# BiGRU for Network Intrusion Detection – Complete Methodology

## 1. Understanding BiGRU: Theoretical Foundations

BiGRU is an extension of the Gated Recurrent Unit (GRU), which processes data in both forward and backward directions, making it highly effective for analyzing time-series data such as network intrusion datasets (e.g., CIC-IDS2017, UNSW-NB15, or NSL-KDD).

### 1.1 GRU Basics

Each GRU cell consists of:

- Update Gate (z\_t): Determines how much past information should be carried forward.

- Reset Gate (r\_t): Decides how much past information should be ignored.

- Candidate Activation (h̃\_t): Generates new information based on input and previous state.

- Hidden State (h\_t): The final state that updates based on the update gate.

### Mathematical Formulation

Update Gate: z\_t = σ(W\_z · x\_t + U\_z · h\_{t-1} + b\_z)

Reset Gate: r\_t = σ(W\_r · x\_t + U\_r · h\_{t-1} + b\_r)

Candidate Activation: h̃\_t = tanh(W\_h · x\_t + r\_t ⊙ (U\_h · h\_{t-1}) + b\_h)

Final Hidden State Update: h\_t = (1 - z\_t) ⊙ h\_{t-1} + z\_t ⊙ h̃\_t

### 1.2 Bidirectional GRU (BiGRU)

Unlike a standard GRU, BiGRU consists of two GRU layers:

- Forward GRU processes data from past to future.

- Backward GRU processes data from future to past.

The final output is the concatenation of the forward and backward hidden states.

## 2. BiGRU Architecture for Network Intrusion Detection

### 2.1 Preprocessing the Network Intrusion Dataset

Most network datasets (CIC-IDS2017, UNSW-NB15) contain features such as source/destination IP, ports, protocols, packet length, flow duration, and attack type. Preprocessing involves:

- Converting categorical features to numerical.

- Feature selection using PCA or mutual information.

- Normalization using MinMaxScaler or Z-score.

- Splitting dataset into training (80%) and testing (20%).

### 2.2 BiGRU Model Architecture

The BiGRU model consists of:

1. Input Layer

2. Embedding Layer (Optional)

3. Bidirectional GRU Layer

4. Dropout Layer

5. Fully Connected Dense Layer

6. Softmax/Sigmoid Activation for classification

## 3. Model Implementation (Python + TensorFlow/Keras)

Below is a Python implementation of BiGRU for intrusion detection using TensorFlow/Keras.

import numpy as np  
import pandas as pd  
import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense, Dropout, Bidirectional, GRU  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
  
# Load the dataset  
data = pd.read\_csv("CIC-IDS2017.csv")  
  
# Preprocessing  
X = data.drop(columns=['Label'])  
y = data['Label'].map(lambda x: 1 if x == 'Attack' else 0)  
  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
# Define BiGRU Model  
model = Sequential([  
 Bidirectional(GRU(64, return\_sequences=True), input\_shape=(X\_train.shape[1], 1)),  
 Dropout(0.3),  
 Bidirectional(GRU(32)),  
 Dropout(0.3),  
 Dense(16, activation='relu'),  
 Dense(1, activation='sigmoid')  
])  
  
# Compile Model  
model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])  
  
# Train Model  
model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(X\_test, y\_test))  
  
# Evaluate Model  
test\_loss, test\_acc = model.evaluate(X\_test, y\_test)  
print(f"Test Accuracy: {test\_acc:.4f}")

## 4. Performance Metrics

To evaluate BiGRU’s effectiveness, consider metrics such as:

- Accuracy

- Precision, Recall, F1-score

- AUC-ROC Curve

- Confusion Matrix

from sklearn.metrics import classification\_report, roc\_auc\_score  
  
y\_pred = (model.predict(X\_test) > 0.5).astype("int32")  
  
print(classification\_report(y\_test, y\_pred))  
print(f"AUC-ROC Score: {roc\_auc\_score(y\_test, y\_pred):.4f}")

## 5. Summary of BiGRU for Network Intrusion Detection

Key steps involved in BiGRU for intrusion detection:

|  |  |
| --- | --- |
| Step | Description |
| Preprocessing | Feature selection, encoding, normalization |
| Model | BiGRU with dropout and dense layers |
| Activation | Sigmoid for binary classification |
| Loss Function | Binary Cross-Entropy |
| Optimizer | Adam |

## 6. Next Steps

- Fine-tune Hyperparameters (Number of GRU units, Dropout rates, Learning rates).

- Experiment with Attention Mechanisms (Self-Attention on BiGRU).

- Apply to Multi-Class Intrusion Detection (Detect multiple attack types).