Evaluating Deep Sequential Models for Flow-Based Network Intrusion Detection Systems

Group 3

Jeong Hoon Choi, Yuxuan Liu

Introduction

- **Problem:** Traditional NIDS treat each network flow as an isolated sample
- **But many attacks unfold over sequences of flows:**
  - **Example: Slowloris Attack**
    - Each individual flow: small traffic, slow → looks normal
    - Sequence pattern: 100+ slow connections sustained over time → DoS
- **Question:** Can we improve intrusion detection by looking at sequences of network flows? And which model works best?
- **Our Approach:** Compare three models to answer this question

Packet-Based vs Flow-Based Detection

- **Packet-Based Detection: (traditional method)**
  - Inspects individual packet contents (payload)
  - Cannot handle encrypted traffic (95%+ of internet)
  - High computational cost
  - Privacy concerns

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	8.254.250.126	192.168.10.5	TCP	60	80 → 49188 [FIN, ACK] Seq=1 Ack=1 Win=329 Len=0
2	0.000004	8.254.250.126	192.168.10.5	TCP	60	[TCP Retransmission] 80 → 49188 [FIN, ACK] Seq=1 Ack=1 Win=329 Len=0
3	0.000005	8.254.250.126	192.168.10.5	TCP	60	[TCP Retransmission] 80 → 49188 [FIN, ACK] Seq=1 Ack=1 Win=329 Len=0
4	0.000006	8.254.250.126	192.168.10.5	TCP	60	[TCP Retransmission] 80 → 49188 [FIN, ACK] Seq=1 Ack=1 Win=329 Len=0
5	0.000007	8.254.250.126	192.168.10.5	TCP	60	[TCP Retransmission] 80 → 49188 [FIN, ACK] Seq=1 Ack=1 Win=329 Len=0
6	0.000008	8.254.250.126	192.168.10.5	TCP	60	[TCP Retransmission] 80 → 49188 [FIN, ACK] Seq=1 Ack=1 Win=329 Len=0
7	0.000009	8.254.250.126	192.168.10.5	TCP	60	[TCP Retransmission] 80 → 49188 [FIN, ACK] Seq=1 Ack=1 Win=329 Len=0
8	0.000010	8.254.250.126	192.168.10.5	TCP	60	[TCP Retransmission] 80 → 49188 [FIN, ACK] Seq=1 Ack=1 Win=329 Len=0
9	5.667609	Cisco_1d:bb:04	CDP/VTP/DTP/PAgP/U…	CDP	457	Device ID: SWCL Port ID: GigabitEthernet1/0/4
10	5.759953	Cisco_1d:bb:06	CDP/VTP/DTP/PAgP/U…	CDP	457	Device ID: SWCL Port ID: GigabitEthernet1/0/6
11	16.308519	Cisco_1d:bb:0c	CDP/VTP/DTP/PAgP/U…	CDP	458	Device ID: SWCL Port ID: GigabitEthernet1/0/12
12	17.306042	Cisco_1d:bb:0c	CDP/VTP/DTP/PAgP/U…	CDP	458	Device ID: SWCL Port ID: GigabitEthernet1/0/12
13	18.109565	Cisco_1d:bb:0c	CDP/VTP/DTP/PAgP/U…	CDP	458	Device ID: SWCL Port ID: GigabitEthernet1/0/12
14	19.112195	Cisco_1d:bb:0c	CDP/VTP/DTP/PAgP/U…	CDP	458	Device ID: SWCL Port ID: GigabitEthernet1/0/12
15	20.109145	Cisco_1d:bb:0c	CDP/VTP/DTP/PAgP/U…	CDP	458	Device ID: SWCL Port ID: GigabitEthernet1/0/12
16	20.986042	Cisco_1d:bb:08	CDP/VTP/DTP/PAgP/U…	CDP	457	Device ID: SWCL Port ID: GigabitEthernet1/0/8
17	21.109124	Cisco_1d:bb:0c	CDP/VTP/DTP/PAgP/U…	CDP	458	Device ID: SWCL Port ID: GigabitEthernet1/0/12
18	22.108872	Cisco_1d:bb:0c	CDP/VTP/DTP/PAgP/U…	CDP	458	Device ID: SWCL Port ID: GigabitEthernet1/0/12
19	23.110389	Cisco_1d:bb:0c	CDP/VTP/DTP/PAgP/U…	CDP	458	Device ID: SWCL Port ID: GigabitEthernet1/0/12
20	23.732710	8.253.185.121	192.168.10.14	TCP	60	80 → 49486 [FIN, ACK] Seq=1 Ack=1 Win=245 Len=0

CIC-IDS-2017 PCAPs (48 GB)

Packet-Based vs Flow-Based Detection

- **Flow-Based Detection:**
  - Analyzes statistical features of traffic flows
  - Works with encrypted traffic
  - Efficient and scalable
  - Privacy-preserving

Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packe
49188	4	2	0	12	0	6	6	
49188	1	2	0	12	0	6	6	
49188	1	2	0	12	0	6	6	
49188	1	2	0	12	0	6	6	
49486	3	2	0	12	0	6	6	
49486	1	2	0	12	0	6	6	
49486	1	2	0	12	0	6	6	
49486	1	2	0	12	0	6	6	
88	609	7	4	484	414	233	0	
88	879	9	4	656	3064	313	0	
88	1160	9	6	3134	3048	1552	0	3

CIC-IDS-2017 CSVs (880 MB) - CICFlowMeter

Packet-Based —> Flow-Based

No.	Time	Source	Destination	Protocol	Length	Info
178	39.435254	192.168.10.9	192.168.10.3	TCP	66	1031 → 88 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
179	39.435302	192.168.10.9	192.168.10.3	TCP	66	[TCP Retransmission] 1031 → 88 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
180	39.435321	192.168.10.3	192.168.10.9	TCP	66	88 → 1031 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
181	39.435324	192.168.10.3	192.168.10.9	TCP	66	[TCP Retransmission] 88 → 1031 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PE
182	39.435351	192.168.10.9	192.168.10.3	TCP	60	1031 → 88 [ACK] Seq=1 Ack=1 Win=65536 Len=0
183	39.435353	192.168.10.9	192.168.10.3	TCP	60	[TCP Dup ACK 182#1] 1031 → 88 [ACK] Seq=1 Ack=1 Win=65536 Len=0
184	39.435401	192.168.10.9	192.168.10.3	KRB5	287	AS-REQ
185	39.435403	192.168.10.9	192.168.10.3	TCP	287	[TCP Retransmission] 1031 → 88 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=233
186	39.435784	192.168.10.3	192.168.10.9	KRB5	261	KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
187	39.435788	192.168.10.3	192.168.10.9	TCP	261	[TCP Retransmission] 88 → 1031 [PSH, ACK] Seq=1 Ack=234 Win=525568 Len=207
188	39.435863	192.168.10.9	192.168.10.3	TCP	60	1031 → 88 [FIN, ACK] Seq=234 Ack=208 Win=65280 Len=0

88	609	7	4	484	414	233	0	69.1428571429	111.9678950584

39.435863 - 39.435254 = 0.000609

Dataset Comparison - Original vs NF-v2

Aspect	UNSW-NB15	CIC-IDS2018	NF-UNSW-NB15-v2	NF-CSE-CIC-IDS2018-v2
Format	PCAP (packets)	PCAP/CSV (packets)	NetFlow v2 (flows)	NetFlow v2 (flows)
Data Level	Individual packets	Individual packets	Aggregated flows	Aggregated flows
Feature Set	UNSW-specific	CIC-specific	Standardized	Standardized
Features	49	80+	43	43
Instances	~2.5M packets	~16M packets	2,390,275 flows	18,893,708 flows
Missing & Duplicates	Yes	Yes	No	No
Payload Data	Included	Included	Not Included	Not Included

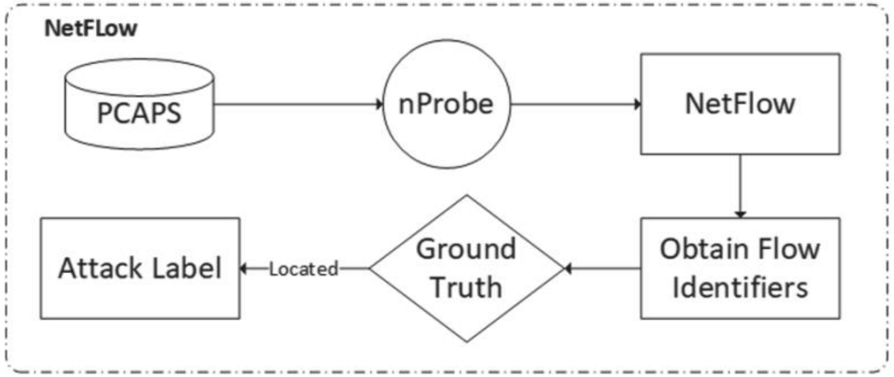
Dataset Selection - Why NF-v2?

- **Background:**
  - Original datasets (UNSW-NB15, CIC-IDS2018) have different feature sets
  - Difficult to compare model performance across datasets
  - Packet-based format not suitable for modern encrypted networks

- **NF-v2 Standardized Datasets:**
  - Created by Sarhan et al. (2022) for fair NIDS comparison
  - 43 standardized NetFlow features across all datasets
  - Preprocessed: no missing values, no duplicates
  - Flow-based aggregation (privacy-preserving, works with encryption)

Data Preprocessing Pipeline (by Sarhan et al., 2022)

1. Input: Original PCAP files UNSW-NB15 & CIC-IDS2018



2. Extraction:
Use nProbe to convert packets → NetFlow format

- ❿Aggregates packets into flows based on 5-tuple
- Extracts 43 standardized NetFlow features

3. Labeling: Match flow identifiers with ground truth labels

- Preserves original attack labels from datasets

4. Output: Clean NetFlow v2 datasets with consistent format

Datasets

	Data Size	# of instances	# of attributes	# of labels (Attack)
NF-UNSW-NB15-v2	421M	2,390,275	43	9
NF-CSE-CIC-IDS-2018-v2	3.0G	18,893,708	43	7

Attack types in our datasets:

- **NF-UNSW-NB15-v2:** Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, Shellcode, Worms
- **NF-CSE-CIC-IDS-2018-v2:** BruteForce, Bot, DoS, DDoS, Infiltration, Web Attacks, Web Attacks

Challenges:

- Severe class imbalance (96% and 88% benign)
- Hard-to-detect attacks (Slowloris, Infiltration) mimic normal traffic

Attack Types

NF-UNSW-NB15

Class	Count	Description
Benign	1550712	Normal unmalicious flows
Fuzzers	19463	An attack in which the attacker sends large amounts of random data which cause a system to crash and also aim to discover security vulnerabilities in a system
Analysis	1995	A group that presents a variety of threats that target web applications through ports, emails and scripts
Backdoor	1782	A technique that aims to bypass security mechanisms by replying to specific constructed client applications
DoS	5051	Denial of Service is an attempt to overload a computer system's resources with the aim of preventing access to or availability of its data
Exploits	24736	Are sequences of commands controlling the behaviour of a host through a known vulnerability
Generic	5570	A method that targets cryptography and causes a collision with each block-cipher
Reconnaissance	12291	A technique for gathering information about a network host and is also known as a probe
Shellcode	1365	A malware that penetrates a code to control a victim's host
Worms	153	Attacks that replicate themselves and spread to other computers

Attack Types

NF-UNSW-NB15

Class	Count	Description
Benign	1550712	Normal unmalicious flows
Fuzzers	19463	An attack in which the attacker sends large amounts of random data to crash and also aim to vulnerabilities in a syste
Analysis	1995	A group that presents a target web applications and scripts
Backdoor	1782	A technique that aims to bypass security mechanisms by replying to specific constructed client applications
DoS	5051	Denial of Service is an a computer system's resou preventing access to or a
Exploits	24736	Are sequences of comma behaviour of a host through a known vulnerability
Generic	5570	A method that targets cryptography and causes a collision with e
Reconnaissance	12291	A technique for gatheri network host and is also
Shellcode	1365	A malware that penetra victim's host
Worms	153	Attacks that replicate themselves and spread to other computers

```
[Analysis]
-- TP: 1, FP: 11, TN: 358186, FN: 344
-- Precision: 0.0833 (1 / 12)
-- Recall: 0.0029 (1 / 345)
-- Specificity: 1.0000 (358186 / 358197)
-- Accuracy: 0.9990
-- F1-score: 0.0056
-- Balanced Accuracy: 50.1434
```

```
[DoS]
-- TP: 182, FP: 1258, TN: 356415, FN: 687
-- Precision: 0.1264 (182 / 1440)
-- Recall: 0.2094 (182 / 869)
-- Specificity: 0.9965 (356415 / 357673)
-- Accuracy: 0.9946
-- F1-score: 0.1576
-- Balanced Accuracy: 60.2959
```

```
[Reconnaissance]
-- TP: 1522, FP: 251, TN: 356374, FN: 395
-- Precision: 0.8584 (1522 / 1773)
-- Recall: 0.7939 (1522 / 1917)
-- Specificity: 0.9993 (356374 / 356625)
-- Accuracy: 0.9982
-- F1-score: 0.8249
-- Balanced Accuracy: 89.6623
```

Attack Types

NF-CSE-CIC-IDS2018

Class	Count	Description
Benign	7373198	Normal unmalicious flows
BruteForce	287597	A technique that aims to obtain usernames and password credentials by accessing a list of predefined possibilities
Bot	15683	An attack that enables an attacker to remotely control several hijacked computers to perform malicious activities
DoS	269361	An attempt to overload a computer system's resources with the aim of preventing access to or availability of its data
DDoS	380096	An attempt similar to DoS but has multiple different distributed sources
Infiltration	62072	An inside attack that sends a malicious file via an email to exploit an application and is followed by a backdoor that scans the network for other vulnerabilities
Web Attacks	4394	A group that includes SQL injections, command injections and unrestricted file uploads

Attack Types

NF-CSE-CIC-IDS2018

Class	Count	Description
Benign	7373198	Normal unma
BruteForce	287597	A technique t and password of predefined
Bot	15683	An attack tha remotely cont to perform ma
DoS	269361	An attempt to overload a computer system's resources with the aim of preventing access to or availability of its data
DDoS	380096	An attempt sim different distrib
Infiltration	62072	An inside attac via an email to followed by a backdoor that scans the network for othe
Web Attacks	4394	A group that inc command injecti uploads

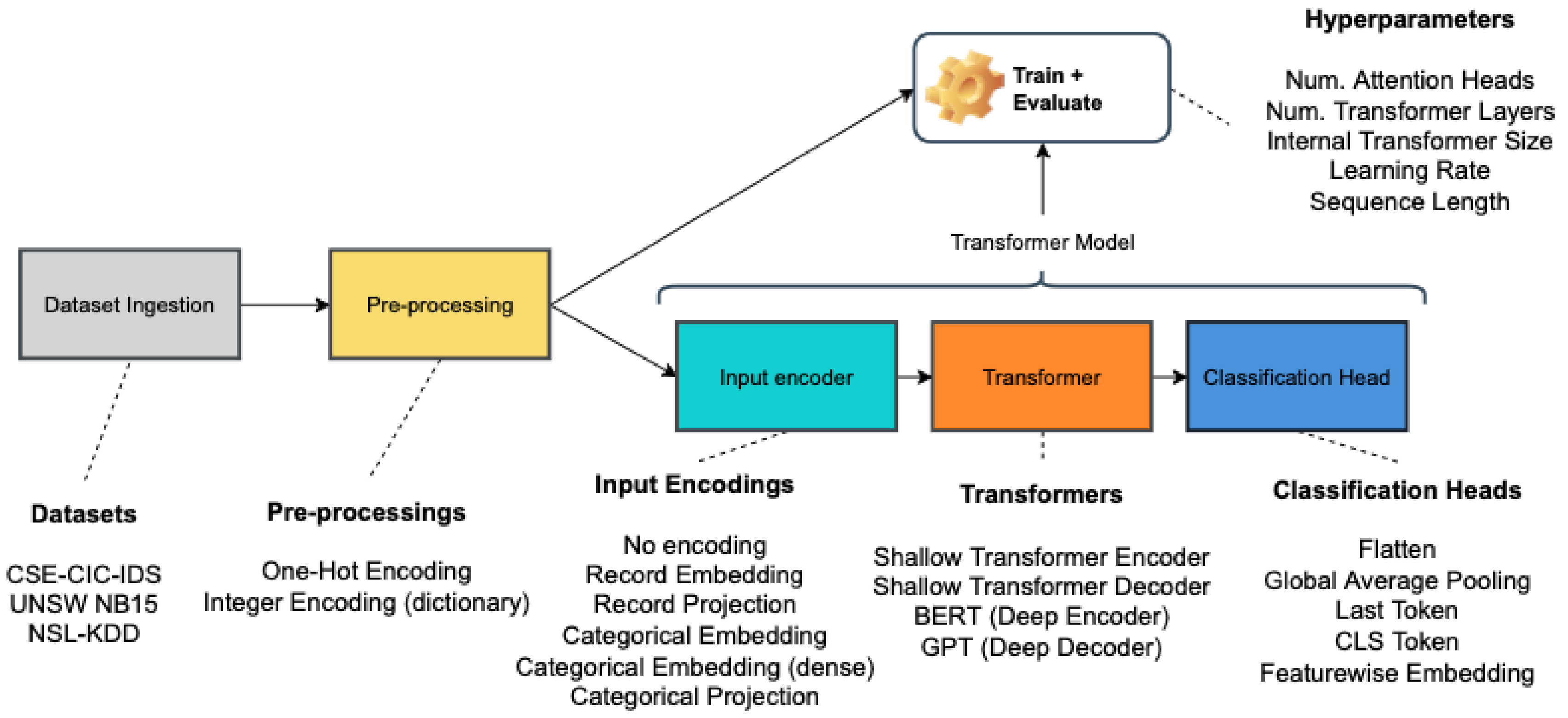
```
[Brute Force -Web]
-- TP: 11, FP: 1121, TN: 448823, FN: 45
-- Precision: 0.0097 (11 / 1132)
-- Recall: 0.1964 (11 / 56)
-- Specificity: 0.9975 (448823 / 449944)
-- Accuracy: 0.9974
-- F1-score: 0.0185
-- Balanced Accuracy: 59.6969

[Brute Force -XSS]
-- TP: 0, FP: 9, TN: 449969, FN: 22
-- Precision: 0.0000 (0 / 9)
-- Recall: 0.0000 (0 / 22)
-- Specificity: 1.0000 (449969 / 449978)
-- Accuracy: 0.9999
-- F1-score: 0.0000
-- Balanced Accuracy: 49.9990
```

```
[DDoS attacks-LOIC-HTTP]
-- TP: 7331, FP: 13, TN: 442656, FN: 0
-- Precision: 0.9982 (7331 / 7344)
-- Recall: 1.0000 (7331 / 7331)
-- Specificity: 1.0000 (442656 / 442669)
-- Accuracy: 1.0000
-- F1-score: 0.9991
-- Balanced Accuracy: 99.9985
```

```
[DDOS attack-HOIC]
-- TP: 2542, FP: 9, TN: 424261, FN: 23188
-- Precision: 0.9965 (2542 / 2551)
-- Recall: 0.0988 (2542 / 25730)
-- Specificity: 1.0000 (424261 / 424270)
-- Accuracy: 0.9485
-- F1-score: 0.1798
-- Balanced Accuracy: 54.9387
```

Related Work
Flow Transformers (by Manocchio et al., 2023)

- Introducing a **Transformer-based** architecture for **flow-level** network traffic
- Leverage self-attention to capture global relationships between packets within and across flows



Related Work
Flow Transformers (by Manocchio et al., 2023)

- **Categorical Embedding:**
  - Netflow features include categorical attributes (e.g., port, protocol)
  - Maps categorical fields to integer indices
  - Uses a fixed size vector: up to 32 unique categorical levels

```
'CLIENT_TCP_FLAGS', 'L4_SRC_PORT', 'TCP_FLAGS', 'ICMP_IPV4_TYPE',
'ICMP_TYPE', 'PROTOCOL', 'SERVER_TCP_FLAGS', 'L4_DST_PORT', 'L7_PROTO'
```

```
pre_processing = StandardPreProcessing(n_categorical_levels=32)
encoding = RecordLevelEmbed(embed_dim)
```

NF-UNSW-NB15-v2-pre.csv: 269 features

NF-CSE-CIC-IDS2018-pre.csv: 291 features

Our Approach

1. BaselineFCN (Baseline)

- Input: Single flow
- Architecture: Fully connected layers
- No temporal modeling

2. CNN+LSTM (Sequential Hybrid)

- Input: Sequence of 8 flows
- Architecture: CNN + LSTM

3. NetFlowBERT (BERT-based Sequential)

- Input: Sequence of 8 flows
- Architecture: BERT with bidirectional attention
- Captures full sequence context

Key Accomplishments

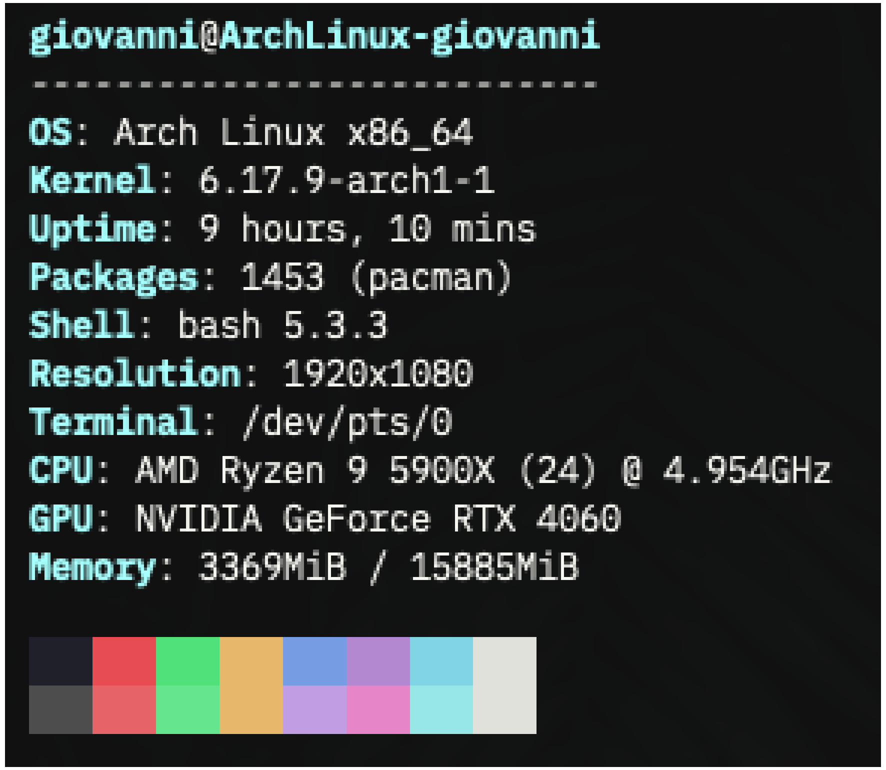
1. Implemented three architectures for systematic comparison

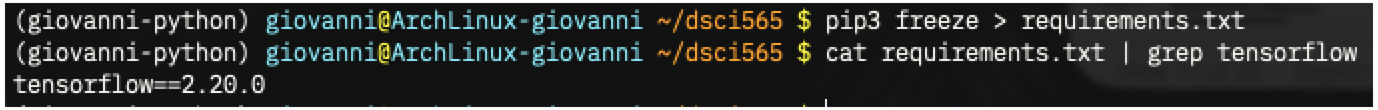
- BaselineFCN (non-sequential)
- CNN+LSTM (sequential hybrid)
- NetFlowBERT (BERT-based sequential)

2. Multi-class classification

- 10 classes on UNSW-NB15
- 8 classes on CIC-IDS2018

Experimental Setup



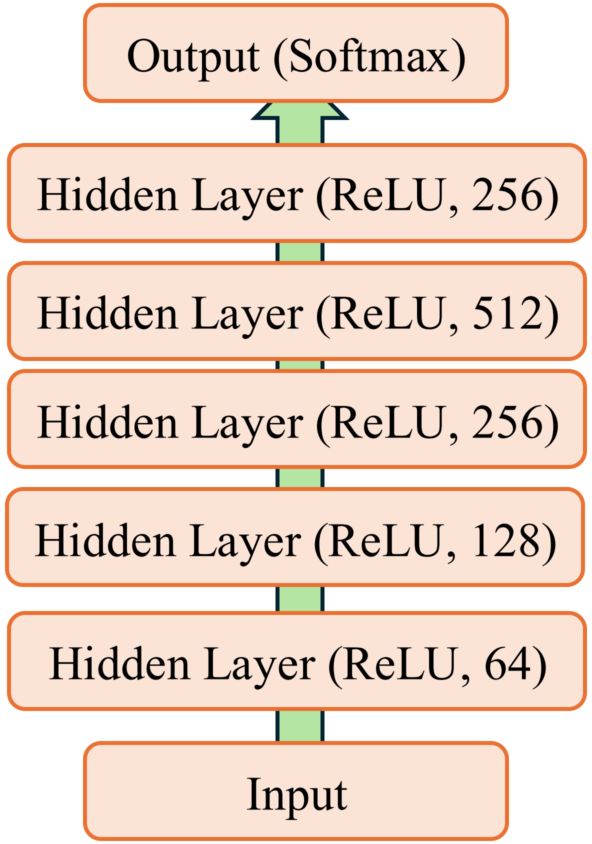


GPU: Nvidia RTX 4060 8 GB GDDR6

Python 3.13.7
TensorFlow 2.20.0

https://github.com/csian98/Anomaly-Detection

Baseline FCN Architecture

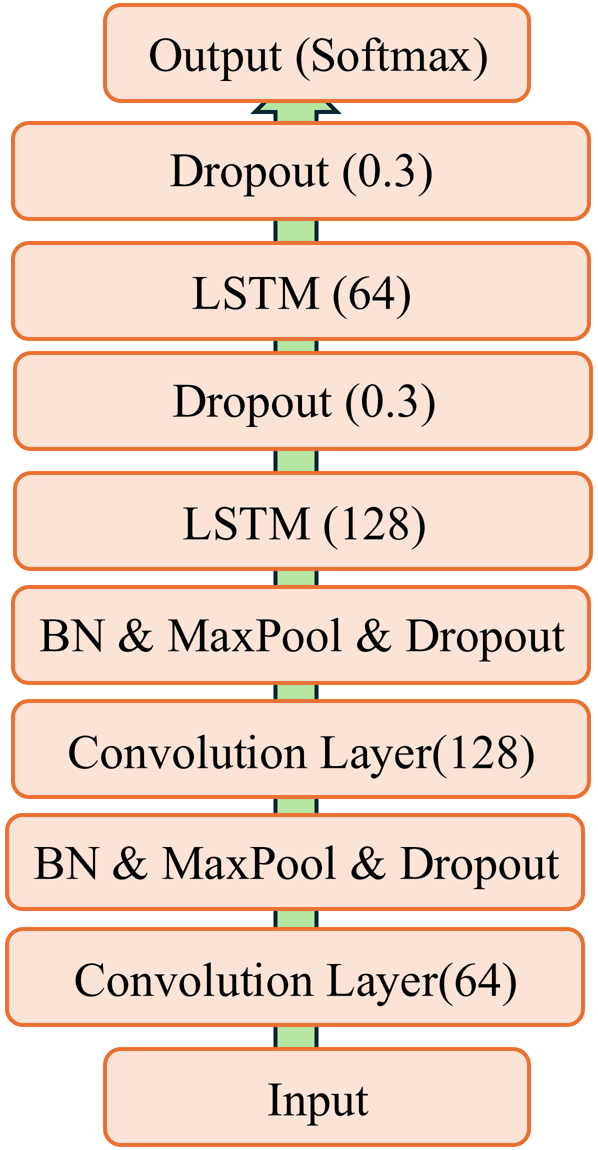



```python
# Data Structures define - class #
class BaselineFCN(tf.keras.Model):
    def __init__(self, hidden_sizes=[128, 64], dropout=0.3, num_classes=10):
        super().__init__()
        self.hidden_layer = []
        for h in hidden_sizes:
            self.hidden_layer.append(layers.Dense(h, activation="relu"))
            self.hidden_layer.append(layers.Dropout(dropout))

        self.output_layer = layers.Dense(num_classes, activation="softmax")

    def call(self, X, training=False):
        for layer in self.hidden_layer:
            X = layer(X, training=training)
        return self.output_layer(X)
```

Epochs	300 (Patience 5)
Hidden Size	(64, 128, 256, 512, 256)
Dropout	0.3
Class Weight	True
Total Parameters	324,106

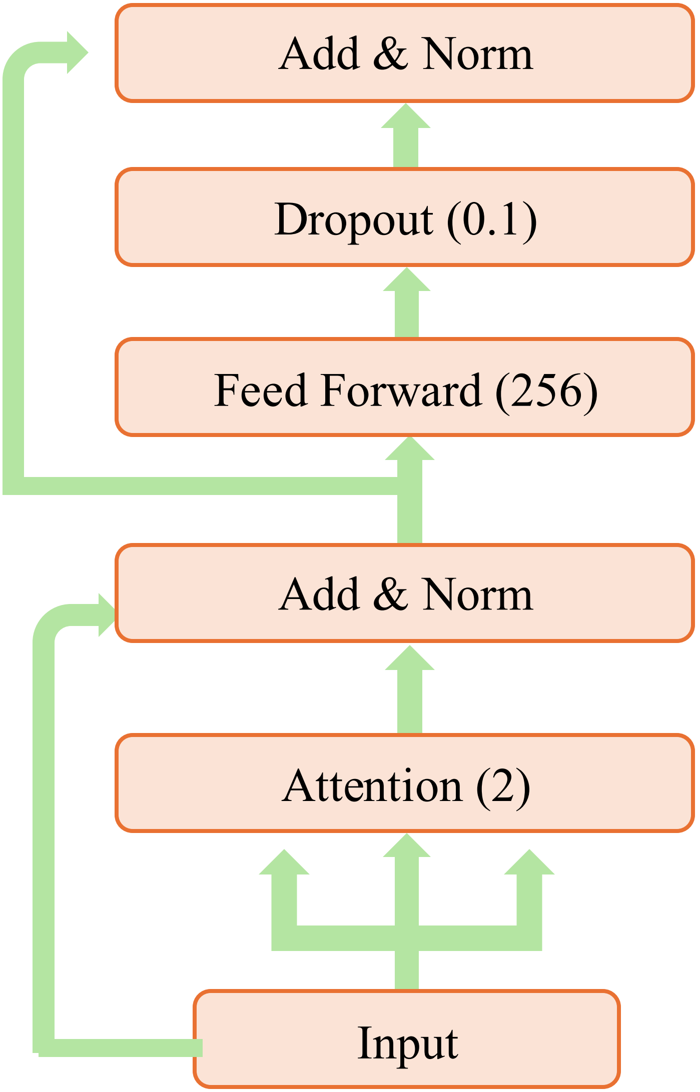
CNN+LSTM Architecture

```python
# Data Structures define - class #
class NetFlowCNNLSTM(tf.keras.Model):
    def __init__(self, num_classes, cnn_filters, kernel_size,
                 lstm_units, dropout_rate):
        super().__init__()

        self.conv1 = layers.Conv1D(cnn_filters[0], kernel_size, padding="same", activation="relu")
        self.conv2 = layers.Conv1D(cnn_filters[1], kernel_size, padding="same", activation="relu")

        self.bn1 = layers.BatchNormalization()
        self.bn2 = layers.BatchNormalization()

        self.maxpool1 = layers.MaxPooling1D(pool_size=2)
        self.maxpool2 = layers.MaxPooling1D(pool_size=2)

        self.drop1 = layers.Dropout(dropout_rate)
        self.drop2 = layers.Dropout(dropout_rate)

        self.lstm1 = layers.LSTM(lstm_units[0], return_sequences=True)
        self.lstm2 = layers.LSTM(lstm_units[0], return_sequences=False)

        self.drop3 = layers.Dropout(dropout_rate)
        self.drop4 = layers.Dropout(dropout_rate)

        self.classifier = layers.Dense(num_classes, activation="softmax")

    def call(self, X, training=False):
        X = self.conv1(X, training=training)
        X = self.bn1(X, training=training)
        X = self.maxpool1(X, training=training)
        X = self.drop1(X, training=training)

        X = self.conv2(X, training=training)
        X = self.bn2(X, training=training)
        X = self.maxpool2(X, training=training)
        X = self.drop2(X, training=training)

        X = self.lstm1(X, training=training)
        X = self.drop3(X, training=training)

        X = self.lstm2(X, training=training)
        X = self.drop4(X, training=training)

        X = self.classifier(X, training=training)
        return X
```



Epochs	300 (Patience 5)
Window Size	8
CNN Filters	[64, 128]
Kernel Size	3
LSTM Units	[128, 64]
Dropout	0.3
Class Weight	True
Total Parameters	341,642

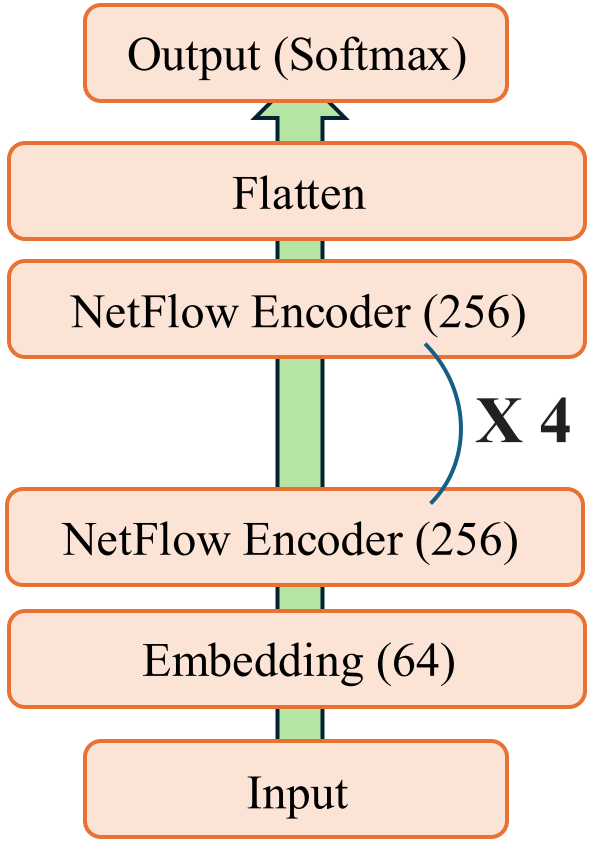
NetFlow BERT Architecture




```
# Data Structures define - class #
class TransformerEncoder(layers.Layer):
    def __init__(self, num_hiddens:int, num_heads:int,
                 ffn_num_hiddens:int, dropout=0.1):
        super().__init__()
        self.attention = layers.MultiHeadAttention(num_heads=num_heads, key_dim=num_hiddens)
        self.ffn = models.Sequential([
            layers.Dense(ffn_num_hiddens, activation="relu"),
            layers.Dense(num_hiddens),
        ])
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(dropout)
        self.dropout2 = layers.Dropout(dropout)

    def call(self, X, training=False):
        fX = self.attention(X, X)   # self attention
        fX = self.dropout1(fX, training=training)
        X = self.layernorm1(X + fX) # add & norm

        fX = self.ffn(X)
        fX = self.dropout2(fX, training=training)
        X = self.layernorm2(X + fX) # add & norm
        return X
```

NetFlow BERT Architecture

```python
class NetFlowBertClassifier(tf.keras.Model):
    def __init__(self, embed_size=64, n_layers=2, internal_size=128,
                 n_heads=2, dropout=0.1, num_classes=10):
        super().__init__()
        self.embedding = layers.Dense(embed_size)
        self.encoders = [
            TransformerEncoder(embed_size, n_heads, internal_size, dropout)
            for _ in range(n_layers)]

        # self.pool = layers.GlobalAveragePooling1D()
        # self.dropout = layers.Dropout(dropout)
        self.classifier = layers.Dense(num_classes, activation="softmax")

    def call(self, X, training=False):
        X = self.embedding(X)
        for encoder in self.encoders:
            X = encoder(X, training=training)

        # X = self.pool(X)
        # X = self.dropout(X, training=training)
        X = X[:, -1, :]
        X = self.classifier(X)
        return X
```



Epochs	300 (Patience 5)
Window Size	8
Layers	4
Embedding Size	64
Internal Size	256
Attention Head	2
Dropout	0.1
Class Weight	True
Total Parameters	284,170

Model Comparison Analysis

Aspect	BaselineFCN	CNN+LSTM	NetFlowBERT
Input	Single flow	8-flow sequence	8-flow sequence
Temporal Modeling	None	LSTM	Bidirectional Attention
Architecture	5-layer FCN	2 Conv1D + 2 LSTM	4-layer BERT
Dropout	0.3	0.3	0.1
Batch Size	64	256	256
Parameters	324,106	341,642	284,170
Pros	Simple, interpretable	Spatial+temporal modeling Balanced performance (+27%)	Best overall (83.44%) Excels on DoS attacks
Cons	No sequence context Poor accuracy	Inconsistent on some DoS	Higher computational cost
Best Use Case	Baseline reference	Balanced deployment	Maximum accuracy

Model Performance Comparison

NF-UNSW-NB15

Model	Macro F1	Accuracy
BaselineFCN	0.5072	98.16%
CNN+LSTM	0.5888	98.41%
NetFlowBERT	0.6233	98.39%

Model	Analysis		DoS		Recon	
	Recall	F1 score	Recall	F1 Score	Recall	F1 Score
BaselineFCN	0.00	0.01	0.21	0.16	0.79	0.82
CNN+LSTM	0.54	0.30	0.39	0.30	0.82	0.82
NetFlowBERT	0.59	0.62	0.50	0.27	0.88	0.87

Model Performance Comparison

NF-CSE-CIC-IDS2018

Model	Macro F1	Accuracy
BaselineFCN	0.6190	50.58%
CNN+LSTM	0.5770	64.45%
NetFlowBERT	0.6389	83.44%

Model	Benign		Brute Force - HTTP		DDoS HOIC	
	Recall	F1 score	Recall	F1 Score	Recall	F1 Score
BaselineFCN	0.50	0.66	0.20	0.02	0.10	0.18
CNN+LSTM	0.65	0.79	0.48	0.00	0.24	0.38
NetFlowBERT	0.87	0.93	0.02	0.00	0.10	0.18

NetFlow Bert Further Training

1. Increase Model Complexity

- Increase Attention Head x4
- Increase total parameters x30

2. SparseCategoricalFocalLoss

- Class Weight+Categorical Focal Loss
- Nonlinear transformation of cross entropy

$$-\sum_{i=1}^{i=n} \alpha_i log_b(p_i)$$

Epochs	300 (Patience 10)
Window Size	8
Layers	8
Embedding Size	64
Internal Size	1024
Attention Head	8
Dropout	0.2
Class Weight	True
Total Parameters	6,414,932

Model	Micro F1	Macro F1	Accuracy
NF-UNSW-NB15-v2	0.9923	0.7779	99.23%
NF-CIC-IDS2018-v2	0.9951	0.7316	99.51%

Challenges & Solutions

1. Severe Class Imbalance

- Problem: 96% and 88% benign traffic
- Solution:
    - Computed balanced class weights
    - Used appropriate metrics (F1, precision/recall per class)

```
class_weights = None

if class_weight:
    classes = np.unique(y[np.where(train_mask)[0]])
    class_weights = compute_class_weight("balanced",
                                         classes=classes,
                                         y=y[np.where(train_mask)[0]])
    class_weights = dict(zip(classes, class_weights))

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    steps_per_epoch=len(train_idx)//batch_size,
    callbacks=[early_stop, lr_scheduler],
    class_weight=class_weights,
    verbose=1)
```

```
=== TRIAL 1 ===
# CONFIGURATION
window_size = 4
batch_size = 256
epochs = 300
n_layers = 2
embed_size = 64
internal_size = 128
n_heads = 2
dropout = 0.2
class_weight = True
```

```
[Worms]
-- TP: 20, FP: 535, TN: 357983, FN: 4
-- Precision: 0.0360 (20 / 555)
-- Recall: 0.8333 (20 / 24)
-- Specificity: 0.9985 (357983 / 358518)
-- Accuracy: 0.9985
-- F1-score: 0.0691
-- Balanced Accuracy: 91.5921
```

```
=== TRIAL 2 ===
# CONFIGURATION
window_size = 4
batch_size = 256
epochs = 300
n_layers = 2
embed_size = 64
internal_size = 128
n_heads = 2
dropout = 0.2
class_weight = False *
```

```
[Worms]
-- TP: 0, FP: 0, TN: 358518, FN: 24
-- Precision: 0.0000 (0 / 0)
-- Recall: 0.0000 (0 / 24)
-- Specificity: 1.0000 (358518 / 358518)
-- Accuracy: 0.9999
-- F1-score: 0.0000
-- Balanced Accuracy: 50.0000
```

Challenges & Solutions

2. Computational Resources

- Problem: Large datasets (2.39M and 18.89M flows), memory-intensive sequences
- Solution:
  - Only use 3M flows from CSE-CIC-IDS2018
    - NF-UNSW-NB15-v2-pre.csv: 3.8 GB
    - NF-CSE-CIC-IDS2018-pre.csv: 5.1GB
  - Used NumPy's *sliding_window_view* to efficiently create windows without copying data
  - GPU acceleration

```python
# Functions define #

def NetFlowSlicing(df, window_size:int=16):
    array = df.to_numpy(dtype=np.float32)
    features = array[:, :-1]
    labels = array[:, -1].astype(np.int32)

    # https://numpy.org/devdocs/reference/generated/numpy.lib.stride_tricks.sliding_window_view.html
    X = np.lib.stride_tricks.sliding_window_view(features, (window_size, features.shape[1]))
    X = X.reshape(-1, window_size, features.shape[1])
    y = labels[window_size - 1:]

    return X, y
```

Conclusions

1. Sequential modeling significantly improves detection

- BaselineFCN (50.58%) → CNN+LSTM (64.45%) → NetFlowBERT (83.44%)
- Improved accuracy by 27% to 65% over the non-sequential baseline

2. NetFlowBERT achieves best overall performance

- Highest accuracy (83.44%) and Macro F1 (0.6389) on CIC-IDS2018
- Excels on DoS attacks (average F1 = 0.8925)
- Bidirectional attention effectively captures temporal attack patterns

3. Flow-based approach is effective for modern networks

- Works with encrypted traffic (no payload inspection)
- Privacy-preserving and scalable

4. Trade-offs exist between models

- Performance: NetFlowBERT > CNN+LSTM > BaselineFCN
- Speed: BaselineFCN > CNN+LSTM > NetFlowBERT
- Model choice depends on deployment requirements

Future Work

1. Combine multiple models for better accuracy
2. Train on complete CIC-IDS2018 datasets with better GPU
3. Improving low Macro F1 caused by imbalance using oversampling techniques such as SMOTE
4. A predictive model for all NetFlow format data

References

[1] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, ‘NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems’, in *Big Data Technologies and Applications*, Springer International Publishing, 2021, pp. 117–135.

[2] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, ‘FlowTransformer: A transformer framework for flow-based network intrusion detection systems’, *Expert Systems with Applications*, vol. 241, p. 122564, May 2024.

[3] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, and J. Chen, 'DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system,' Security and Communication Networks, vol. 2020, Article ID 8890306, 2020.

[4] M. S. Sakib and N. Tabassum, 'Analyzing Deep Learning Model Performance for Intrusion Detection on CIC-IDS2017 Dataset,' SSRN Electronic Journal, April 2025.

[5] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems," in Big Data Technologies and Applications, Springer, 2021, pp. 117-135.