Anomaly Based DDOS detection using Deep Auto encoder .....

Introduction:

Today, our world is highly connected through networks and the internet. Because of this, **Distributed Denial of Service (DDoS) attacks** have become a big security threat. In a DDoS attack, many infected computers send huge amounts of traffic to a server, network, or service. This can make the service slow or completely unavailable. Detecting these attacks quickly is very important to keep online services running.

Most traditional detection systems use **signature-based methods**, which look for patterns of known attacks. These methods work only if the attack is already known. They often **fail to detect new or unknown attacks**, leaving systems at risk.

To solve this problem, our project uses an **anomaly-based approach**. This method does not need prior knowledge of attack patterns. Instead, it **learns what normal network traffic looks like** and flags anything unusual as a possible attack.

 In a **Distributed Denial of Service (DDoS)** attack, the attacker instructs **all the infected computers** to send massive amounts of traffic (requests or packets) to a target server or network.

 Since this traffic comes from **many sources**, it's much harder to block compared to a single-source attack.

Term	Meaning	Example
Known Attack	An attack that has been observed before and its signature is recorded.	SYN Flood that floods a server with a known packet pattern.
Unknown Attack	A new or modified attack that hasn't been seen before, so there's no signature yet.	A new type of DDoS attack using a different packet size pattern.

🔽 Why signature-based methods fail for unknown attacks

- If the attack **doesn't match any stored signature**, the system **won't recognize it**.
- That's why **anomaly-based methods** (like your Deep Autoencoder project) are better—they detect **unusual behavior**, not just known patterns.

1️⃣ What is a Deep Autoencoder?

A **Deep Autoencoder** is a type of neural network that learns to **compress and then reconstruct data**. It has two main parts:

1. **Encoder** → Compresses the input data into a smaller representation (latent space).
2. **Decoder** → Tries to reconstruct the original data from this compressed version.

Think of it like **summarizing a long paragraph into key points** and then **trying to recreate the original paragraph** from those points.

2️⃣ How it works with network traffic

- During **training**, the Autoencoder sees **only normal network traffic**.
    - It learns the typical patterns of packets: their size, timing, frequency, SYN/ACK ratios, etc.
    - After training, it becomes **very good at reconstructing normal traffic**.
- During **testing**, the network may have abnormal traffic (like a DDoS attack).
    - Because this traffic is different from what the Autoencoder has learned, it **fails to reconstruct it accurately**.
    - This failure is measured using **reconstruction error**.

3️⃣ What is Reconstruction Error?

- **Reconstruction Error** = Difference between the original input and the reconstructed output.
- High reconstruction error → The input data is unusual.
- Low reconstruction error → The input data is normal.

Example:

Packet Size	Actual	Reconstructed	Error
350 bytes	350	348	2 (low → normal)
64 bytes	64	200	136 (high → abnormal, possible attack)

- The Autoencoder detects **anomalies by looking at which samples have high errors**.

🔽 Why network metrics improve detection accuracy

Your system uses **additional network metrics** to strengthen detection:

Metric	How it helps detect DDoS
Packet Size & Duration	Very small packets, or unusually short durations, often occur in **UDP or ICMP floods**. Helps distinguish attacks from normal traffic.
Packets Per Second	Sudden spikes in packet rate indicate **volumetric attacks** that aim to overwhelm the network.
SYN/ACK Ratio	A high number of SYN packets compared to ACKs signals **SYN flood attacks** (connection requests are being sent, but not completed).
Unique Source IPs	Many different IPs in a short time can indicate **reflection/amplification attacks**, where attackers spoof source IPs.

- These metrics act as **indicators** that highlight suspicious traffic.

- Combining these metrics with **reconstruction error** allows the system to **not just detect anomalies but also classify the type of attack**.

5️⃣ Step-by-step workflow

1. **Collect network traffic** → Each packet has size, protocol, timestamp, flags, etc.
2. **Feature engineering** → Calculate metrics: Packets per second, SYN/ACK ratio, unique IPs, average packet size.
3. **Train Autoencoder** → Use **only normal traffic**, learn how it looks.
4. **Test Autoencoder** → Feed new traffic (normal + possibly attack).
5. **Calculate reconstruction error** →
    - Low error → Normal traffic
    - High error → Potential attack
6. **Combine with network metrics** → Decide attack type and confidence.

✅ Summary in simple words:

- **Deep Autoencoder** = Learns normal traffic patterns.
- **Reconstruction error** = Tells us if traffic is unusual.
- **Network metrics** = Give extra clues about attack type.

- **Together** → System detects attacks and classifies them, even if they are **new or unknown**.

Core Concept: Deep Autoencoder

The main technology we use is a **Deep Autoencoder**, which is a type of neural network used for **unsupervised learning**.

- During **training**, the Autoencoder learns to **reconstruct normal network traffic**.

- During **testing**, if traffic is abnormal (like during a DDoS attack), the Autoencoder cannot reconstruct it well, leading to **high reconstruction errors**. These errors help us detect anomalies.

This means the system can **detect attacks even if they are new or unknown**, without needing pre-defined attack patterns.

Key Features and Indicators

We use several network metrics to improve detection accuracy:

- **Packet Size & Duration:** Very small packets may indicate UDP floods.
- **Packets Per Second:** Sudden spikes in traffic may indicate volumetric attacks.
- **SYN/ACK Ratio:** A high ratio may indicate a SYN flood attack.
- **Unique Source IPs:** A large number of different IPs may indicate reflection or amplification attacks.

By combining these features with Autoencoder-based anomaly detection, the system **not only detects attacks but also identifies their type**, giving useful insights for response.

Why This Project is Unique

1. **Unsupervised Learning:** Can detect unknown or zero-day attacks.
2. **Multi-metric Detection:** Uses multiple network indicators for better analysis.
3. **Automated Attack Classification:** Detects anomalies and identifies attack types like SYN Flood, UDP Flood, or Reflection attacks.

4. **Scalable & Adaptive:** Can work in different network setups and adjust as normal traffic changes.

Why We Chose This Approach

- **Evolving Threats:** Attack patterns keep changing, and anomaly-based detection adapts automatically.
- **Real-Time Detection:** Can detect attacks as they happen.
- **Deep Learning Advantage:** Autoencoders capture complex patterns in network traffic that simple statistical methods cannot detect.

Summary:

This project uses **network metrics and deep learning** to detect DDoS attacks in a smart and adaptive way. Unlike traditional methods, it can detect unknown attacks and classify them, making it a **powerful tool for modern network security**.

1️⃣ Problem Statement

Your project focuses on **detecting network anomalies and DDoS attacks** using **machine learning (Autoencoder)** and heuristic rules.

- Input: Network traffic logs (Timestamp, Src_IP, Dst_IP, Protocol, Packet_Size, Duration, Flags)
- Goal: Identify unusual behavior in the network, especially DDoS attacks, in real time.
- Challenge: Normal traffic varies; DDoS attacks often involve high packet rates, SYN floods, small packets, or many unique source IPs.

2️⃣ Data Preparation

You work with **two types of datasets**:

1. **Training Dataset (Normal Traffic)**
    - Only contains legitimate network traffic.
    - Features extracted for modeling:
        - Packets_Per_Second
        - SYN_Count
        - ACK_Count
        - Unique_IPs_Per_Second
        - Packet_Size, Duration, Protocol
    - Purpose: Train the Autoencoder to learn the normal network behavior.
2. **Testing Dataset (Normal + DDoS Traffic)**
    - Contains a mix of normal traffic and simulated DDoS attack traffic.
    - DDoS indicators:
        - High Packets_Per_Second
        - High SYN_Count / ACK_Count ratio (SYN flood)
        - Many Unique_IPs_Per_Second (reflection/amplification attack)
        - Very small Packet_Size (UDP flood)

3️⃣ Feature Engineering

Before feeding the data to the Autoencoder:

1. Convert Timestamp to datetime format.
2. Encode or extract numeric features:
    - Keep numeric columns for ML model: Packet_Size, Duration, Packets_Per_Second, SYN_Count, ACK_Count, Unique_IPs_Per_Second.
3. Standardize features using a **scaler (e.g., StandardScaler)** for stable Autoencoder training.

4️⃣ Model Training

You are using an **Autoencoder**, which is an unsupervised neural network.

- **Purpose**: Learn the "normal" pattern of network traffic.
- **Workflow**:
    1. Input: Normal traffic numeric features.
    2. Autoencoder encodes input into a compressed representation.
    3. Autoencoder decodes it back to reconstruct input.
    4. **Loss**: Mean Squared Error (MSE) between original and reconstructed input.
    5. Result: Trained model that can reconstruct normal traffic well.
- **Important**: If the input deviates from normal (DDoS traffic), reconstruction error is **high**, flagging it as an anomaly.

5️⃣ Anomaly Detection

For each sample in the **test dataset**:

1. **Scale** the features using the same scaler as training.
2. **Predict reconstruction** using the trained Autoencoder.
3. **Calculate reconstruction error** (MSE).
4. **Define threshold** for anomalies:
    - Example: threshold = mean(error) + 1*std(error)
    - If error > threshold → sample is an **anomaly**.
- **Output**:
    - Reconstruction_Error, Anomaly_Score (normalized), Is_Anomaly (True/False)

6️⃣ DDoS Detection Heuristics

Your project also checks **DDoS indicators using rules**, which helps identify attack type:

- **High Packet Rate** → volumetric DDoS
- **High SYN/ACK Ratio** → SYN flood
- **High Unique IPs** → Reflection/Amplification attack
- **Small Packet Size** → UDP/ICMP flood

Algorithm:

1. Check each feature for thresholds.
2. Assign **confidence score** for each indicator.
3. Determine **attack type** based on detected indicators.
4. Mark the traffic as DDoS if any indicator is triggered.

7️⃣ Results and Reporting

After anomaly detection and DDoS evaluation:

- **CSV Outputs**:
    - detection_results.csv: Contains all traffic with anomaly scores, DDoS attack type, and confidence.
    - attack_summary.txt: Summary of detected attacks.
- **Visualizations**:

1. Histogram of reconstruction errors → highlights anomalies.
2. Time series of anomaly scores → shows when attacks occurred.
3. Optional: Plots for Packets_Per_Second, Unique_IPs_Per_Second over time.

- **Statistics**:
    - Number of anomalies detected
    - Percentage of anomalies
    - Attack type and confidence

8️⃣ Workflow Diagram (Stepwise)

Data Collection (Network Logs)

↓

Data Cleaning & Feature Extraction

↓

Scaler Fit (Normalize Features)

↓

Train Autoencoder on Normal Traffic

↓

Test Data → Scaled → Autoencoder Predict

↓

Compute Reconstruction Error

↓

Thresholding → Flag Anomalies

↓

Check DDoS Heuristics → Assign Attack Type & Confidence

↓

Save Results (CSV + Text)

↓

Generate Visualizations (Histogram + Time Series)

🚦 Key Points to Explain to Sir

- **Why Autoencoder?**
  Detects anomalies in **unsupervised manner** without labeled attack data.
- **Why Threshold + Heuristics?**
  Threshold identifies general anomalies. Heuristics **confirm DDoS type** and **increase detection confidence**.
- **Interpretation**:
  - High reconstruction error → potential anomaly.
  - Heuristic checks → categorize type of attack (SYN Flood, UDP Flood, Reflection).
  - Combining ML + rules → more robust detection.

- **Advantages**:
    - Can detect **novel attacks** not seen in training.
    - Works with **moderate-size datasets**.
    - Visualizations help in real-time monitoring.

1️⃣ Project Overview

Project Title: *Anomaly-based DDoS Attack Detection Using Deep Autoencoder*

Goal: Detect abnormal network traffic (DDoS attacks) automatically using deep learning without relying on pre-defined attack signatures.

Key Idea:

- Train a Deep Autoencoder on **normal traffic features**.
- During testing, any unusual pattern (attack traffic) will have **high reconstruction errors**, which are flagged as anomalies.
- Combine anomaly detection with DDoS indicators (like SYN/ACK ratio, packets per second) to classify attack types.

2️⃣ Module-wise Explanation

A. model_train.py – Autoencoder Training

Purpose:

- Train the Deep Autoencoder model to learn normal traffic patterns.

Key Points to Explain:

1. **build_autoencoder():**
    - Creates a fully connected neural network with **encoder → bottleneck → decoder**.
    - Bottleneck compresses input features, forcing the network to learn the essence of normal traffic.
2. **train_autoencoder_from_npy():**
    - Loads training data (only normal traffic) from a .npy file.

- Trains the model using **mean squared error (MSE)** loss.
- Saves trained model (autoencoder.h5), history (training_history.json), and loss plot.
- Uses **EarlyStopping** to prevent overfitting.

Functionality:

- Learns normal behavior. During attack, the Autoencoder **fails to reconstruct abnormal traffic**, which results in high reconstruction error.

B. model_detect.py – Anomaly & DDoS Detection

Purpose: Detect anomalies and DDoS attacks in new traffic data.

Key Points to Explain:

1. **Preprocessing:**
   - Loads test CSV data.
   - Converts timestamps to datetime and scales numeric features using the previously saved scaler.
2. **Reconstruction Error:**
   - Autoencoder predicts reconstructed traffic.
   - **Reconstruction Error = difference between original and reconstructed traffic**.
   - Errors above the **95th percentile** threshold are flagged as anomalies.
3. **DDoS Attack Indicators:**
   - **High Packet Rate:** Sudden traffic spike.
   - **High SYN/ACK Ratio:** Potential SYN flood.
   - **High Unique IPs:** Reflection/amplification attack.
   - **Small Packet Size:** UDP flood.
4. **Attack Classification:**
   - Combines anomalies + indicators to classify attack type: SYN Flood, UDP/ICMP Flood, Reflection, Generic DDoS.
5. **Output:**

- CSV file with reconstruction error, anomaly score, and attack type.
- Summary TXT file describing attack and confidence.
- Plots:
  - Histogram of reconstruction errors.
  - Time-series anomaly score over traffic.
  - Traffic metrics over time (packets/sec, unique IPs/sec).

Functionality:

- Allows **automatic detection** and **classification** of attacks in new network traffic.

C. model_utils.py – Data Preprocessing

Purpose: Prepare dataset for training/testing.

Key Points:

- Converts numeric columns, fills missing values, encodes categorical features like Protocol.
- Scales data using **MinMaxScaler**.
- Saves scaler for consistent preprocessing during testing.

Why Important:

- Autoencoder only works with **numeric, scaled features**, so preprocessing ensures data is compatible.

D. data_utils.py – Dataset Handling & Feature Extraction

Purpose: Handle raw input data and compute DDoS-related features.

Key Points:

1. **File Validation:** Ensures uploaded CSV/XLS files are valid.
2. **Feature Extraction:**
   - Required: Packet_Size, Duration, Protocol
   - DDoS-specific: Packets_Per_Second, Unique_IPs_Per_Second, SYN_Count, ACK_Count, etc.
   - Optional: Src_IP, Dst_IP, Timestamp, Flags

3. **Time-based Features:** Calculates per-second metrics.
4. **TCP Flags:** Counts SYN, ACK, RST, FIN packets for analysis.

Why Important:

- Provides all metrics needed for **anomaly detection** and **DDoS classification**.

3️⃣ Workflow (Step-by-Step)

1. **Prepare Training Data:**
    - Only normal traffic, numeric features scaled.
2. **Train Autoencoder:**
    - model_train.py learns normal behavior.
3. **Upload Test Dataset:**
    - Can include normal + attack traffic.
4. **Preprocess Test Data:**
    - Encode, scale, extract features (data_utils.py + model_utils.py).
5. **Predict & Calculate Reconstruction Error:**
    - Autoencoder attempts to reconstruct test traffic.
    - High error → potential anomaly.
6. **Detect DDoS Attack:**
    - Check traffic metrics (packet rate, SYN/ACK ratio, etc.).
    - Combine with anomalies to classify attack type.
7. **Save & Visualize Results:**
    - CSV of anomaly scores and attack types.
    - Plots of reconstruction error and metrics over time.
    - Summary TXT for quick review.

4️⃣ Key Points to Emphasize to Professor

- **Unsupervised Learning:** Model doesn't need prior attack examples. Can detect **unknown attacks**.

- **Multi-metric Detection:** Combines statistical features + deep learning anomaly detection.
- **Explainable Output:** Provides metrics and confidence for each detected attack.
- **Scalable & Adaptable:** Works with any network dataset if columns are consistent.
- **Visualization:** Makes it easy to **see attacks and anomalies** over time.

threshold = np.mean(reconstruction_error) + 1.0 * np.std(reconstruction_error)

What it means:

- reconstruction_error measures how different the Autoencoder's output is from the actual input for each sample.
- np.mean(reconstruction_error) is the average error for all normal traffic.
- np.std(reconstruction_error) is the standard deviation, showing how much the errors vary.
- Adding them gives a **threshold value**: anything above this is considered abnormal (an anomaly).

Using mean + std gives a balance between detecting anomalies and avoiding false alarms."

Anomaly = single abnormal traffic window.

DDoS = continuous anomalies + sudden traffic burst + many new source IP

Step 2 – Training the Autoencoder

We train a Deep Autoencoder (DAE) only on normal traffic.

The Autoencoder learns to compress and then reconstruct normal traffic patterns.

During training, it minimizes the difference between the input features and the reconstructed features.

Explanation format for sir

Step 1 – Data Preparation

We collect network traffic data and divide it into small time windows (e.g., 1–5 seconds).

For each window, we calculate important features such as packet rate, byte rate, unique source IPs, SYN/ACK ratio, etc.

These features represent the “normal behavior” of traffic.

Step 2 – Training the Autoencoder

We train a Deep Autoencoder (DAE) only on normal traffic.

The Autoencoder learns to compress and then reconstruct normal traffic patterns.

During training, it minimizes the difference between the input features and the reconstructed features.

Step 3 – Anomaly Detection

When new traffic comes in, we pass its features through the trained Autoencoder.

If the reconstruction error (difference between input and output) is small, the traffic is considered normal.

If the error is high, the traffic is considered anomalous (potential attack).

Step 4 – DDoS Identification

Not every anomaly means DDoS. So, we check patterns:

If many consecutive windows are anomalous,

If there is a sudden burst of new source IPs,

Or if SYN/ACK ratio is highly imbalanced,

→ then we declare it as a DDoS attack.

“We used a Deep Autoencoder algorithm for anomaly detection. It learns normal traffic patterns, and whenever the reconstruction error crosses a threshold, it flags the traffic as anomalous. If multiple anomalies occur together with high source diversity, we classify it as a DDoS attack.”

Why Deep Autoencoder is Powerful

- It can **capture complex non-linear patterns** in network traffic.
- Works without labeled data (unsupervised).

- Can detect **zero-day or unknown attacks** (those not seen before).
- Learns **general normal behavior**, not specific signatures.

💡 What is a Deep Autoencoder?

A **Deep Autoencoder** is a type of **neural network** used for **unsupervised learning** — meaning it learns patterns in data **without needing labels** (like "attack" or "normal").

It has two main parts:

1. **Encoder** — compresses the input into a smaller representation (like summarizing important info).
2. **Decoder** — reconstructs the original data back from that smaller form.

⚙️ How It Works

Let's say we give it normal network traffic data:

[Packet_Size, Duration, Packets_Per_Second, SYN_Count, ACK_Count, Unique_IPs]

🔹 Step 1: Encoding

- The **encoder** part of the model learns to capture the key patterns of normal traffic.
- It reduces the input into a smaller "latent vector" — kind of like a **summary** of normal behavior.
- Example: From 6 input features → compress to 2 or 3 key features.

🔹 Step 2: Decoding

- The **decoder** tries to reconstruct the original input from that summary.
- If the input is **normal**, the reconstruction will be **accurate** (small error).
- If the input is **abnormal (DDoS attack)**, the reconstruction will be **poor** (large error), because the model never learned that pattern.

If the input belongs to *normal traffic*, the Autoencoder can **reconstruct it very accurately** because it has learned that pattern during training.

But if the input is *abnormal (like a DDoS attack)*, it **cannot reconstruct it well**, producing a **high reconstruction error**.

Attack Type	Behavior / Indicator	How Your Model Detects It
SYN Flood Attack	Many SYN packets without corresponding ACKs	High SYN_Count, abnormal SYN/ACK ratio → high reconstruction error
UDP Flood Attack	Continuous small UDP packets from many sources	Abnormally low Packet_Size + high Packets_Per_Second
ICMP (Ping) Flood	Too many ICMP requests	Sudden traffic spike with short Duration
Reflection/Amplification Attack	Many unique source IPs targeting one destination	Abnormally high Unique_IPs_Per_Second
Generic / Mixed Floods	Random spikes in various metrics	Detected as general anomalies via high reconstruction error

⚙️ 2. How It Differentiates

Normal Traffic

- Matches what the model saw during training.
- Reconstruction error = low
  ✅ Classified as *Normal*

Anomalous but not DDoS

- Slightly unusual (like network delay, sudden small burst).
- Reconstruction error = moderate
  ⚠️ Classified as *Anomaly* but not DDoS-level.

DDoS Attack

- Very unusual traffic — too many packets, IPs, or SYNs.
- Reconstruction error = very high, exceeds threshold.
  🚨 Classified as *DDoS Attack*

Additionally, pattern rules (like high SYN count or many IPs) help label **attack type**

When you run your project, you'll get outputs like:

🔹 **Case 1 — Normal Traffic:**

[+] No DDoS attack detected. Network traffic appears normal.

🔹 **Case 2 — Detected Attack:**

=== DEBUG: DDoS Detection Results ===

Is DDoS: True

Attack Type: Generic DDoS

Confidence: 0.80

Indicators Found: ['High Packet Rate']

The output shows:

- Whether DDoS was detected.
- The **type of attack** (based on features).
- A **confidence score** (based on reconstruction error).
- The **indicators** that triggered detection.

💎 **5. Why This is Unique**