Neural Network-Based Intrusion Detection System

Submitted to: Prof. David Elizondo

Babu Pallam (P2849288)
MSc. Artificial Intelligence

CSIP5103 – Neural Systems and NLP
November 1, 2024

Project Objective and Steps

- **Objective:** Develop and test a neural network model to detect network intrusions.
- **Goal:** The model should distinguish between malicious (intrusive) and normal connections.
- **Key Steps:**
    - Define the problem and analyze the dataset.
    - Preprocess the KDD dataset for model readiness.
    - Design a feedforward neural network architecture.
    - Train the model and evaluate its effectiveness.
    - Analyze results and assess model performance.

Dataset Overview

- **KDD Cup 1999 Dataset:[2][3]**
  - A benchmark dataset for testing intrusion detection systems.
  - Provides labeled data with both normal and attack connections.
- **Classes:**
  - **Total Classes:** 22 attack types plus normal connections.
  - **Total Features:** 41 network features, including:
    - **Categorical Features:** Protocol type, service type, etc.
    - **Continuous Features:** Duration, connection count, etc.
- **Challenges:**
  - **Class Imbalance:** Over-representation of certain attack types compared to others and normal traffic.
  - **Feature Variety:** Combination of categorical and continuous features requires careful preprocessing.

Data Preparation

- **Data Loading:**
  - Load and explore the dataset to understand its structure and content.
- **Preprocessing:**
  - **Encoding:** Convert categorical features into numerical values for model compatibility.
  - **Shuffling:** To introduce randomness and reduce order bias.
  - **Handling Missing Values:** To ensure data integrity.
  - **Normalization:** Scale for balanced and stable training.
  - **Binary Transformation:** Multi-class labels into binary for simplified modeling.
- **Data Splitting:**
  - **Train-Test Split:** Separate data into training and testing sets.
  - **Cross-Validation:** Apply cross-validation for robust performance estimation.

Neural Network Creation

- **Architecture:** Feedforward Neural Network Architecture
- **Hyperparameter Selection:** Define optimal hyperparameters based on dataset characteristics
- **Tuning Algorithm:** Bayesian Optimization
- **Hyperparameter Search Space:**

```matlab
% Define optimization variables for hyperparameter tuning

optimVars = [
    optimizableVariable('learningRate', [0.001, 0.1], 'Transform', 'log'),  % Continuous range for learning rate (log scale)
    optimizableVariable('momentum', [0.5, 0.9]),                            % Continuous range for momentum
    optimizableVariable('numLayers', [1, 3], 'Type', 'integer'),            % Integer range for the number of hidden layers
    optimizableVariable('numNeurons', [10, 50], 'Type', 'integer'),         % Integer range for neurons per layer
    optimizableVariable('epochs', [10, 50], 'Type', 'integer'),             % Integer range for the number of epochs
    optimizableVariable('activationFunction', {'logsig', 'tansig', 'purelin'}, 'Type', 'categorical'), % Activation function options
    optimizableVariable('performFcn', {'mse', 'crossentropy'}, 'Type', 'categorical'), % Performance function options
    optimizableVariable('trainingFunction', {'traingd', 'trainlm', 'trainbfg', 'trainrp', ... % Training functions
                                             'trainscg', 'traincgb', 'traincgf', 'traincgp', 'trainoss', 'traingdx'}, ...
                                             'Type', 'categorical')
];
```

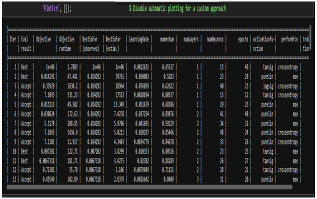
1. Hyperparameter Search Space

Neural Network Creation

- **Handling Overfitting**
  1. **Cross-Validation:** Used to fine-tune hyperparameters and enhance model robustness.
  2. **Early Stopping:**
- **Loss Handling**
  1. **MSE:** Used primarily for **regression tasks** where the output is continuous
  2. **CrossEntropy**: Commonly used for **classification tasks**, particularly where outputs are probabilities over classes.

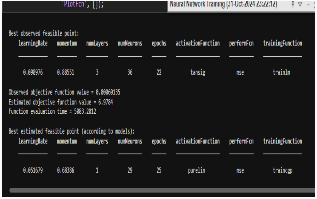
Results and Insights

These are the progress of the result obtained during different state of Neural Network Model construction


'PlotFcn', []); % Disable automatic plotting for a custom approach

Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	learningRate	momentum	numLayers	numNeurons	epochs	activationFu- nction	performFcn	trai tior
1	Best	1e+06	1.7803	1e+06	1e+06	0.0012631	0.65517	1	13	49	tansig	crossentropy	
2	Best	0.014292	47.441	0.014292	39761	0.050083	0.5283	1	23	28	purelin	mse	
3	Accept	0.33929	1630.2	0.014292	28964	0.076839	0.62622	1	40	23	logsig	crossentropy	
4	Accept	7.1093	531.23	0.014292	17553	0.0028834	0.80537	1	15	12	tansig	crossentropy	
5	Accept	0.015113	49.502	0.014292	13.349	0.051679	0.68386	1	29	25	purelin	mse	
6	Accept	0.030824	172.63	0.014292	7.4278	0.037254	0.89876	2	41	48	purelin	mse	
7	Accept	5.5178	208.85	0.014292	5.9706	0.041691	0.59129	3	34	32	purelin	mse	
8	Accept	7.1093	1436.9	0.014292	5.0211	0.010297	0.85446	2	48	24	purelin	crossentropy	
9	Accept	7.1101	11.917	0.014292	4.3403	0.0034779	0.84678	2	33	26	purelin	crossentropy	
10	Best	0.007202	122.71	0.007202	3.8299	0.010333	0.80526	2	23	25	tansig	mse	
11	Best	0.0067318	181.71	0.0067318	3.4275	0.02382	0.88289	2	26	27	tansig	mse	
12	Accept	0.71302	35.78	0.0067318	3.106	0.0079049	0.71151	2	28	21	tansig	crossentropy	
13	Accept	0.45549	102.09	0.0067318	2.8379	0.0026642	0.8409	3	31	20	purelin	mse	

2. Hyperparameter Tuning Results


'PlotFcn', []);

Neural Network Training (31-Oct-2024 23:22:12)

Best observed feasible point:

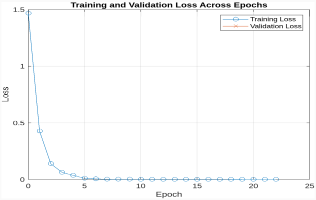
learningRate	momentum	numLayers	numNeurons	epochs	activationFunction	performFcn	trainingFunction
0.098976	0.88551	3	36	22	tansig	mse	trainlm

Observed objective function value = 0.00060135
Estimated objective function value = 6.9784
Function evaluation time = 5083.2012

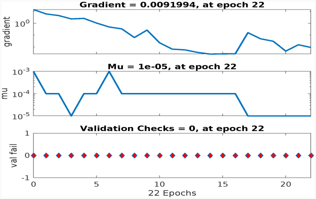
Best estimated feasible point (according to models):

learningRate	momentum	numLayers	numNeurons	epochs	activationFunction	performFcn	trainingFunction
0.051679	0.68386	1	29	25	purelin	mse	traincgp

3. Best Hyperparameter Received

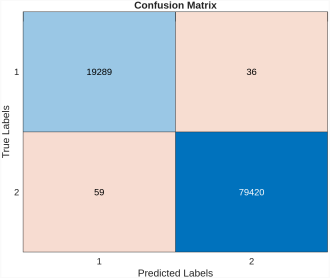


4. Best Model Performance



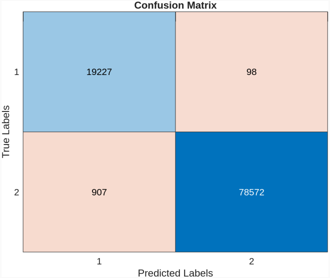
5. Best Model Training State

Results and Insights



6. hyperparameter-best-observed

- **Accuracy:** 99.88%
- **Precision:** 99.81%
- **Recall:** 99.69%
- **F1 Score:** 99.75%



6. hyperparameter-best-feasible

- **Accuracy:** 98.85%
- **Precision:** 99.49%
- **Recall:** 95.49%
- **F1 Score:** 97.12%

Further Experiments

1. Prediction with the entire dataset **kddcup.data.gz** [2] for Testing the Best Final Model (as unseen data)
    - Impressive Performance has been Noted
2. Binary to Multi Class with the Best Final Model
    - Performance Observed and Challenges Exist has been Verified
3. Bayesian Optimization without Cross Validation
    - Compared the result with Previous Optimization Result
    - Observation has been noted

Conclusion and Contributions

- **Summary of Findings:**
  - Feedforward neural network effectively detects intrusions.
  - Satisfactory precision and recall for intrusion detection.
  - Effective control over overfitting.
- **Contributions:** Showcases practical intrusion detection and insights on feedforward network limitations for imbalanced data.
- **Future Work:**
  - Explore advanced architectures (CNNs, RNNs).
  - Address class imbalance with techniques like SMOTE.
  - Further hyperparameter tuning, including Batch Size, Momentum, Weight Initialization, Dropout Rate.
  - Alternative tuning methods:
    - Grid Search, Random Search, Evolutionary Algorithms, Simulated Annealing, Particle Swarm Optimization.

References

- MATLAB. (2023). *MATLAB and Simulink* [Software]. MathWorks, Inc. Retrieved from https://www.mathworks.com/products/matlab.html
- KDD Cup 1999 Dataset. (1999). *The UCI KDD Archive: Full Dataset*. Information and Computer Science, University of California, Irvine. Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data.gz (File name: kddcup.data.gz)
- KDD Cup 1999 Dataset. (1999). *The UCI KDD Archive: 10% Sample Dataset*. Information and Computer Science, University of California, Irvine. Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz (File name: kddcup.data$_{1}0_{p}ercent.gz$)