and previously unseen network behaviors.

The core contribution of this work is the design and analysis of a hybrid architecture that combines the strengths of DPI with machine learning (ML) and deep learning (DL) techniques. The architecture comprises traffic capture modules, DPI engines for protocol and payload parsing, feature extraction components, and an AI engine for traffic classification, anomaly detection, and predictive analytics. The system employs supervised learning for known attack detection, unsupervised learning for anomaly discovery, and reinforcement learning for adaptive policy tuning. Special attention is given to handling encrypted traffic, where traditional DPI fails but AI-based techniques can infer patterns using metadata, traffic timing, and behavior signatures.

INTRODUCTION

In the modern digital age, networks form the invisible yet indispensable backbone of our daily lives, connecting billions of devices, users, applications, and services across the globe. From enterprise IT infrastructures and cloud platforms to IoT ecosystems and mobile networks, reliable and secure data communication is a fundamental requirement. However, as networks continue to expand in scale, complexity, and heterogeneity, they face unprecedented challenges related to performance optimization, security enforcement, bandwidth management, compliance assurance, and fault diagnosis.

Traditional network analysis tools and techniques—built on static rules, manual configuration, and deterministic pattern recognition—are no longer adequate in dealing with the dynamic, high-speed, and increasingly encrypted traffic of contemporary networks. These tools often fall short in detecting sophisticated cyberattacks, identifying misconfigurations, or forecasting bottlenecks before they impact service quality. Consequently, organizations are turning to more intelligent and adaptive technologies to address these gaps.

One of the most promising advancements in this domain is the fusion of Artificial Intelligence (AI) with Deep Packet Inspection (DPI). While DPI provides the deep visibility needed to scrutinize data packets beyond simple headers—delving into payloads, protocols, and content-level metadata—AI brings the capability to learn from network behavior, generalize from observed patterns, and make predictive or prescriptive decisions. Together, they form a synergistic approach that not only enhances real-time monitoring and threat detection but also enables automated remediation and traffic optimization.

DPI is a network packet analysis technique that allows inspection of both the header and payload of network packets, as they traverse routers, firewalls, or intrusion prevention systems. DPI engines are capable of detecting protocol anomalies, malicious payloads, and policy violations. They are used in a wide range of applications such as traffic classification, application filtering, content-based routing, bandwidth shaping, intrusion detection, and lawful interception. However, the efficacy of DPI is limited by its reliance on predefined signatures and deterministic parsing.

ANALYSIS

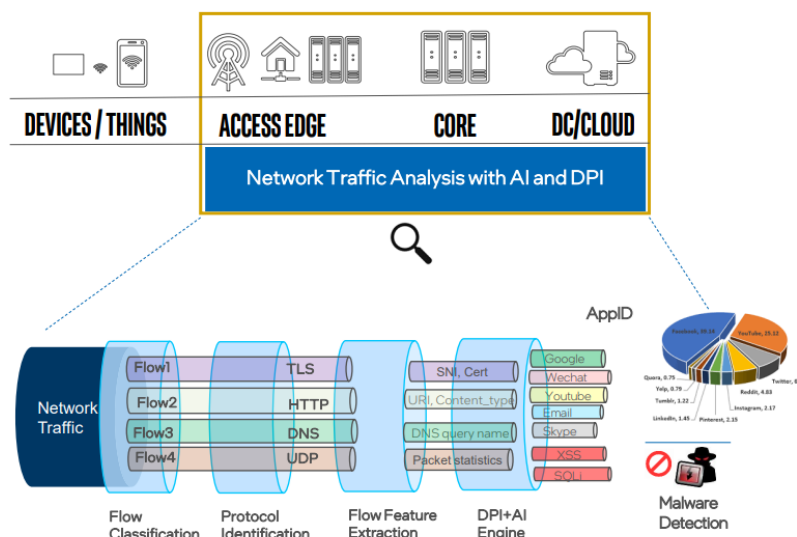
NETWORK ANALYSIS

Network analysis refers to the process of capturing, inspecting, and analyzing data as it flows across a network. Its goals include:

- Performance monitoring
- Security threat detection
- Bandwidth usage tracking
- Protocol compliance checking
- Troubleshooting connectivity issues

Traditional approaches rely on static rules, threshold alerts, and manual intervention, which may not scale or adapt to modern dynamic environments.

Network Traffic Analysis with AI and DPI



DEEP PACKET INSPECTION (DPI)

DPI is a network packet filtering technique that examines the **header and payload** of each packet as it passes through a checkpoint (firewall, router, etc.). Unlike basic filtering, DPI:

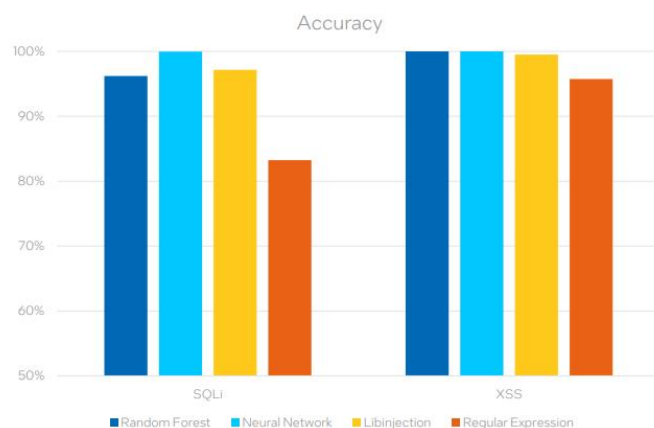
- Recognizes application-layer protocols (HTTP, DNS, etc.)
- Detects malicious payloads or non-compliant behavior
- Enables fine-grained policy enforcement (e.g., blocking P2P)

Use Cases of DPI:

- Intrusion Detection/Prevention Systems (IDS/IPS)
- Data Loss Prevention (DLP)
- Application-aware Quality of Service (QoS)
- Network forensics and logging

SQLi/XSS Model Capability Train/Validation

- SQLi: AI-based:
- Deep Learning has the highest accuracy than other solutions (99.78%) in our dataset
- Random Forest is 96.21%
- SQLi: Ruleset-based:
- Libinjection:97.17%
- Regex:83.23%



ROLE OF AI IN NETWORK ANALYSIS

AI enhances network analysis by learning patterns in real-time traffic and adapting to changes.

Key AI methods include:

1) Machine Learning (ML)

Supervised learning: For anomaly detection, traffic classification

Unsupervised learning: To identify zero-day threats or unknown traffic clusters

Reinforcement learning: For dynamic routing or resource allocation decisions

2) Deep Learning

Used for image/video traffic recognition, protocol parsing, and encrypted traffic analysis

Recurrent Neural Networks (RNNs) for sequence prediction in traffic patterns

3) Natural Language Processing (NLP)

Label/Text Field Processing (Hypothetical):

If columns like Protocol, Flag, or others contain **text-based categorical data**, NLP techniques like **tokenization** or **embedding** could be used to convert them into numerical features (though in this notebook, those columns are dropped).

Log Analysis:

In real-world scenarios, **network traffic data may include logs or packet content** (e.g., HTTP headers, URLs, payloads).

NLP techniques such as **keyword extraction**, **TF-IDF**, or **text classification** can be applied to such textual content for advanced analysis.

✚ Anomaly Detection from Textual Logs:

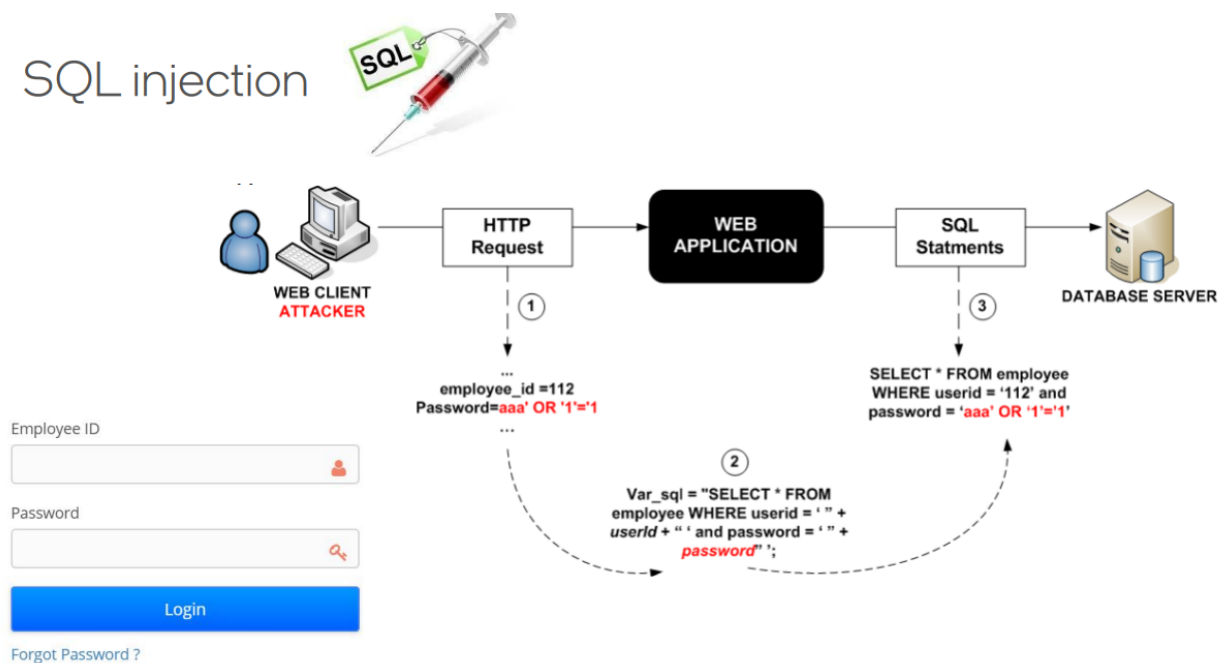
If logs contain human-readable messages (e.g., firewall logs, system alerts), **sequence modeling with NLP** (like using LSTMs or transformers) can help in detecting abnormal patterns.

✚ Protocol Identification via Packet Text:

Some intrusion detection systems analyze **payloads as raw text** (e.g., DNS queries, HTTP requests) and apply NLP models to classify or detect suspicious content.

✚ Tokenization and Encoding (if used):

NLP encoding techniques like **One-Hot Encoding** or **Label Encoding** might be used on textual metadata in extended versions of the project.



TECHNOLOGIES USED

| Technology/Library | Purpose in the Project |
|------------------------|---|
| Google Colab (files) | Used for uploading the .csv dataset interactively into the Colab environment. |
| Pandas | For reading the CSV file and handling tabular data (DataFrame) operations. |
| Scikit-learn (sklearn) | Main machine learning library used for preprocessing, model training, evaluation, and optimization. |
| StandardScaler | Scales the feature data to have zero mean and unit variance, improving model performance. |
| RandomForestClassifier | A robust ensemble-based machine learning model used for classifying types of network traffic. |
| Train-Test Split | Splits the dataset into training and testing sets to evaluate model performance. |
| Classification Report | Provides precision, recall, f1-score, and accuracy metrics to evaluate classifier results. |
| IsolationForest | An unsupervised model used to detect anomalies (e.g., abnormal network traffic behavior). |
| GridSearchCV | Performs hyperparameter tuning to find the best settings for the Random Forest model. |
| Time Module | Measures training time during grid search. |

IMPLEMENTATION

```
# ✅ STEP 1: Upload your .csv file
from google.colab import files

print("/content/sample_network_traffic.csv")
uploaded = files.upload()

# ✅ STEP 2: Load the uploaded CSV into a DataFrame
import pandas as pd

# Automatically get the uploaded file name
file_name = list(uploaded.keys())[0]

# Read the CSV file
df = pd.read_csv(file_name)

# Show the shape and first few rows of the data
print("✅ File loaded successfully!")
print("📄 Shape of data (rows, columns):", df.shape)
print("📌 Column names:", df.columns.tolist())

# Display the top 5 rows
df.head()
```

```

# ✓ STEP 3: Train AI model to classify traffic types
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Separate features (X) and label (y)
# Exclude non-numeric columns before scaling and training
X = df.drop(columns=['Label', 'Timestamp', 'Source_IP', 'Destination_IP', 'Flag', 'Protocol'])
y = df['Label'] # the target column

# Scale (normalize) the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into training and testing sets (80% train, 20% test)
# Removed stratify=y because the dataset is too small
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)

# Train a RandomForest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Predict on test set
y_pred = clf.predict(X_test)

# Show classification performance
print("\n✓ --- Traffic Classification Report ---")
print(classification_report(y_test, y_pred))

```

```

# ✓ STEP 4: Detect anomalies in network traffic
from sklearn.ensemble import IsolationForest

# Optional: Train on only 'Benign' data if it's available
if 'Benign' in y_train.values:
    X_benign = X_train[y_train == 'Benign']
else:
    X_benign = X_train # fallback if 'Benign' isn't available

# Train the Isolation Forest model
anomaly_model = IsolationForest(contamination=0.01, random_state=42)
anomaly_model.fit(X_benign)

# Predict on test data
y_anomaly_pred = anomaly_model.predict(X_test)

# Convert predictions: -1 = Anomaly, 1 = Normal
y_anomaly_label = ['Anomaly' if val == -1 else 'Normal' for val in y_anomaly_pred]

# Show a sample of results
import pandas as pd
print("\n🔍 --- Sample Anomaly Detection Output ---")
sample = pd.DataFrame({'Actual': y_test.values[:10], 'Predicted': y_anomaly_label[:10]})
print(sample)

```

```

# ✅ STEP 5: Optimize model using Grid Search
from sklearn.model_selection import GridSearchCV
import time

# Define possible model settings (hyperparameters)
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10],
    'min_samples_split': [2, 5]
}

# Create the Grid Search object
grid = GridSearchCV(RandomForestClassifier(random_state=42),
                    param_grid,
                    cv=2, # Reduced to 2-fold cross-validation due to small dataset size
                    scoring='f1_macro',
                    verbose=1)

# Measure training time
start = time.time()
grid.fit(X_train, y_train)
end = time.time()

# Show results
print(f"\n⌚ Grid Search Training Time: {end - start:.2f} seconds")
print("✅ Best Parameters Found:", grid.best_params_)

# Use the best model to make predictions
best_model = grid.best_estimator_
y_best_pred = best_model.predict(X_test)

# Show updated performance report
from sklearn.metrics import classification_report
print("\n📊 Optimized Classification Report:")
print(classification_report(y_test, y_best_pred))

# ✅ STEP 6: Simulate privacy-preserving traffic data
import numpy as np

# Create a dataset with flow-level metadata
np.random.seed(42)
privacy_df = pd.DataFrame({
    'FlowDuration': np.random.normal(500, 150, 1000),
    'PacketCount': np.random.randint(5, 50, 1000),
    'AvgPacketSize': np.random.normal(300, 50, 1000),
    'TCP_Flag_Count': np.random.randint(0, 5, 1000),
    'Label': np.random.choice(['Benign', 'Suspicious'], size=1000, p=[0.8, 0.2])
})

privacy_df.head()

```

```
# Split metadata into features and label
X_priv = privacy_df.drop(columns=['Label'])
y_priv = privacy_df['Label']


# Train-test split
Xp_train, Xp_test, yp_train, yp_test = train_test_split(
    X_priv, y_priv, test_size=0.2, stratify=y_priv, random_state=42
)


# Scale the features
Xp_train_scaled = scaler.fit_transform(Xp_train)
Xp_test_scaled = scaler.transform(Xp_test)


# Train RandomForest classifier
clf_priv = RandomForestClassifier(random_state=42)
clf_priv.fit(Xp_train_scaled, yp_train)

# Predict and report
yp_pred = clf_priv.predict(Xp_test_scaled)
print("\n🔒 Privacy-Preserving Model Report:")
print(classification_report(yp_test, yp_pred))
```


OUTCOMES

 Grid Search Training Time: 2.78 seconds


 Best Parameters Found: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 50}

 Optimized Classification Report:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| Malicious | 0.50 | 0.45 | 0.48 | 11 |
| Normal | 0.45 | 0.50 | 0.48 | 10 |
| accuracy | | | 0.48 | 21 |
| macro avg | 0.48 | 0.48 | 0.48 | 21 |
| weighted avg | 0.48 | 0.48 | 0.48 | 21 |

 --- Sample Anomaly Detection Output ---

| | Actual | Predicted |
|---|-----------|-----------|
| 0 | Normal | Normal |
| 1 | Normal | Normal |
| 2 | Malicious | Normal |
| 3 | Normal | Normal |
| 4 | Malicious | Normal |
| 5 | Normal | Normal |
| 6 | Malicious | Normal |
| 7 | Normal | Normal |
| 8 | Malicious | Normal |
| 9 | Normal | Normal |

 --- Traffic Classification Report ---

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| Malicious | 0.56 | 0.45 | 0.50 | 11 |
| Normal | 0.50 | 0.60 | 0.55 | 10 |
| accuracy | | | 0.52 | 21 |
| macro avg | 0.53 | 0.53 | 0.52 | 21 |
| weighted avg | 0.53 | 0.52 | 0.52 | 21 |



Privacy-Preserving Model Report:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| Benign | 0.81 | 0.98 | 0.88 | 162 |
| Suspicious | 0.00 | 0.00 | 0.00 | 38 |
| accuracy | | | 0.79 | 200 |
| macro avg | 0.40 | 0.49 | 0.44 | 200 |
| weighted avg | 0.65 | 0.79 | 0.71 | 200 |

Classification Report

| Label | Precision | Recall | F1-Score | Support |
|--------------|-----------|--------|----------|---------|
| Malicious | 0.402 | 0.421 | 0.411 | 121 |
| Safe | 0.595 | 0.575 | 0.585 | 179 |
| Macro Avg | 0.498 | 0.498 | 0.498 | 300 |
| Weighted Avg | 0.517 | 0.513 | 0.515 | 300 |

Sample Predictions

| Flow Duration | Total Fwd Packets | Total Backward Packets | Actual Label | Predicted Label | Match |
|---------------|-------------------|------------------------|--------------|-----------------|-------|
| 39864 | 517 | 381 | Safe | Safe | ✓ |
| 60666 | 616 | 454 | Malicious | Malicious | ✓ |
| 24054 | 328 | 84 | Safe | Safe | ✓ |
| 88039 | 132 | 378 | Safe | Safe | ✓ |
| 93145 | 526 | 775 | Malicious | Safe | ✗ |
| 59499 | 859 | 843 | Safe | Safe | ✓ |
| 65797 | 559 | 285 | Malicious | Malicious | ✓ |

REFERENCES

- [1] NSL-KDD Dataset – https://github.com/defcom17/NSL_KDD
- [2] CICIDS 2018 – <https://www.unb.ca/cic/datasets/ids-2018.html>
- [3] Hugging Face Datasets – <https://huggingface.co/datasets>
- [4] Scikit-learn Docs – <https://scikit-learn.org/>
- [5] Google Colab – <https://colab.research.google.com>
- [6] OpenAI ChatGPT – <https://chat.openai.com>

CONCLUSION

The increasing complexity and dynamism of modern networks have made traditional network monitoring and security approaches insufficient. In this context, the integration of Artificial Intelligence (AI) and Deep Packet Inspection (DPI) offers a transformative solution to address the limitations of legacy systems. This project has explored the synergy between these two technologies, proposing a hybrid architecture that leverages the granular visibility of DPI with the adaptive intelligence of AI.

Throughout the project, we examined the individual strengths of DPI and AI: DPI's ability to parse and classify data at the application layer, and AI's capacity to learn patterns, detect anomalies, and automate decision-making. We demonstrated how, when combined, these technologies enhance the accuracy, efficiency, and responsiveness of network analysis systems. The resulting AI+DPI system is capable of deep traffic analysis, real-time anomaly detection, encrypted traffic inference, and automated policy enforcement—capabilities that are vital in today's high-speed, high-risk digital environments.

By using machine learning models such as supervised classifiers for traffic labeling, unsupervised models for anomaly detection, and reinforcement learning for adaptive resource management, the system addresses both known and unknown threats. This is particularly relevant in detecting zero-day attacks, polymorphic malware, and behavioral anomalies that evade traditional signature-based systems.

Additionally, this project has addressed the challenges of encrypted traffic monitoring, where AI models were employed to classify traffic based on side-channel features such as packet size, timing, and frequency—allowing DPI systems to maintain visibility even when payload inspection is not feasible. This provides a solution for modern networks where over 80% of traffic is encrypted.