

msg Plaut - Tecthoughter Spring 2026

Network Anomaly Detection - Solution Documentation

Áron Kovács, Gábor Funk

April 2026

1 Initial State and Bug Discovery

During the review of the original notebook, several critical errors and areas for optimization were identified that prevented the model from learning effectively and hindered the measurement of real performance:

1. **Incorrect Hyperparameters:** The model was set to train for only 1 epoch, which is insufficient for convergence. Additionally, the optimizer was set to a non-existent algorithm named 'eva'.
2. **Data Leakage:** Feature scaling was not properly separated between the training and test datasets.
3. **Architectural Flaws:** The output layer of the network used a ReLU activation function. This approach is mathematically incorrect for binary classification tasks.
4. **Data Split Ratios:** The 60% testing ratio (`test_size=0.6`) was too high, significantly reducing the amount of data available for training.
5. **Lack of Metrics:** The target F1-score was not properly measured or implemented during evaluation.

2 Fixes and Optimization Decisions

- **[Decision] Choosing Binary Classification over Multiclass**

The provided 20 MB dataset contains a total of 23 different network categories; however, their distribution is extremely imbalanced. While there are 67,343 samples of *normal* traffic and 41,214 of *neptune* attacks, the other end of the spectrum contains classes with almost no representation (e.g., *phf* - 4 samples, *perl* - 3 samples, *spy* - 2 samples).

- **[Optimization] More Robust Predictions**

If we had attempted to train the model to recognize all 23 categories (multiclass classification), the network would have been unable to learn generalizable patterns from classes with so few samples. These noisy, under-learned categories would have drastically lowered the overall score (especially the macro F1 score required for the task). Therefore, we decided to simplify the problem: by merging all attack types into a single *anomaly* class, we performed binary classification. This makes the model much more stable and reliable in a real-world Anomaly Detection (IDS) environment.

- **[BugFix] Correction of Training Parameters**

The number of epochs was initially set to 1.

- **[Fix] How we fixed it**

We increased the number of epochs to 15.

- **[Optimization] Why the fix works**

A higher number of epochs allows the network to iterate over the training data multiple times. This ensures that weight fine-tuning occurs, the error value decreases, and underfitting is avoided.

- **[BugFix] Data Preparation and Splitting**

The model lacked proper binary encoding for the target variable (y), and the original testing ratio was too high.

- **[Fix] Logical Pipeline and Data Splitting**

We introduced binary encoding for the target variable (normal: 0, anomaly: 1) and removed redundant features ('label', 'difficulty'). The testing ratio was reduced to 20%. By applying the 'stratify=y' parameter, we ensured that the ratio of normal to anomaly traffic remained consistent across the split sets.

- **[Optimization] Feature Encoding**

Using the pandas 'get_dummies' method, we converted categorical variables (protocol_type, service, flag) into binary vectors, resulting in a numerical format interpretable by the model.

- **[BugFix] Data Leakage during Scaling**

Scaling occurred before the data split, meaning the statistics of the test data influenced the training process beforehand.

- **[Fix] Consistent Application of StandardScaler**

We applied the 'fit_transform' method of StandardScaler exclusively to the training data (X_{train}), while only performing the transformation on the test data (X_{test}) based on the learned statistics.

- **[Optimization] Generalization Capability**

This method guarantees that the model encounters completely unseen data during testing, making the resulting performance metrics realistic and reliable in a real environment.

- **[BugFix] Network Architecture Errors and Dimensions**

The output layer was configured for a multiclass task, and the specified 'eva' optimizer was invalid.

- **[Fix] Rebuilding Structure and Dimensions**

The model was converted for binary classification: the output layer received 1 neuron and 'Sigmoid' activation. In the hidden layers, we designed a funnel-like architecture with 64, then 32 neurons. Since the input dimension increased significantly after data preparation (one-hot encoding) to over 100 features, the first 64-neuron layer provides

sufficient capacity to learn complex, non-linear network patterns. The subsequent 32-neuron layer functions as information compression and dimensionality reduction. This gradual narrowing helps the model emphasize the most important abstractions while avoiding learning unnecessary noise. To further prevent overfitting, we applied Dropout layers at ratios of 0.3 and 0.2 after the 'ReLU' activation layers.

- **[Optimization] Optimizer and Loss Function**

The optimizer was replaced with the industry-standard **Adam** (Adaptive Moment Estimation) algorithm. The great advantage of Adam is that it calculates the learning rate adaptively for each parameter based on previous moments of the gradients. This results in faster and more stable convergence, as it automatically handles sparse gradients and noisy data.

The **Loss** function of the network was set to *binary_crossentropy*. This metric determines how much the model's predictions deviate from the true labels. Since we are performing binary classification (0 or 1), the Sigmoid function at the output of the network returns a probability value. Binary crossentropy is specifically designed for this: it punishes errors logarithmically and strictly where the model predicts the wrong class with high confidence, thus effectively guiding the Adam optimizer toward finding the correct weights.

3 Results

The fine-tuned machine learning model achieved outstanding performance on independent test data. On the 25,195 test samples, the network showed an accuracy of 0.99. The model successfully exceeded the required macro F1 score of at least 0.75, reaching a macro F1 value of 0.99. For both normal and anomaly traffic, precision and recall values were near perfect (0.99).

The final model ('msg_network_model.keras') and the scaler object ('msg_scaler.pkl') were successfully saved for future deployment or prediction tasks.