Beyond All-Pairs Cosine: Leveraging GNN Smoothing for Enhancing CNN-Based Flow Correlation Attacks in Tor

Gurjot Singh * [1]

Abstract

Traffic correlation attacks remain a fundamental threat to the anonymity of users in networks such as Tor. Recent advancements, particularly DeepCoFFEA, have demonstrated state-of-the-art performance in correlating network flows by leveraging feature embedding and amplification techniques. While DeepCoFFEA achieves high correlation accuracy, further refinement can enhance its precision by reducing false positives without compromising true positive retention. In this work, we propose a Graph Neural Network-based post-processing framework to refine DeepCoFFEA's correlation predictions. Our approach constructs a graph representation of correlated flows, allowing for structural feature propagation and improved decision-making. Experimental results indicate that our GNN model achieves a Bayesian Detection Rate of 42.23%, compared to 40.99% for DeepCoFFEA, demonstrating an improvement in precision. These findings suggest that incorporating graph-based learning techniques can enhance the effectiveness of flow correlation attacks.

1. Introduction

The Tor network provides a critical layer of privacy by anonymizing internet traffic through a series of encrypted relays. This ensures that an observer monitoring traffic at any point cannot directly link a user's source IP address to their destination. As illustrated in Figure 1, Tor achieves anonymity by forwarding packets through multiple relays before reaching their final destination, making direct tracking significantly harder. However, it remains vulnerable to **traffic correlation attacks**, where an adversary with access to both the entry and exit points of the network attempts to

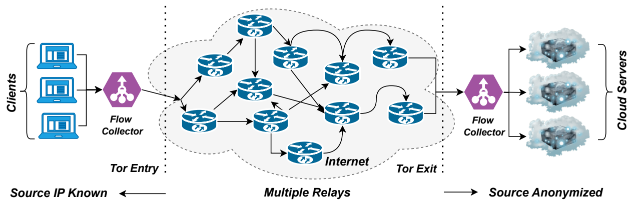

Figure 1. Anonymity in Tor: The source IP is anonymized by forwarding traffic through multiple relays, preventing direct observation of the user's origin.

match incoming and outgoing traffic flows to de-anonymize users. These attacks exploit statistical patterns in packet timing, volume, and sequence to establish a correlation between Tor entry and exit traffic. As a result, even though Tor encrypts data at multiple layers, a well-positioned adversary can still infer user activity by analyzing flow characteristics rather than packet content. Traffic correlation attacks exploit timing, volume, and statistical similarities between Tor entry and exit traffic to infer associations between users and their online activity (Murdoch & Danezis, 2005; Li et al., 2021; Palmieri, 2021). Among the state-of-the-art approaches for flow correlation attacks ((Nasr et al., 2018; Oh et al., 2022; Chawla et al., 2024)), *DeepCoFFEA* (Oh et al., 2022) has demonstrated significant improvements in detecting correlated flows. *DeepCoFFEA* employs deep feature embeddings and amplification techniques to enhance attack efficiency while reducing the number of required pairwise comparisons. By mapping Tor entry and exit flows into a lower-dimensional embedding space using a *triplet network*, DeepCoFFEA improves computational efficiency and enhances correlation accuracy. However, a critical challenge remains: **DeepCoFFEA still produces a high number of false positives**. While its amplification mechanism reduces some false positives, the overall precision remains limited when applied to large-scale network traffic. High false positive rates introduce ambiguity in correlation results, reducing the attack's reliability and increasing the number of incorrect flow pairings.

Proposed Approach To address this issue, we introduce a *Graph Neural Network (GNN)-based post-processing framework* that further refines DeepCoFFEA's results. Instead of

[1]Department of Computer Science, University of Waterloo, Canada. Correspondence to: Gurjot Singh <gurjot.singh1@uwaterloo.ca>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

relying purely on all-pairs cosine similarity for determining correlated flows, we construct a graph representation of traffic flows, where nodes represent flow embeddings and edges denote similarity scores. By applying GNN smoothing techniques, our approach refines DeepCoFFEA's correlation decisions by leveraging structural flow relationships, effectively reducing false positives while maintaining high true positive retention. Our experimental results demonstrate that integrating GNN post-processing improves Bayesian Detection Rate (BDR) compared to DeepCoFFEA alone. This suggests that graph-based learning techniques can enhance flow correlation attacks, leading to more precise traffic analysis in anonymity networks. The rest of this paper explores the design, implementation, and evaluation of this approach in detail.

2. Methodology

Our approach consists of two main stages. First, we establish a baseline using the DeepCoFFEA method to correlate network flows. Then, we refine these results using various graph neural networks including: **Graph Attention Network (GAT)** & **Transformer Graph Neural Network** to reduce false positives while maintaining true positives.

Stage 1: DeepCoFFEA Baseline Model DeepCoFFEA leverages a **triplet network** to learn a low-dimensional embedding space where correlated Tor flows are mapped closer together. Given an entry flow $\mathbf{t}$ and an exit flow $\mathbf{x}$, DeepCoFFEA trains two neural networks G and H such that their respective embeddings $G(\mathbf{t})$ and $H(\mathbf{x})$ are close if the flows are correlated and distant otherwise. The model is trained using the **triplet loss function**, which minimizes the cosine distance between correlated pairs while maximizing the distance between non-correlated pairs:

$$\mathcal{L} = \max(0, d(G(\mathbf{t}), H(\mathbf{x}_n)) - d(G(\mathbf{t}), H(\mathbf{x}_p)) + \alpha) \quad (1)$$

where $\mathbf{x}_p$ is the positive (correlated) exit flow, $\mathbf{x}_n$ is the negative (non-correlated) exit flow, $d(\cdot,\cdot)$ is the cosine similarity function, and α is a margin to enforce a separation between positive and negative pairs. DeepCoFFEA then uses an **amplification** technique where the flow is divided into k segments (time windows), each generating an embedding. Correlation is determined by majority voting across all k embeddings:

$$S(\mathbf{t}, \mathbf{x}) = \sum_{i=1}^{k} \nVdash [d(G(\mathbf{t}_i), H(\mathbf{x}_i)) \geq \tau] \quad (2)$$

where τ is the similarity threshold, and the pair $(\mathbf{t}, \mathbf{x})$ is classified as correlated if $S(\mathbf{t}, \mathbf{x})$ exceeds a predefined threshold.

Stage 2: GAT-Based Post-Processing To address the high number of false positives inherent to the DeepCoFFEA baseline, we propose a post-processing stage using a Graph Attention Network (GAT). As shown in Figure 2, we construct a graph where:

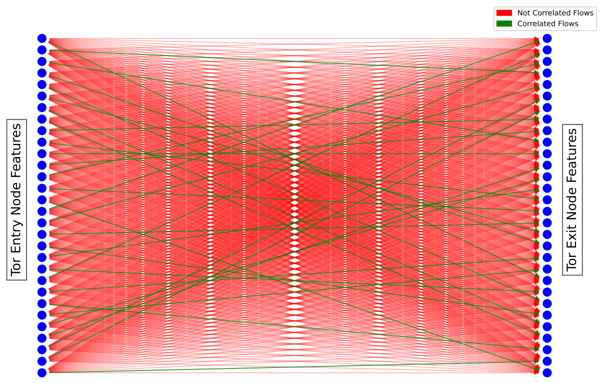

Figure 2. Graph representation of correlated flows: Entry and exit node embeddings extracted from DeepCoFFEA are linked based on predicted correlations.

- Each node represents a flow embedding extracted form the trained triplet network.
- Edges between nodes indicate a predicted correlation.
- Positive edges represent correlated pairs, while negative edges represent non-correlated pairs.

Graph Attention Networks (GATs) (Veličković et al., 2017) are a class of graph neural networks that dynamically assign different importance (or attention) to neighboring nodes when aggregating features. Unlike Graph Convolutional Networks (GCNs) (Kipf & Welling, 2016), which treat all neighboring nodes equally, GATs introduce self-attention mechanisms that allow the model to focus more on important neighbors while suppressing less relevant connections.

Given a graph constructed from DeepCoFFEA correlations, we define the node feature representation for a Tor entry or exit node v as:

$$\mathbf{h}_v^{(0)} = G(\mathbf{t}_v) \quad \text{or} \quad \mathbf{h}_v^{(0)} = H(\mathbf{x}_v), \quad (3)$$

where $\mathbf{h}_v^{(0)}$ represents the initial embedding obtained from the trained triplet network for entry and exit nodes, respectively. Each node updates its representation based on the features of its one-hop neighbors using a learnable attention coefficient. Using these attention scores, the local node embeddings $\mathbf{h}_v^{(0)}$ are converted into global node embeddings $\mathbf{Global}_v^{(0)}$ (aggregated information). This allows the model to suppress weak or spurious correlations by reducing attention to nodes that contribute to false positives. Additionally, it enhances strong correlations by reinforcing connections between nodes that share consistent neighborhood structures and improves robustness to noise in the DeepCoFFEA predictions by leveraging broader graph topology rather than relying on local similarity alone. The final output from GAT is a refined correlation prediction: $S_{\text{GAT}}(v,u) = \sigma\left(\sum_{l=1}^{L} \alpha_{vu}^{(l)} \mathbf{h}_u^{(l)}\right)$ where $S_{\text{GAT}}(v,u)$ represents the final correlation confidence score for the node pair (v,u) & $\sigma(\cdot)$ is an activation function such as ReLU. By applying a refined threshold on S_{GAT}, we can filter out false positives while preserving true positives. This attention-based refinement effectively improves precision in traffic correlation attacks by leveraging network-wide structural dependencies, reducing misclassified links introduced by DeepCoFFEA's initial predictions.

Stage 3: Tansformer GNN-Based Post-Processing The architecture comprises two key components:

1. A **CNN feature extractor** that encodes raw input flow data into high-dimensional feature representations.
2. A **Transformer GNN** that refines the node embeddings via multi-head attention over graph-structured data.

The first stage in our architecture is a Convolutional Neural Network (CNN) (Kiranyaz et al., 2021) that encodes raw input flow data into a compact feature representation. Let the input signal be represented by a vector

$$\mathbf{x} \in \mathbb{R}^n,$$

where n is the input dimension (e.g., $n = 704$). The CNN consists of several 1D convolutional layers. In general, a 1D convolution operation is defined as:

$$y(t) = \sum_{\tau=-k}^{k} x(t+\tau) \cdot w(\tau), \tag{4}$$

where $w(\tau)$ is the convolution kernel of size $2k+1$. In our network, the layers are structured as follows:

- **Layer 1:** Apply a convolution with kernel size $k_1 = 8$, stride 1, and appropriate padding to preserve the input length. This is followed by a batch normalization (BN) and an ELU activation (Ding et al., 2018):
$$\mathbf{x}^{(1)} = \text{ELU}\Big(\text{BN}_1\big(\text{Conv1D}(\mathbf{x};\, \mathbf{W}_1, b_1)\big)\Big).$$
- **Layer 2:** Similarly, a second convolution is applied:
$$\mathbf{x}^{(2)} = \text{ELU}\Big(\text{BN}_2\big(\text{Conv1D}(\mathbf{x}^{(1)};\, \mathbf{W}_2, b_2)\big)\Big).$$
- **Layer 3:** A third convolution is then performed:
$$\mathbf{x}^{(3)} = \text{ELU}\Big(\text{BN}_3\big(\text{Conv1D}(\mathbf{x}^{(2)};\, \mathbf{W}_3, b_3)\big)\Big).$$

After the convolutional layers, the feature maps are flattened into a single vector:

$$\mathbf{x}_{\text{flat}} = \text{flatten}(\mathbf{x}^{(3)}).$$

This flattened representation is then fed into a fully connected (FC) layer to obtain the final feature embedding:

$$\mathbf{h} = \text{Dropout}\Big(\text{ReLU}\big(\mathbf{W}_4\, \mathbf{x}_{\text{flat}} + b_4\big)\Big), \tag{5}$$

where $\mathbf{h} \in \mathbb{R}^d$ is the encoded feature vector and d is chosen to be equal to the input dimension for consistency (e.g., $d = 704$). Once the CNN extracts features for each node in our graph, these features are refined using a Transformer-based Graph Neural Network (Yun et al., 2019; Min et al., 2022). Let the matrix of node features be

$$\mathbf{H} \in \mathbb{R}^{N \times d},$$

where N is the number of nodes and each row $\mathbf{h}_i$ is the feature vector obtained from the CNN.

Multi-Head Attention Mechanism

For each node i with feature $\mathbf{h}_i$, we first project the feature into query, key, and value vectors:

$$\mathbf{q}_i = \mathbf{W}_Q\, \mathbf{h}_i, \tag{6}$$
$$\mathbf{k}_i = \mathbf{W}_K\, \mathbf{h}_i, \tag{7}$$
$$\mathbf{v}_i = \mathbf{W}_V\, \mathbf{h}_i, \tag{8}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d' \times d}$ are learnable weight matrices, and d' is the dimension of the projected vectors.

With M attention heads, these projections are performed separately for each head m:

$$\mathbf{q}_i^{(m)} = \mathbf{W}_Q^{(m)} \mathbf{h}_i, \tag{9}$$
$$\mathbf{k}_i^{(m)} = \mathbf{W}_K^{(m)} \mathbf{h}_i, \tag{10}$$
$$\mathbf{v}_i^{(m)} = \mathbf{W}_V^{(m)} \mathbf{h}_i. \tag{11}$$

Attention over Graph Neighborhood

For node i, we only consider its neighborhood $\mathcal{N}(i)$ in the graph. The attention coefficient between node i and a neighboring node j for head m is computed as:

$$\alpha_{ij}^{(m)} = \frac{\exp\left(\text{LeakyReLU}\left(\frac{\mathbf{q}_i^{(m)} \cdot \mathbf{k}_j^{(m)}}{\sqrt{d'}}\right)\right)}{\sum_{l \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\frac{\mathbf{q}_i^{(m)} \cdot \mathbf{k}_l^{(m)}}{\sqrt{d'}}\right)\right)}, \tag{12}$$

where the dot product $\mathbf{q}_i^{(m)} \cdot \mathbf{k}_j^{(m)}$ measures the similarity between the query of node i and the key of node j. The scaling factor $\sqrt{d'}$ ensures that the softmax operates in a numerically stable regime.

Aggregation of Neighboring Features

Using the attention coefficients, the output for node i in head m is given by:

$$\mathbf{h}_i^{(m)'} = \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(m)} \mathbf{v}_j^{(m)}. \tag{13}$$

The outputs from all heads are concatenated to form the new representation:

$$\mathbf{h}_i' = \Vert_{m=1}^{M} \mathbf{h}_i^{(m)'} \quad \in \mathbb{R}^{M \cdot d'}, \tag{14}$$

where $\Vert$ denotes the concatenation operation.

In subsequent layers (e.g., the second TransformerConv layer), the multiple head outputs can be aggregated (e.g., via averaging or a linear transformation) to yield the final refined node embedding:

$$\mathbf{z}_i \in \mathbb{R}^{d''}.$$

After obtaining the refined node embeddings $\mathbf{z}_i$ from the Transformer GNN, we perform link prediction by evaluating the likelihood that an edge exists between any two nodes i and j. For an edge (i, j), the edge feature is constructed by concatenating the embeddings of the source and target nodes:

$$\mathbf{e}_{ij} = \mathbf{z}_i \Vert \mathbf{z}_j \quad \in \mathbb{R}^{2d''}. \tag{15}$$

The concatenated edge feature is then passed through a two-layer feed-forward neural network:

$$\mathbf{h}_{ij} = \text{ReLU}\Big(\mathbf{W}_1 \mathbf{e}_{ij} + b_1\Big), \tag{16}$$

$$\hat{y}_{ij} = \sigma\Big(\mathbf{W}_2 \mathbf{h}_{ij} + b_2\Big), \tag{17}$$

where:

- $\mathbf{W}_1$ and $\mathbf{W}_2$ are weight matrices,
- b_1 and b_2 are bias terms,
- $\sigma(\cdot)$ denotes the sigmoid activation function,
- $\hat{y}_{ij}$ is the predicted probability that a link exists between nodes i and j.

By applying a refined threshold on $\hat{y}_{ij}$, we can filter out false positives while preserving true positives.

3. Results and Analysis

DeepCoFFEA Training and Evaluation We train the DeepCoFFEA model using a triplet network architecture until the triplet loss reaches 0.0061 after 217 epochs. The model is trained on 10,000 flows and tested on 2,094 flows. Given that each flow can be correlated with any other, the total number of possible flow combinations in the worst case is $N_{\text{pairs}} = N_{\text{flows}} \times N_{\text{flows}}$, resulting in $N_{\text{pairs}} = 2094 \times 2094 = 4,384,836$. By tuning the cosine similarity threshold, we balance true positives (TP) and false positives (FP), achieving an 89% true positive rate (TPR) with a 0.07% false positive rate (FPR). Although this FPR seems low, it translates to approximately 3,015 false positives, significantly affecting correlation accuracy.

Performance of GNN-Based Post-Processing To refine DeepCoFFEA's results, we introduce a Graph Attention Network (GAT) post-processing step. We evaluate our approach based on the true positive rate (TPR), false positive rate (FPR), and the Bayesian Detection Rate (BDR). The true positive rate, also known as recall, measures the proportion of actual correlated flows that were correctly identified. Similarly, the false positive rate measures the proportion of incorrectly correlated flows among all non-correlated flows. While TPR and FPR provide insights into classification accuracy, they do not fully capture the trade-off between the two. The Bayesian Detection Rate (BDR) provides a more comprehensive measure, balancing true positives and false positives relative to the total number of flows in the dataset. BDR is defined as:

$$\text{BDR} = \frac{(\text{TPR} \times c)}{(\text{TPR} \times c) + (\text{FPR} \times u)} \tag{18}$$

where:

$$c = \frac{1}{2094}, \quad u = \frac{2093}{2094} \tag{19}$$

For our GNN-based post-processing model, we obtain a **BDR of 0.39651641270943405**, whereas for DeepCoFFEA, the **BDR is 0.38442995978169797**. indicating a moderate improvement. The (BDR) metric is particularly useful in evaluating correlation models, as it balances high recall (TPR) and low false positive rates (FPR) within a single measure. Our approach achieved a higher BDR while maintaining a TPR of 0.9952, which is comparable to DeepCoFFEA's performance. Specifically, DeepCoFFEA correctly identified 1,882 true positives out of 2,094, whereas our model achieved 1,873 true positives out of 2,094. Additionally, our approach successfully reduced the number of false positives from 3,015 in DeepCoFFEA to 2,852. By leveraging a graph-based refinement technique, our model enhances precision while preserving strong TPR.

Performance of Transformer GNN-Based Post-Processing The new Transformer GNN architecture further refines these results. As shown in Table 1, by varying the threshold from 0.0001 to 0.0020 (see Figures 5 & 6 for more deatils), the model consistently maintains a TPR around 0.86 while lowering the false positive rate (FPR) to as low as approximately **0.000348**. This corresponds to a BDR that steadily increases from **0.39651641270943405** (results from GAT based post processing which was initally greater than the baseline) up to **0.5413974962092674**. These improvements indicate that the Transformer GNN not only continues to preserve high recall but also enhances precision by significantly reducing the number of false positives. This effect is primarily attributed to the multi-head attention mechanism of the Transformer GNN, which better aggregates neighborhood information and suppresses spurious correlations.

As a result, the model achieves a more favorable trade-off between true and false positives, thereby improving the overall Bayesian Detection Rate (BDR). After extensive evaluation (see Table 2, we found that the best-performing hyperparameters were a learning rate of 1×10^{-4}, weight decay of 5×10^{-4}, 4 transformer heads, a hidden dimension of 2048, and a positive class weight of 4. This configuration achieved an optimal balance between reducing false positives and maintaining a high true positive rate. In particular, the choice of a learning rate of 1×10^{-4} and weight decay of 5×10^{-4} provided a stable training dynamic, while **4** transformer heads and a hidden dimension of **2048** allowed the model to capture rich representations of the graph-structured data. Additionally, a positive class weight of **4** effectively countered class imbalance, further enhancing the precision of our predictions.

Comparison of GAT vs. Transformer-Based GNN Training As shown in Figure 4 and Figure 3, the Transformer-based GNN achieves smoother and lower loss convergence compared to the GAT-based model. While the GAT model's training loss drops quickly, its validation loss fluctuates between 0.25 and 0.35, indicating instability or overfitting. In contrast, the Transformer GNN demonstrates a gradual decline in both training and validation loss, converging near or below 0.2 with a smaller gap, which suggests better generalization. This improvement is largely due to the multi-head attention mechanism that captures richer node relationships and results in a more stable training process.

Effect of Threshold on TPR, BDR, and FPR Figures 5 and 6 illustrate the variation of True Positive Rate (TPR), Bayesian Detection Rate (BDR), and Scaled False Positive Rate (FPR) as the decision threshold increases, compared to the baseline performance. As the threshold increases from very low values (e.g., 0.0001) to moderate values (e.g., 0.004), the FPR gradually decreases while the BDR improves. This indicates that raising the threshold effectively reduces false positives and enhances the overall detection metric (BDR), with only a minimal reduction in TPR—which is desirable since it implies that most true positives are preserved.

However, when the threshold is increased substantially (As shown in Table 3), the TPR drops from 0.8505 to 0.8352. While this further decreases the FPR and increases the BDR, it also leads to a considerable loss of true positive detections. This trade-off demonstrates that although a higher threshold can improve precision by filtering out spurious detections, it does so at the expense of sensitivity. Therefore, a moderate threshold is preferred to maintain a balance between high detection rates and low false positives.

Table 1. Performance Metrics for Various Thresholds

Threshold	**TPR**	**FPR**	**BDR**
0.0001	**0.8672**	0.000489	0.4586
0.0002	0.8663	0.000466	0.4704
0.0003	0.8663	0.000452	0.4779
0.0004	0.8658	0.000440	0.4844
0.0005	0.8649	0.000433	0.4881
0.0006	0.8634	0.000427	0.4914
0.0007	0.8634	0.000421	0.4952
0.0008	0.8634	0.000413	0.4996
0.0009	0.8634	0.000410	0.5016
0.0010	0.8634	0.000405	0.5044
0.0011	0.8634	0.000400	0.5074
0.0012	0.8629	0.000398	0.5091
0.0013	0.8629	0.000394	0.5111
0.0014	0.8629	0.000392	0.5123
0.0015	0.8629	0.000391	0.5135
0.0016	0.8629	0.000388	0.5154
0.0017	0.8629	0.000384	0.5177
0.0018	0.8629	0.000382	0.5191
0.0019	0.8620	0.000380	0.5203
0.0020	0.8620	0.000378	0.5216
0.0021	0.8620	0.000376	0.5230
0.0022	0.8620	0.000374	0.5244
0.0023	0.8620	**0.000372**	**0.5251**

Table 2. Grid Search Hyperparameters

Parameter	**Values Explored**
Learning Rate	1×10^{-4}, 5×10^{-5}, 1×10^{-5}
Weight Decay	5×10^{-4}, 1×10^{-3}, 5×10^{-3}
Transformer Heads	2, 4, 8
Hidden Dimension	1024, 2048, 4096
Positive Class Weight	2, 4, 8

4. Conclusion

In this project, we presented an integrated framework that combines the strengths of DeepCoFFEA with graph-based post-processing techniques to address the high false positive rates inherent in conventional flow correlation attacks in

Table 3. Performance Metrics for Higher Thresholds

Threshold	TPR	FPR	BDR
0.1000	0.8505	0.000183	0.6899
0.2000	0.8477	0.000137	0.7475
0.3000	0.8453	0.000110	0.7864
0.4000	0.8448	0.000086	0.8240
0.5000	0.8429	0.000071	0.8503
0.6000	0.8419	0.000058	0.8746
0.7000	0.8352	0.000045	0.8988

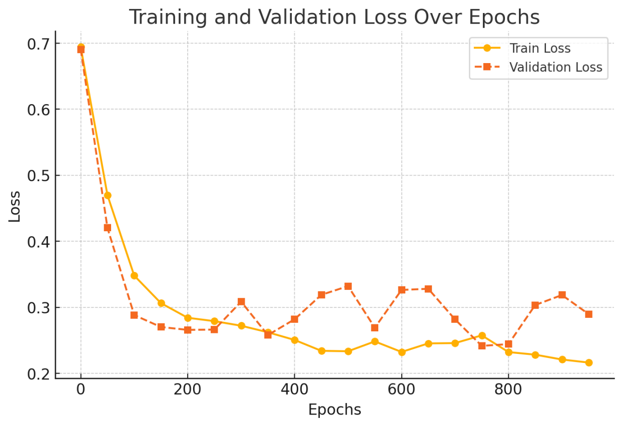

Figure 3. Training and validation loss decrease over epochs but remain stable at higher values, indicating the model did not fit the data completely.

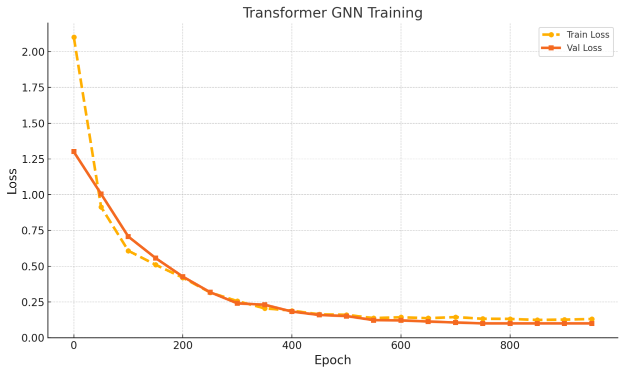

Figure 4. Training and validation loss of the Transformer GNN decrease over epochs indicating the model did fit the data completely.

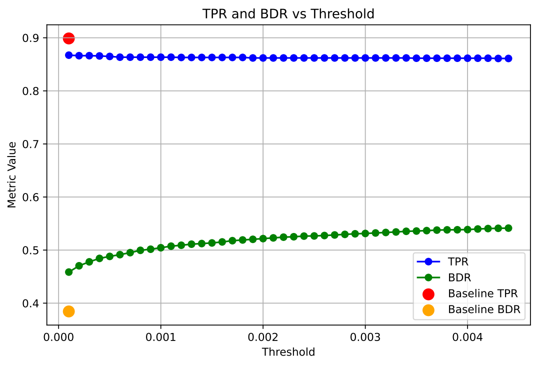

Figure 5. TPR and BDR rates of the Transformer Based GNN model as compared to the Baseline (DeepCoffea)

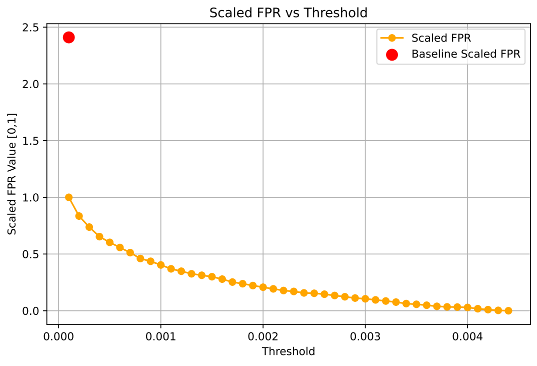

Figure 6. FPR rates of the Transformer Based GNN model as compared to the Baseline (DeepCoffea)

Tor. While DeepCoFFEA has demonstrated state-of-the-art performance in correlating network flows, its high false positive rate limits its effectiveness. To mitigate this issue, we explored two post-processing approaches: one based on Graph Attention Networks (GAT) and another using a Transformer-based GNN. Our experimental results indicate that the GAT-based approach improved the Bayesian Detection Rate (BDR) from a baseline of approximately 0.38 to 0.39, albeit with a slight reduction in true positive rate (TPR). More notably, the Transformer-based GNN further enhanced the BDR to as high as 0.54 & 0.89 (in cases with a drop in TP rate), demonstrating a substantial reduction in false positives while still maintaining a strong detection capability. Overall, these findings prove that incorporating advanced graph-based techniques, particularly the Transformer-based GNN, is an effective strategy for enhancing flow correlation attacks in Tor. This integrated solution not only addresses the shortcomings of DeepCoFFEA by reducing false positives but also paves the way for more precise and reliable detection mechanisms in the realm of traffic analysis and network security.

References

Chawla, T., Mittal, S., Mathews, N., and Wright, M. Espresso: Advanced end-to-end flow correlation attacks on tor. In *Proceedings of the 8th Asia-Pacific Workshop on Networking*, pp. 219–220, 2024.

Ding, B., Qian, H., and Zhou, J. Activation functions and their characteristics in deep neural networks. In *2018 Chinese control and decision conference (CCDC)*, pp. 1836–1841. IEEE, 2018.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., and Inman, D. J. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

Li, J., Gu, C., Zhang, X., Chen, X., and Liu, W. Attcorr: A novel deep learning model for flow correlation attacks on tor. In *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pp. 427–430. IEEE, 2021.

Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananiadou, S., and Rong, Y. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.

Murdoch, S. J. and Danezis, G. Low-cost traffic analysis of tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pp. 183–195. IEEE, 2005.

Nasr, M., Bahramali, A., and Houmansadr, A. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pp. 1962–1976, 2018.

Oh, S. E., Yang, T., Mathews, N., Holland, J. K., Rahman, M. S., Hopper, N., and Wright, M. Deepcoffea: Improved flow correlation attacks on tor via metric learning and amplification. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1915–1932. IEEE, 2022.

Palmieri, F. A distributed flow correlation attack to anonymizing overlay networks based on wavelet multi-resolution analysis. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2271–2284, 2021. doi: 10.1109/TDSC.2019.2947666.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.