On Evaluation of Data Fusion Methods for Network Traffic Classification

1st Richard Plny
FIT CTU & CESNET, Prague, Czechia
richard.plny@fit.cvut.cz

2nd Karel Hynek
FIT CTU & CESNET, Prague, Czechia
karel.hynek@cesnet.cz

***Abstract*—Data fusion plays a critical role in modern network security, enhancing explainability and trust in ML/AI-based detectors. Yet, the effect of standard fusion methods has been largely overlooked. This paper evaluates seven common late- and hard-fusion techniques—Majority Voting, Weighted Majority Voting, Recall Combiner, Naive Bayes, Behavior-Knowledge Space, Decision Tree, and Logistic Regression—in the context of traffic classification and attack detection. Using both synthetic data from controlled environments and real-world datasets, we analyze their performance across diverse scenarios. The results highlight the strengths and limitations of each method and offer practical guidance for selecting effective fusion strategies in network security applications.**

***Index Terms*—network traffic classification, data fusion, network security**

I. Introduction

Heterogeneous (or multimodal) classifiers are increasingly gaining traction in computer security research due to their ability to enhance robustness and improve overall detection accuracy [7]. Similar to classical ensemble methods, leveraging diverse data sources boosts accuracy while mitigating the risk of overfitting. Additionally, heterogeneous classifiers are explainable by design [15], which is essential for critical applications where non-AI experts need to interpret and trust classification results [3]. Typical use cases for such methods include network attack and intrusion detection, where security analysts respond to alerts produced by AI-driven systems.

Heterogeneous classifiers operate on the principle of individual classification pipelines, whose results are subsequently combined through data fusion to yield the final classification outcome. Their operational principle is depicted in the Fig. 1. However, prior research in computer security has largely overlooked the data fusion aspect. Instead, methods like majority voting, logistic regression, or other custom techniques have been employed [7], [15], often without thoroughly investigating the characteristics and implications of the chosen fusion approach. Moreover, deep neural networks (DNNs) are used to perform fusion, as in [2]. We argue that even though complex DNNs and potentially LLMs can be used for data fusion as well, we should not skip the consideration of the common data fusion methods. Especially in the face of recent research results [9], where DNN performance on networking datasets is lower than traditional ML methods such as k-NN or Decision Trees. The common data fusion methods have to be investigated as they can yield similar results more efficiently, potentially establishing a baseline for future research of DNN- and LLM-based methods.

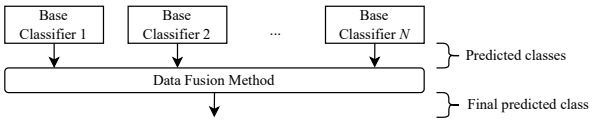

Fig. 1. Heterogeneous classification method

This research aims to evaluate the performance and properties of various data fusion methods in network security use cases. Motivated by the statements above, we selected seven common fusion methods from related work and assessed them across three distinct test cases in a controlled environment on synthetic data, ensuring reproducibility and maximal transparency of the evaluation protocol. Moreover, observations were then evaluated on two real-world datasets widely recognized by the network security community. The main contribution of this paper is the implementation[1] and comprehensive analysis of diverse data fusion techniques, providing a deeper understanding of their effectiveness in the context of network security.

II. Related Work

In this section, we summarize related work. We can divide the related work into two parts: 1) Data fusion studies, exploring algorithms designed to integrate diverse data sources effectively, and 2) Heterogeneous classification approaches that utilize data fusion to improve the accuracy, robustness, and explainability of network traffic classifiers.

A. Data fusion studies

Data fusion combines multiple records of the same entity into a unified representation and can be performed at early, intermediate, or late stages [8], [14]. Early fusion integrates raw modalities to capture intra- and inter-modality relations, intermediate fusion learns them separately before combining, while late fusion relies on modality-specific classifiers, often yielding higher per-modality accuracy. Although it ignores cross-modality links, late fusion improves explainability by exposing classifier outputs [12], and is the focus of this study.

[1] Available at https://github.com/plnyrich/DataFusionExperiments

978-3-903176-75-1 ©2025 IFIP

Fusion approaches are further divided into hard, which combine predicted labels, and soft, which combine class probabilities. Kuncheva et al. [5] showed that the widely used majority vote (MAJ) can vary sharply in accuracy based on label distributions. Their later work [6] compared MAJ with weighted majority vote (WMJ), recall (REC), and Naive Bayes (NB). While NB was advised for imbalanced cases, performance differences with WMJ were not statistically significant, with WMJ sometimes outperforming NB. Given network security's extreme imbalance ratios (often 99:1), broader evaluation of these methods in this domain remains necessary.

B. Heterogeneous classification approaches

Li et al. [7] presented a thorough review of heterogeneous classification approaches that leverage data fusion techniques for network traffic classification. However, their work primarily focuses on a single data fusion method without delving into its optimality or comparing it against other approaches.

Aceto et al. [1] remains the only study comparing multiple data fusion methods for network traffic classification. Evaluating six hard and four soft techniques on encrypted traffic analysis, they reported a +9.5% recall gain, with BKS excelling in accuracy and Naive Bayes in recall. However, their work was limited to a single dataset on mobile app identification.

Compared to related work, this paper offers a more in-depth analysis of the properties of data fusion methods in the context of network traffic classification. We evaluate these methods using synthetic datasets with well-defined properties, allowing for a controlled performance examination. Furthermore, we validate and deepen the insights gained from the synthetic data by applying the same methods to two real-world network traffic classification use cases.

III. Evaluation of Data Fusion Methods

This study focuses on hard and late fusion, where the inputs are predictions from individual classifiers and the output is the final class. To ensure controlled conditions, we evaluate fusion methods on synthetic datasets with defined properties, avoiding inconsistencies seen in real network datasets. Experiments are repeated with different random seeds to reduce bias. Unlike real data, synthetic datasets include rare classifier-output combinations, which make fusion more challenging, prevent trivial near-perfect accuracy caused by correlations, and provide a more meaningful benchmarking scale.

Three scenarios were designed: 1) binary classification 1 (BC1), 2) binary classification 2 (BC2), and 3) n-classification (NC3). All scenarios explore the data fusion of hard labels provided by base classifiers. We experimented with different numbers of classifiers and their respective performances, as well as several class imbalances. The performance of classifiers is controlled by their recalls, explained in subsection III-A.

BC1 simulates N base classifiers where one underperforms, starting at 50% recall and improving by +1% per step until 100%, while the others remain fixed at 95%. This tests fusion robustness to a weak input. **BC2** models N classifiers of equal performance, with recalls increasing from 50% to 100% in +1% steps, assessing how classifier strength and number affect fusion. **NC3** extends BC2 to multiclass classification with N classes and N classifiers of identical performance, again varying recall from 50% to 100%. In BC1 and BC2, the task is binary classification (e.g., detecting a rare attack), while NC3 examines multiclass separation of network traffic.

The task in BC1 and BC2 scenarios is to reveal the positive class in binary classification. Since we are targeting network security, an example can be traffic type or an attack that rarely occurs in the network but is significant from the security point of view. The multiclass classification task is presented in NC3, simulating traffic separation into different classes or groups.

A. Data generation

Since the recall of the i-th classifier for class c is calculated as a percentage of positively identified elements from a set of all elements of the class c, it can be used as an estimation of the probability of the i-th classifier to assign class c ($l_i = c$) to an element with ground truth label c, see Equation 1. Therefore, we control classifiers' performance by setting their per-class recalls to the desired probability.

$$\text{REC}_{i,c} = \frac{TP_{i,c}}{TP_{i,c} + FN_{i,c}} \approx P(l_i = c | \text{ground truth} = c) \quad (1)$$

The process begins by seeding a random generator and creating ground truth labels according to the class imbalance parameter. For each label c, classifier i outputs c with probability $REC_{i,c}$ and a different class with probability $1 - REC_{i,c}$. In binary tasks (BC1, BC2), the wrong class is fixed, while in multiclass (NC3) it is chosen uniformly at random. Generator also supports class similarity output to produce more correlated results; but such feature is not used in our study.

B. Selected data fusion methods

Data fusion methods evaluated in this paper were carefully selected based on literature review [1], [6], [13]. Majority[2] (MAJ), Weighted Majority (WMJ), Recall (REC), Naive Bayes (NB), Behavior-Knowledge Space (BKS), Decision Tree (DT), and Logistic Regression (LG). Furthermore, we employed the Oracle (ORA) method [13], which assigns the correct label if at least one of the input labels is correct. Otherwise it assigns a wrong label. The ORA method was chosen because it is widely regarded as an ideal approach for data fusion, often serving as a benchmark for theoretical upper-bound performance. However, this method is only theoretical and cannot be used in practice as it leverages ground truth information. On the other hand, as demonstrated in our observations in Section IV, it does not always provide the ideal fusion output, as other methods can outperform ORA under certain conditions. In total, we implemented eight data fusion methods for our experimental evaluation.

[2]This method is more accurately called Prevailing Vote, as it does not require a 50% threshold.

C. Experimental protocol

Firstly, we defined possible parameter values together with ten different random seeds. The common parameters for both BC1 and BC2 scenarios are:

- Synthetic dataset size is 10 000
- Number of classifiers are 3, 5, and 7
- Class imbalances are 50:50, 90:10, and 99:1

Parameters of NC3 differ in some cases. The considered number of classifiers is 3, 5, and 7; the number of classes is equal to the number of classifiers in each configuration. Class imbalance is inferred from the number of classes and is set to be balanced.

We evaluate the data fusion methods using a standard supervised classification evaluation protocol. For each parameter combination, we create a series of datasets where the recall of the base classifiers gradually increases, depending on the specific scenario. Each dataset includes generated outputs (labels) of the base classifiers (representing the features used by the fusion method), along with the corresponding ground truth labels. As the recall of base classifiers gradually increases in the series, the ground truth labels stay the same as the random seed in the series is fixed. The input features for the fusion method are specifically designed to align with the selected recall values of the base classifiers. This setup allows us to systematically assess the performance of the fusion method under controlled variations of base classifiers' accuracy.

Each dataset is randomly split into training and testing subsets in a 7:3 ratio. The fusion methods are trained on the training subset and evaluated on the testing subset. This procedure is applied to every dataset in the series, allowing us to observe the fusion method's performance as a function of the input classifiers' recall values.

Accuracy, recall, precision, and F1-score metrics are tracked for each method. Recall, precision, and F1-score are used in a binary variant in BC1 and BC2, reporting the metric only for the positive class. Macro-averaged variants are then used in the multiclass experiments in NC3.

Furthermore, we present an F1-Recall AuC metric, which will also be tracked during our experiments. A curve is composed of points where input recall of base classifier(s) denotes the position on the x-axis, and the F1-score of the resulting method defines the position on the y-axis. An area under this curve (integral) is a value of our F1-Recall AuC metric. For example, imagine *method 1* with F1-Recall AuC of `0.4974` and the F1-Recall AuC of `0.4598` for the *method 2*. As the curves show the resulting F1-score for given input recall, the *method 1* can overall perform better than the *method 2*. Denote that the maximum value of the F1-Recall AuC metric is `0.5` in this case, as the input recall is from `0.5`—`1.0` with the maximum possible value of F1-score `1.0`. However, this can differ depending on the axis scale used.

Finally, performance metrics for each parameter combination are averaged across the ten runs with different random seeds to suppress the noise introduced by random data generation.

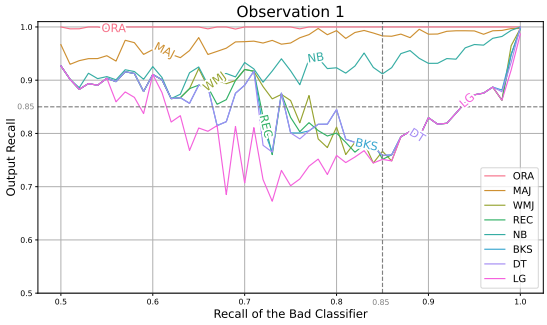

Fig. 2. Observation 1: Fusion achieves worse output recall than the input classifiers in BC1 with three classifiers

IV. Results

The BC1 and BC2 experiments with class imbalances of 50:50 and 90:10 did yield results in line with Kuncheva et al. [6]. Therefore, we focused on 99:1 class imbalance in BC1 and BC2 since these are the most relevant to network security and were not evaluated in previous work. Firstly, we point out a few observations, then we evaluate methods behavior through our F1-Recall AuC metric. Please note that we do not have enough space for graphs with F1-Recall curves for all versions of our experiments. Therefore, we omit them and use the F1-Recall AuC metric for the final evaluation. Finally, note that the graphs for BC1 use the recall of the "badly"-performing base classifiers on the x-axis, whereas the other graphs use the average recall of the base classifiers. Furthermore, BC1 and BC2 uses binary variants of the metrics, whereas NC3 uses macro-averaged variants.

Observation 1: Fusion methods WMJ, REC, BKS, DT, and LG yield recall performance lower than each individual input classifier in BC1 with three classifiers. This phenomenon can be seen in the Fig. 2, for example when the input recall is larger than 85%.

Observation 2: The so-called ideal ORA method does not consistently perform as the best method in all scenarios. This is evident in the Fig. 3 depicting methods behavior in BC2 with three classifiers. Specifically, the ORA method (e.g., described by Sedlacek et al. [13] as ideal) is surpassed by WMJ, REC, NB, BKS, and DT. These methods are capable of correctly identifying the correct class even when none of the input predictions is correct.

Observation 3: Fusion performance improves as the number of classifiers and—interestingly—also classes increase. In NC3, MAJ, WMJ, and REC achieve $\sim$56% F1-score with three classes and classifiers when input recalls are set to 50%, see the Fig. 4. The triplet achieves F1-score $\sim$71% with the same input recalls in a scenario with five classes and classifiers. Finally, the triplet achieves F1-score $\sim$82% is achieved with seven classes and classifiers. On the other hand, performance of BKS and LG methods decrease with increasing number of classes and classifiers.

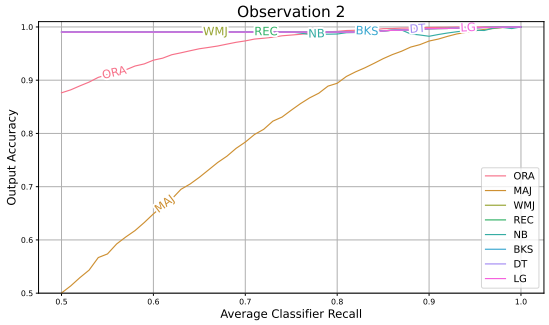

Fig. 3. Observation 2: Oracle is surpassed by other methods in BC2 with 3 classifiers

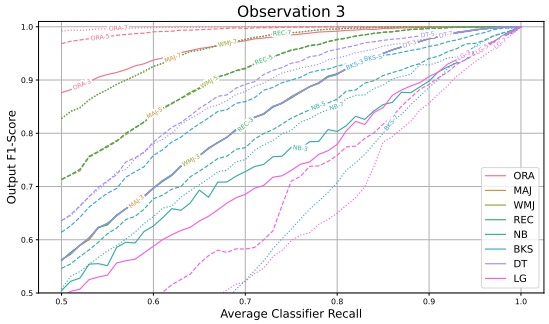

Fig. 4. Observation 3: Behavior of methods in NC3, curves are labeled as Method X-N denoting the performance of the method X in the NC3 scenario with N classes and N classifiers

A. Final Evaluation

Finally, we integrated the F1-Recall curves to obtain the F1-Recall AuC metric. We also present the difference from the Oracle method, denoted as Δ. However, as the ORA method is only a theoretical upper bound approximation, we do not consider it when evaluating the best data fusion method.

The Table I shows the F1-Recall AuC in the BC1. WMJ achieved the best results, closely followed by REC. However, WMJ, REC, DT, and LG achieved high accuracy and consistent results with the increasing number of classifiers. In the most challenging task with three classifiers, NB performed significantly worse, and MAJ achieved only 0.25 F1-Recall AuC compared to the best method with 0.42 (ORA excluded). As the number of input labels (and their combinations) increased with the number of used classifiers, BKS could not keep up with the increasing performance of other methods and fell to the last place. This is not surprising as the number of combinations exponentially rises, BKS does not have enough data in the training set, and its fusion capabilities decrease. On the other hand, as more labels are used as the fusion input, performance of MAJ and NB rises.

The Table II shows results in BC2. Here, NB outperforms all other methods. However, WMJ, REC, DT, and LG performed only slightly worse. All methods showed similar behavior except MAJ, which performed significantly worse. Even with an increasing number of classifiers, MAJ performed much worse than other methods, where the performance is comparable. As none of the classifiers yields consistently good predictions, a majority is harder to achieve. Meanwhile, more complex methods can work with such behavior. As MAJ was outperformed in BC1 as well, we do not consider it as an ideal method for highly imbalanced TC problems and intrusion detection.

TABLE I
BC1 - METHODS PERFORMANCE IN F1-RECALL AUC METRIC

	ORA	WMJ	REC	BKS	DT	LG	NB	MAJ
Classifier Count = 3								
AuC	.48307	.42258	.42119	.42027	.42015	.41512	.35950	.23204
Δ	—	.06049	.06188	.06280	.06292	.06795	.12357	.25103
Classifier Count = 5								
	ORA	WMJ	REC	LG	BKS	DT	NB	MAJ
AuC	.50000	.48721	.48542	.48096	.48064	.47986	.44770	.41476
Δ	—	.01279	.01458	.01904	.01936	.02014	.05230	.08524
Classifier Count = 7								
	ORA	WMJ	REC	LG	NB	MAJ	DT	BKS
AuC	.50000	.49770	.49723	.49429	.48507	.48432	.48377	.46800
Δ	—	.00230	.00277	.00571	.01493	.01568	.01623	.03200

TABLE II
BC2 - METHODS PERFORMANCE IN F1-RECALL AUC METRIC

	ORA	NB	WMJ	REC	DT	BKS	LG	MAJ
Classifier Count = 3								
AuC	.28561	.14419	.14182	.14182	.14181	.14178	.13246	.11388
Δ	—	.14142	.14379	.14379	.14381	.14383	.15315	.17174
Classifier Count = 5								
	ORA	NB	WMJ	BKS	DT	REC	LG	MAJ
AuC	.42317	.20420	.20383	.20321	.20255	.20253	.19594	.15678
Δ	—	.21897	.21934	.21997	.22062	.22064	.22723	.26639
Classifier Count = 7								
	ORA	NB	REC	WMJ	LG	DT	BKS	MAJ
AuC	.47812	.24592	.24388	.24347	.23562	.23443	.23318	.18967
Δ	—	.23220	.23424	.23466	.24250	.24370	.24494	.28845

The Table III shows results in NC3. MAJ, WMJ, and REC achieved the highest performance across the increasing number of classifiers and classes. BKS shows similar behavior as before. When the number of input label combinations is small (Classifier Count = 3), BKS is one of the best methods. However, with the exponential increase in label combinations, its performance dramatically worsens. DT also produces good performance with three classifiers and labels, but cannot keep up with MAJ, WMJ, and REC when the number of classes and classifiers increases. Interestingly, NB achieves similar F1-Recall AuC over all the cases, suggesting that NB cannot utilize information from classifiers in the evaluated tasks. Finally, LG performs poorly throughout the experiment and its performance only worsens as the number of classes and classifiers increases.

V. CASE STUDIES

We evaluate data fusion methods on two real-world tasks: TOR detection and QUIC service classification. For both, we designed classifiers to obtain hard labels but do not aim for SOTA performance, as tuning classifiers lies beyond this study's scope. ORA is included only as a hypothetical upper bound, not as a practical baseline. Datasets were split 70/30 into train and test sets, with both base classifiers and fusion methods trained on the former. Each experiment was repeated

ten times with different random states, and results are reported as averages. Standard deviations were negligible (max 0.3) and thus omitted.

A. TOR detection

In this use case, we use the ISCX-Tor-2016 dataset [4], which contains TOR and non-TOR traffic. We processed the pcaps with ipfixprobe[3]. It is a highly imbalanced dataset with 99.33% of non-Tor flows and only 0.67% flows representing the TOR class. The heterogeneous method for this task is composed of:

1) **Basic Traffic Shape** The first classifier is based on machine learning and uses the LGBM model. This model processes the standard flow features: number of packets and bytes transmitted in both directions; macro recall is 93.93%.
2) **SNI** This classifier is based on the Server Name Indication (SNI) extension of the TLS protocol. This classifier uses the k-nearest neighbors model ($k = 5$) frequencies of all alphanumerical characters as a feature vector; macro recall is 52.07%.
3) **IP** The last classifier is based on blocklists. IP blocklist is created from the train part of the dataset and used for the classification of the test part; macro recall is 100%.

Results

Since two classifiers achieved 100% and the last one 52%, we can compare the methods' performance with our BC1 experiment with three classifiers, 99:1 class imbalance, and the input recall 52.5% (x-axis). The results of methods are available in the Table IV. DT and LG achieved the best performance, corresponding with our findings in the BC1.

As expected, we found the real-world data to be less noisy and more correlated than randomly generated data. Therefore, all methods performed better than expected. NB was even able to achieve the same performance as the DT and LG.

B. QUIC service classification

CESNET-QUIC22 [10] contains HTTP/3 (QUIC) traffic from the backbone lines of the ISP network with traffic of many services. We selected the top 40 most frequent services from the dataset. Then, we balanced the dataset to contain the same number of flows for each label, because the CESNET-QUIC22 dataset is extremely imbalanced towards the Google services, which could skew results in data fusion [11]. The heterogeneous method for this task consists of Basic Traffic Shape (macro recall is 58.14%) and IP (macro recall is 63.95%) classifiers (same as in the subsection V-A) and: i)

1) **Detailed Traffic Shape** This classifier is based on LGBM; however, its feature vector is composed of information about the first 30 packets, especially inter-arrival times in milliseconds, directions, and lengths. Flows with less than 30 packets have their feature vector padded with zeros; macro recall is 83.35%.
2) **Similarity** Histograms with eight bins are created from packet sizes for each direction. Then, a k-nearest neighbors model ($k = 3$) uses histogram statistics to look for flows with similar patterns; macro recall is 58.64%.
3) **ASN** This classifier is similar to the IP blocklist, but it uses Autonomous System Numbers rather than IP addresses; macro recall is 13.86%. Macro recall of this classifier is very low; however, per-class recalls for five classes achieve 100%. We argue that this is a perfectly reasonable indicator since it provides a strong result for a subset of classes.

Results

The Table V presents the results when all five previously described base classifiers are employed. Among these, the DT consistently achieved the best performance, confirming its superiority in data fusion problems. REC and WMJ also achieved high performance, confirming our findings from NC3. Even though MAJ achieved worse results by approx. 7% than the WMJ and REC, it still delivered an F1-score of 79.86%. We hypothesized that the ASN classifier's inaccuracy contributes to degradation by introducing noise into the fusion process. To validate this, we conducted additional tests excluding the ASN classifier, with the results also shown in the Table V. Notably, the MAJ method experienced a substantial performance boost across all metrics, supporting our hypothesis. Similarly, WMJ, REC, and NB classifiers showed performance improvements, but the improvements were less noticeable. These methods weigh each input's contribution by accuracy, so while the ASN classifier reduces overall accuracy, its influence on the final label is limited.

In contrast, the BKS, DT, and LG classifiers demonstrated a performance decrease when the ASN classifier was excluded. These methods are capable of learning patterns from individual predictions, and despite the ASN classifier's lower macro recall, its contributions can significantly enhance fusion results.

[3]https://github.com/CESNET/ipfixprobe

TABLE III
NQ3 - METHODS PERFORMANCE IN F1-RECALL AUC METRIC

Classifier Count = 3								
	ORA	**MAJ**	**WMJ**	**DT**	**BKS**	**REC**	**NB**	**LG**
AuC	.48457	.41967	.41959	.41957	.41955	.41952	.38194	.37137
Δ	—	.06490	.06498	.06500	.06502	.06504	.10263	.11320
Classifier Count = 5								
	ORA	**REC**	**MAJ**	**WMJ**	**DT**	**BKS**	**NB**	**LG**
AuC	.49742	.45984	.45981	.45973	.44480	.43271	.40356	.34438
Δ	—	.03757	.03760	.03769	.05262	.06471	.09386	.15304
Classifier Count = 7								
	ORA	**REC**	**MAJ**	**WMJ**	**DT**	**NB**	**LG**	**BKS**
AuC	.49949	.48213	.48209	.48207	.44081	.39388	.31385	.30811
Δ	—	.01736	.01740	.01742	.05868	.10561	.18564	.19138

TABLE IV
RESULTS IN THE TOR DETECTION CASE STUDY

Metric	**MAJ**	**WMJ**	**REC**	**NB**	**BKS**	**DT**	**LG**	**ORA**
Accuracy	99.93	99.92	99.93	100.0	99.99	100.0	100.0	100.0
F1-score	94.14	93.55	94.14	100.0	99.81	100.0	100.0	100.0
Recall	88.93	87.89	88.93	100.0	99.62	100.0	100.0	100.0
Precision	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

TABLE V
RESULTS IN THE QUIC SERVICE CLASSIFICATION CASE STUDY

5 Classifiers								
Metric	**MAJ**	**WMJ**	**REC**	**NB**	**BKS**	**DT**	**LG**	**ORA**
Accuracy	79.81	85.69	86.40	79.64	79.07	88.78	55.88	95.54
F1-score	79.86	85.73	86.65	79.81	78.60	88.76	54.01	95.52
Recall	79.81	85.69	86.40	79.64	79.07	88.78	55.88	95.54
Precision	83.01	86.41	87.54	81.26	78.62	89.02	56.94	95.81
4 Classifiers (without ASN)								
Metric	**MAJ**	**WMJ**	**REC**	**NB**	**BKS**	**DT**	**LG**	**ORA**
Accuracy	+2.15	+0.14	+0.23	+0.52	−0.56	−0.15	−32.36	−0.14
F1-score	+1.99	+0.13	+0.20	+1.75	−0.52	−0.16	−34.41	−0.14
Recall	+2.15	+0.14	+0.23	+0.52	−0.56	−0.15	−32.36	−0.14
Precision	+0.63	+0.13	+0.05	+3.06	−0.45	−0.16	−33.59	−0.14

This distinction highlights the varying importance of input recall across methods. The BKS, DT, and LG methods rely less on the input recall compared to the others. Feature importance analysis from the DT further supports this observation: the ASN classifier was found to be an order of magnitude more influential than both the Basic Traffic Shape and the Similarity classifiers, despite its significantly lower accuracy.

Behavior of the LG method

As the LG method experienced more than a 30% drop in all metrics when the ASN base classifier was not used, we examined individual predictions. We found out that 19 combinations of input base labels (11 175 elements in total; 13.11% of the test set) were classified incorrectly, even though every base classifier predicted the correct class. Such behavior is surprising, as a maximum probability would be assigned when all base classifiers predict the same (and correct) class.

For example, input base labels vector `(21,21,21,21)` was classified as class `32` even though such combination was not present in the training set. When the label from the ASN base classifier was added to the vector (in this case `28`), the class predicted by LG for the enhanced vector `(21,21,21,21,28)` was the correct class `21`. Even though all base classifiers correctly classified every one of the mentioned 11 175 elements, LG was not able to perform fusion, resulting in the wrong output. Moreover, it can be considered a failure to perform data fusion as it produces a wrong output on the ideal fusion input. Such behavior makes the LG method potentially unsuitable for data fusion. The same behavior also showed the NB method, however, in only one combination (26 elements total), making it statistically insignificant.

On the other hand, we found the described behavior interesting, since the fusion output was corrected by enriching the input vector with the incorrect label. This indicates that some data fusion methods can gain performance boosts by utilizing additional base classifier(s), even when they provide lower individual accuracy and exploit their weaknesses. However, further exploration of such behavior is needed, and it may potentially become a part of our future work.

VI. CONCLUSION

As ML/AI models become widespread, explainability is increasingly critical. Heterogeneous classification, which improves both accuracy and robustness, addresses this need, with final data fusion as its key step. In this study, we implement and evaluate seven common fusion methods using a controlled synthetic data generator[4] and validate them on two real-world traffic classification tasks (TOR detection and QUIC service classification). To support fair comparison, we also introduce a custom AuC metric. Our results show that most methods perform similarly, though Decision Trees consistently achieve high accuracy across binary and multiclass settings. Weighted Majority Voting and Recall perform well on multiclass tasks, while Logistic Regression performs poorly, even under ideal conditions, cautioning against its common default use in tools like scikit-learn. Notably, some methods even surpassed the Oracle baseline, demonstrating that ORA should no longer be considered an absolute upper bound. Overall, Decision Trees emerge as a strong and reliable default for data fusion in heterogeneous classifiers.

ACKNOWLEDGMENT

This work was supported by GN5-2 project, Grant no. 101194278 funded by EU, and by the Grant Agency of the Czech Technical University in Prague grant No. SGS23/207/OHK3/3T/18.

REFERENCES

[1] Aceto, G. et al.: Multi-classification approaches for classifying mobile app traffic. Network and Computer Applications (2018)
[2] Ahmad, R. et al.: Data fusion and network intrusion detection systems. Cluster Computing **27**(6), 7493–7519 (2024)
[3] Alahmadi B. et al.: 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. In: 31st USENIX Security. USENIX Association (2022)
[4] Habibi Lashkari A. et al.: Characterization of tor traffic using time based features. In: Proceedings of the 3rd ICISSP. INSTICC, SciTePress (2017). https://doi.org/10.5220/0006105602530262
[5] Kuncheva L. et al.: Limits on the majority vote accuracy in classifier fusion. Pattern Analysis & Applications (2003)
[6] Kuncheva L. et al.: A weighted voting framework for classifiers ensembles. Knowledge and information systems (2014)
[7] Li G. et al.: Data fusion for network intrusion detection: A review. Security and Communication Networks (2018). https://doi.org/10.1155/2018/8210614
[8] Li G. et al.: Data fusion for network intrusion detection: a review. Security and Communication Networks (2018)
[9] Luxemburk, J., Hynek, K., Plný, R., Čejka, T.: Universal embedding function for traffic classification via quic domain recognition pretraining: A transfer learning success (2025), https://arxiv.org/abs/2502.12930
[10] Luxemburk J. et al.: CESNET-QUIC22: A large one-month quic network traffic dataset from backbone lines. Data in Brief (2023)
[11] Luxemburk J. et al.: Encrypted traffic classification: the quic case. In: TMA 2023. pp. 1–10 (2023). https://doi.org/10.23919/TMA58422.2023.10199052
[12] Plný R. et al.: WIF: Efficient library for network traffic analysis. In: 2024 20th International Conference on Network and Service Management (CNSM). p. 1–5. IEEE (Oct 2024). https://doi.org/10.23919/cnsm62983.2024.10814378
[13] Sedláček O. et al.: Fusing heterogeneous data for network asset classification – a two-layer approach. In: NOMS 2024-2024 IEEE Network Operations and Management Symposium (2024). https://doi.org/10.1109/NOMS59830.2024.10575154
[14] Stahlschmidt, R.: Multimodal deep learning for biomedical data fusion: a review. Briefings in Bioinformatics (2022)
[15] Uhříček D. et al.: BOTA: Explainable iot malware detection in large networks. IEEE Internet of Things Journal **10**(10) (2023). https://doi.org/10.1109/jiot.2022.3228816

[4]Available at https://github.com/plnyrich/DataFusionExperiments