Bayesian Neural Networks

Diego Cesar Villa Almeyda

Master of Science

August, 2025

School of Mathematics
The University of Edinburgh

Dissertation Presented for the Degree of MSc in Statistics with Data Science

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

Executive summary

1 Introduction

This is a book created from markdown and executable code.

2 Methods

2.1 The UNSW-NB15 Network Dataset

The UNSW-NB15 dataset (1,2) was created to overcome the limitations of earlier benchmark datasets such as KDD99 and NSL-KDD, which have been criticised for outdated attack types, unrealistic normal traffic, and inconsistent distributions between training and testing sets. In contrast, UNSW-NB15 combines modern real-world network activity with synthetically generated attack behaviours, making it highly suitable for evaluating contemporary Network Intrusion Detection Systems (NIDSs). The dataset contains 49 features encompassing both flow-level host interactions and deep packet inspection metrics, enabling effective discrimination between normal and malicious traffic. It includes nine categories of contemporary cyberattacks alongside updated profiles of normal network behaviour. Statistically, UNSW-NB15 is more complex than its predecessors (2).

The full dataset comprises 2,540,044 records, of which 2,218,761 (approximately 87%) correspond to normal traffic, resulting in a highly imbalanced class distribution that reflects real-world network conditions (3). The training and testing subsets were obtained directly from the UNSW website (4), consisting of 175,341 and 82,332 records, respectively. Statistical analysis has demonstrated that the training and test sets share similar non-linear and non-normal feature distributions. Furthermore, high statistical correlation between the two sets supports their appropriateness as benchmark data for evaluating statistical and machine learning models tasked with distinguishing complex attack patterns from normal traffic (2). Non-informative features were excluded from the distributed datasets, yielding a total of 42 usable predictors and two target variables: `label` (binary attack indicator) and `attack_cat` (attack category), as described in Table S4.1.

An initial examination of the training dataset revealed that it contains a disproportionate number of attack records (68.06%) compared to normal traffic (31.93%), which does not reflect realistic conditions. To create a more representative imbalanced subset for our analysis, we retained only the normal traffic and denial-of-service (DoS) attack instances. This resulted in a subset with 82.03% normal and 17.97% DoS traffic, closely aligning with the class distribution in the full dataset.

2.2 Feature Engineering

Exploratory data analysis was conducted to assess the quality and distribution of the features. One of the first issues identified was with the features `is_ftp_login` and `ct_ftp_cmd`, which were found to be identical in the training dataset and contained integer values ranging from 0 to 4. This was unexpected, as `is_ftp_login` is defined as a binary variable in the data dictionary, suggesting possible data corruption. Given that the observed values appeared to align with the definition of `ct_ftp_cmd`, we chose to drop `is_ftp_login` from the training set. Interestingly, this anomaly was not observed in the test set, where the two features differed as expected.

The nominal features `proto`, `service`, and `state` had 133, 13, and 9 unique levels, respectively. However, only a small subset of these levels accounted for the vast majority of records in the training data. To reduce dimensionality and improve model interpretability, we grouped the infrequent levels in each feature into a single "other" category. We applied a Pareto principle approach, retaining the levels that together covered approximately 90% of the records. After grouping, `proto` was reduced to 5 levels, `service` to 4, and `state` to 4, resulting in a more manageable set of categories for downstream modelling.

Most numeric features exhibited strong right-skewness, with a pronounced peak at zero, likely indicating unsuccessful or dropped connections. Further inspection revealed that several variables take values from a limited set of predefined ranges. For example, `sttl` and `dttl`, which represent source and destination time-to-live (TTL) values, frequently appeared near 0, 30, 60, and 250. According to (5), typical initial TTL values are 64, 128, and 255, which gradually decrease during transmission—consistent with the observed values, along with the additional cluster near 30. As these values reflect discrete categories rather than continuous magnitudes, we recoded them as ordinal variables with levels "0", "~30", "~64", and "~255". Similarly, the features `swin` and `dwin` predominantly took values of 0 and 255, with other values appearing very infrequently (often only once). This pattern suggested that 0 and 255 may also be predefined values. Accordingly, we discretised these numeric features into the nominal levels "0", "255", and "rare".

Before feature selection and model building, nominal features were one-hot encoded, ordinal features were mapped to integers, and numeric features were log-transformed and scaled using a robust scaler. The robust scaler subtracts the median and divides by the interquartile range (IQR), making it more suitable for highly skewed data than standard scaling, which assumes a roughly symmetric distribution. Because most numeric features were heavily skewed and included zero values, we explored various log-based transformations. For features with zeros, we tested log transformations with two offsets: adding 1, and adding half the smallest non-zero value, as suggested by (6). We also tested the Yeo–Johnson transformation. Visual inspection showed that the latter offset approach provided the best normalisation, so we applied it to features containing zeros, while using the standard log transformation for strictly positive features.

2.3 Feature Selection

To reduce redundancy and retain the most informative features, we applied a two-stage feature selection strategy combining correlation analysis and model-based permutation importance.

First, after preprocessing (see earlier section), we computed the Spearman correlation matrix for the numeric features in the training set. Feature pairs with a high correlation (0.9) were flagged, and for each pair, we retained the feature with the higher mutual information (MI) score relative to the binary target variable. MI measures the amount of shared information between two variables, capturing any form of statistical dependency—beyond linear correlation—and is particularly well-suited for assessing relationships between a discrete and a continuous variable (7). MI was estimated using a nearest-neighbour-based method, which avoids the resolution loss associated with binning and provides a more accurate, non-parametric measure of association (8). This filtering step helped reduce redundancy while preserving features most informative for the classification task.

In the second stage, we employed a random forest (RF) classifier to assess feature relevance using permutation importance. The training data was split into a sub-training and validation set using stratified sampling to preserve the class distribution, allocating 20% of the data for validation. A model was then trained using a grid search with 5-fold stratified cross-validation, optimising the F1 score. Although this was not the final predictive model, we carefully selected the hyperparameter grid to reduce overfitting. Specifically, we tuned the number of trees ({1000, 1500}), maximum tree depth ({3, 5}), and minimum number of samples required to split an internal node ({10, 15}) (9). In addition, we addressed class imbalance by applying class weights in the learning algorithm, using the "balanced" scheme, which assigns weights inversely proportional to class frequencies (9). After selecting the best-performing model, permutation importance was computed on the validation set by measuring the average decrease in F1 score when each feature was randomly shuffled across 10 repetitions. To further simplify the feature space, we grouped the importance scores of one-hot encoded variables by their original categorical variable (e.g., all `proto_*` columns were aggregated under `proto`). We then selected the top ten base features and retained all corresponding encoded columns, yielding a compact and interpretable set of predictors for subsequent modelling.

2.4 Bayesian Neural Networks (BNN)

Neural networks (NNs) are hierarchical models composed of an input layer, one or more hidden layers, and an output layer, where each layer consists of units that perform a linear transformation followed by a non-linear activation function (10). Training a neural network involves finding the set of weights and biases at the hidden and output layers that minimise a specified loss function, typically using gradient-based optimisation algorithms, such as stochastic gradient descent (SGD), and backpropagation (11).

Formally, given an input vector $\mathbf{x} \in \mathbb{R}^n$, a neural network with L hidden layers of widths $H_1, \dots, H_L$, and a non-linear activation function $\phi : \mathbb{R} \to \mathbb{R}$, the computations at layer l ($l = 1, \dots, L$) are:

$$\begin{aligned}\mathbf{g}^{(l)}(\mathbf{x}) &= \mathbf{w}^{(l)}\mathbf{h}^{(l-1)}(\mathbf{x}) + \mathbf{b}^{(l)}\\ \mathbf{h}^{(l)}(\mathbf{x}) &= \phi\left(\mathbf{g}^{(l)}(\mathbf{x})\right),\end{aligned}$$

where $\mathbf{w}^{(l)}$ is the weight matrix of dimensions $H_l \times H_{l-1}$, $\mathbf{b}^{(l)}$ is a bias vector of length H_l, $\mathbf{h}^{(l-1)}(\mathbf{x})$ denotes the post-activation values of the previous layer (with $\mathbf{h}^{(0)} = \mathbf{x}$), and $\mathbf{g}^{(l)}(\mathbf{x})$ are the pre-activation values. For the output layer, the pre-activation values are computed as $\mathbf{g}^{(L+1)}(\mathbf{x}) = \mathbf{w}^{(L+1)}\mathbf{h}^{(L)}(\mathbf{x}) + \mathbf{b}^{(L+1)}$, and the activation function for the output layer is chosen to match the distribution of the target variable; for example, a sigmoid function for a Bernoulli-distributed binary outcome $y \in 0, 1$. While ϕ is often fixed across layers, it may vary depending on the network architecture or specific application (10).

A widely used loss function for binary classification is the binary cross-entropy (BCE), also known as log loss. Let $\mathbf{w}$ denote the set of all parameters (weights and biases) in the neural network, and let $f(\mathbf{x}_i; \mathbf{w})$ represent the predicted probability output by the network for input $\mathbf{x}_i$. Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $y_i \in \{0, 1\}$, the BCE is defined as

$$J(\mathbf{w}) = -\sum_{i=1}^{N} \left[y_i \log f(\mathbf{x}_i; \mathbf{w}) + (1 - y_i) \log\left(1 - f(\mathbf{x}_i; \mathbf{w})\right)\right].$$

Training a NN for binary classification therefore amounts to finding the parameter set $\mathbf{w}$ that minimises this loss:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w}).$$

This optimisation is typically performed using SGD, where gradients are estimated at each iteration using randomly selected subsets of the data, known as mini-batches (10). Because the optimiser does not need to process the entire dataset at each step, but only a small random portion of it, SGD is particularly well suited for large-scale datasets. Another widely used stochastic optimiser is Adam (adaptive moment estimation), which extends SGD by incorporating adaptive learning rates and momentum terms. Adam is computationally efficient, requires minimal memory, and often converges faster in practice (12).

2.4.1 Priors

2.4.2 Inference Methods

2.4.2.1 Markov Chain Monte Carlo (MCMC)

2.4.2.2 Variational Inference (VI)

2.4.3 Convergence

2.5 Model Benchmarking

2.5.1 Prediction Accuracy

2.5.2 Calibration

2.5.3 Running Time

2.6 Interpretability Analysis

3 Results

4 Conclusions

References

1. Moustafa N, Slay J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 military communications and information systems conference (MilCIS). IEEE; 2015. p. 1–6.

2. Moustafa N, Slay J. The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Information Security Journal: A Global Perspective. 2016;25(1-3):18–31.

3. Zoghi Z, Serpen G. UNSW-NB15 computer security dataset: Analysis through visualization. arXiv preprint arXiv:210105067. 2021;

4. University of New South Wales (UNSW). UNSW-NB15 dataset [Internet]. 2015. Available from: https://research.unsw.edu.au/projects/unsw-nb15-dataset

5. Imperva. Time to live (TTL) [Internet]. n.d. Available from: https://www.imperva.com/learn/performance/time-to-live-ttl/

6. Hyndman RJ. Transforming data with zeros [Internet]. 2013. Available from: https://robjhyndman.com/hyndsight/transformations/

7. Ross BC. Mutual information between discrete and continuous data sets. PloS one. 2014;9(2):e87357.

8. scikit-learn Developers. Scikitlearn: mutual_info_classif. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html; 2025.

9. scikit-learn Developers. Scikitlearn: RandomForestClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html; 2025.

10. Arbel J, Pitas K, Vladimirova M, Fortuin V. A primer on bayesian neural networks: Review and debates. arXiv preprint arXiv:230916314. 2023;

11. Jospin LV, Laga H, Boussaid F, Buntine W, Bennamoun M. Hands-on bayesian neural networks—a tutorial for deep learning users. IEEE Computational Intelligence Magazine. 2022;17(2):29–48.

12. Adam KDBJ et al. A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014;1412(6).

Appendix

Supplementary Tables

Name	Type	Description
id	Integer	Record ID.
dur	Float	Record total duration.
proto	Nominal	Transaction protocol.
service	Nominal	Such as http, ftp, smtp, ssh, dns and ftp-data.
state	Nominal	Indicates to the state and its dependent protocol (such as ACC, CLO and CON).
spkts	Integer	Source to destination packet count.
dpkts	Integer	Destination to source packet count.
sbytes	Integer	Source to destination transaction bytes.
dbytes	Integer	Destination to source transaction bytes.
rate	Float	Ethernet data rates transmitted and received.
sttl	Integer	Source to destination time to live value.
dttl	Integer	Destination to source time to live value.
sload	Float	Source bits per second.
dload	Float	Destination bits per second.
sloss	Integer	Source packets retransmitted or dropped.
dloss	Integer	Destination packets retransmitted or dropped.
sinpkt	Float	Source interpacket arrival time (mSec).
dinpkt	Float	Destination interpacket arrival time (mSec).
sjit	Float	Source jitter (mSec).
djit	Float	Destination jitter (mSec).
swin	Integer	Source TCP window advertisement value.
stcpb	Integer	Source TCP base sequence number.
dtcpb	Integer	Destination TCP base sequence number.
dwin	Integer	Destination TCP window advertisement value.
tcprtt	Float	TCP connection setup round-trip time, the sum of synack and ackdat.
synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
smean	Integer	Mean of the flow packet size transmitted by the src.
dmean	Integer	Mean of the flow packet size transmitted by the dst.
trans_depth	Integer	Represents the pipelined depth into the connection of http request/response transaction.
response_body_len	Integer	Actual uncompressed content size of the data transferred from the server's http service.
ct_srv_src	Integer	No. of connections that contain the same service and source address in 100 connections according to the last time.
ct_state_ttl	Integer	No. for each state according to specific range of values for source/destination time-to-live.
ct_dst_ltm	Integer	No. of connections of the same destination address in 100 connections according to the last time.
ct_src_dport_ltm	Integer	No of connections of the same source address and the destination port in 100 connections according to the last time.
ct_dst_sport_ltm	Integer	No of connections of the same destination address and the source port in 100 connections according to the last time.
ct_dst_src_ltm	Integer	No of connections of the same source and the destination address in in 100 connections according to the last time.
is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
ct_ftp_cmd	Integer	No of flows that has a command in ftp session.
ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
ct_src_ltm	Integer	No. of records of the srcip in 100 records according to the ltime.
ct_srv_dst	Integer	No. of connections that contain the same service and destination address in 100 connections according to the last time.
is_sm_ips_ports	Binary	If source and destination IP addresses equal and port numbers equal then, this variable takes value 1 else 0.
attack_cat	Nominal	The name of each attack category.
label	Binary	Normal (0) or attack (1) label.

Table S4.1