Technical Appendix for: From Conflicts to Efficiency: Learning Constraint Orderings for Conflict Detection

1. Introduction

This document serves as a technical appendix for the main paper, "From Conflicts to Efficiency: Learning Constraint Orderings for Conflict Detection." The purpose of this appendix is to provide supplementary material that offers greater depth and transparency regarding our experimental methodology and results.

Herein, we present detailed information on our data preprocessing steps, the complete optimal hyperparameter configurations discovered for each model and dataset, visualizations of the MLP model training process via loss curves, and exhaustive performance metrics. This information is intended to support the findings presented in the main paper and to ensure the full reproducibility of our work.

2. Data Preprocessing Details

Before model training, we applied a standardized preprocessing pipeline to each dataset to simplify the learning task and improve model performance. This process involves removing input constraints (features) with low variance and output constraints (labels) that have a constant value across the entire dataset. This preprocessing was applied uniformly for both the Multi-Layer Perceptron (MLP) and the Decision Tree / Random Forest (DT/RF) models. Labels that were removed during this phase were programmatically re-inserted into the final predictions with their known constant value to ensure a complete output.

	Arcade	**BusyBox**	**B2C**
Constraints total	47	683	194
Removed Features	0	0	0
Removed Labels	3	539	72

Table 1: Summary of features and labels removed during preprocessing.

The threshold for removing low-variance features was set to 0.01. This means any input constraint that had the same value (e.g., always true or always false) for more than 99% of the samples would be removed. Such features provide little to no information for a machine learning model and can be safely discarded. Across all three datasets, no features met this removal criterion.

For the output labels, each dataset contained constraints that were never part of any Minimal Conflict Set (MCS) across all generated samples (i.e., had a constant value of 0). For the model, predicting a constant is a trivial task that offers no learning value. These constant-zero labels were therefore removed before training to allow the model to focus on the more complex, non-trivial relationships. As shown in Table 1, the BusyBox dataset was the most affected, with 78% of its labels being constant, while Arcade was the least affected (6%). When the trained model makes a prediction on new data, it correctly assigns a '0' for these removed labels, ensuring the final predicted MCS is complete and accurate.

3. Optimal Hyperparameter Configurations

The following tables detail the optimal hyperparameter configurations found using the Optuna library for our best-performing MLP and DT/RF models on each dataset. These configurations were selected based on their performance on the validation set and are provided here to ensure reproducibility.

MLP Configurations

Parameter Explanations (for Table 2:

- **Loss Function:** For the Arcade dataset, 'BCEWithLogitsLoss' (Binary Cross-Entropy) was optimal. For the more imbalanced BusyBox and B2C datasets, 'Focal Loss' was superior. Focal Loss is a modification of cross-entropy that reduces the loss assigned to well-classified examples, forcing the model to focus on harder, misclassified examples.
- **Focal Loss Gamma & Alpha:** These parameters are specific to Focal Loss. *Gamma* is the focusing parameter; a higher gamma value increases the down-weighting of easy examples. *Alpha* is a weighting factor that balances the importance of positive versus negative examples.
- **Common Parameters:** Across all datasets, the Adam optimizer and an early stopping patience of 20 proved to be universally effective. Batch normalization was consistently beneficial, helping to stabilize and accelerate training.

Hyperparameter	Arcade	BusyBox	B2C
Hidden Layers	[64, 64]	[128, 64]	[128, 64, 32]
Dropout Rate	None	None	None
Activation Func	Leaky ReLU	Leaky ReLU	ReLU
Batch Size	512	256	256
Batch Normalization	true	true	true
Max Epochs	200	300	300
Patience	20	20	20
Loss Func	BCEWithLogitsLoss	Focal Loss	Focal Loss
Optimizer	Adam	Adam	Adam
Learning Rate	0.039464	0.004512	0.001737
Weight Decay	0.001579	3.310980	7.453246
PCA	false	false	false
Focal Loss Gamma	None	3.304386	3.656835
Focal Loss Alpha	None	0.409502	0.451800

Table 2: Optimal hyperparameters for the best MLP model on each dataset.

Decision Tree / Random Forest Configurations

Parameter Explanations (for table 3):

- **Estimator Type:** The final model was a Random Forest for the Arcade dataset and a single Decision Tree for BusyBox and B2C.
- **Max Depth:** This parameter controls the maximum depth of the tree(s) to prevent overfitting. For B2C, an unlimited depth yielded the best results.
- **Number of Estimators:** This is specific to Random Forest and defines the number of trees in the forest. A value of 600 was optimal for Arcade.
- **Common Parameters:** The use of 'balanced' class weights was a consistent and critical choice across all datasets, helping the models handle the inherent imbalance in the data where non-conflict constraints vastly outnumber conflict constraints. The choice between 'ClassifierChain' and 'MultiOutputClassifier' varied, suggesting that the degree of correlation between labels differs across the knowledge bases.

4. MLP Training & Validation Loss Curves

The following plots illustrate the training and validation loss over epochs for the best-performing MLP model on each dataset. These curves provide insight into the learning dynamics and generalization capability of the models.

Analysis for Arcade (Figure 1): The training loss curve drops very rapidly, indicating that the model quickly learns the relationships between constraints. The initial gap between the training and validation loss suggests some early overfitting, but the curves converge to a low, stable value after approximately 150 epochs. The final proximity of the two curves shows that the model successfully generalized to the unseen validation data without overfitting. Even though the model ran the full max epochs that was set to 200, the two loss curves being a near straight line in the end and close to 0, meaning model has already had a very good performance and wont improve much more, so further training is also not necessary.

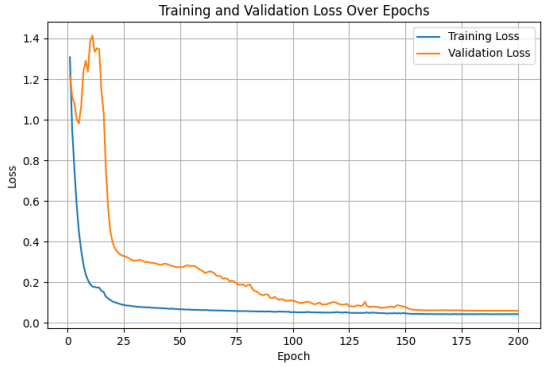

Figure 1: Training and validation loss for the Arcade dataset.

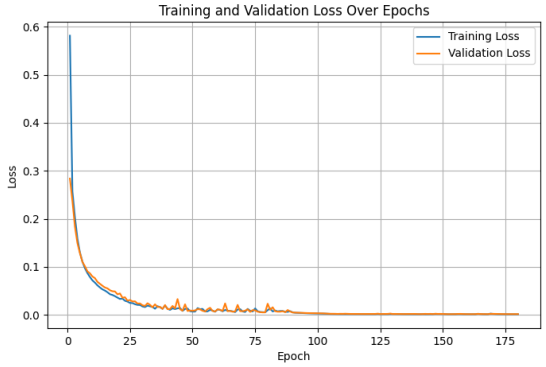

Figure 2: Training and validation loss for the B2C dataset.

Analysis for B2C (Figure 2): The loss curves for the B2C dataset demonstrate exceptionally fast learning, with both training and validation loss dropping to a near-zero value within the first 25 epochs. The curves then overlap almost perfectly as they continue to slowly improve until the early stopping mechanism halts training around epoch 175, which

Hyperparameter	Arcade	BusyBox	B2C
Estimator Type	Random Forest	Decision Tree	Decision Tree
Max Depth	35	80	None (unlimited)
Multi-Output Type	ClassifierChain	MultiOutputClassifier	ClassifierChain
PCA	false	false	false
Class Weight	balanced	balanced	balanced
Number of Estimators	600	None	None

Table 3: Optimal hyperparameters for the best DT/RF model on each dataset.

is a strong indicator that the model is not overfitting. Both curves having loss of almost exactly 0 means the model has learned a highly robust and accurate mapping from user requirements to conflict sets for this particular knowledge base.

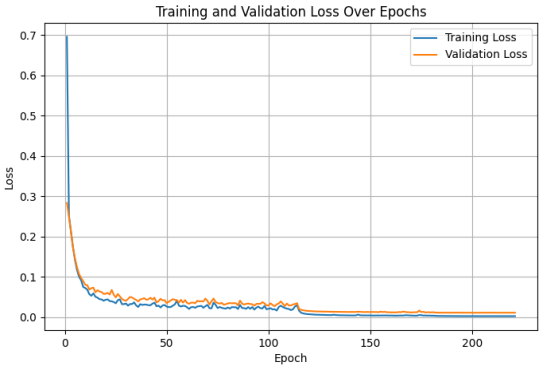

Figure 3: Training and validation loss for the BusyBox dataset.

Analysis for BusyBox (Figure 3): Similar to B2C, the model for the complex BusyBox dataset learns very quickly, with both loss curves dropping below 0.1 after just 25 epochs. The validation loss closely tracks the training loss throughout the process, albeit at a very slightly higher value, which is a sign of not overfitting. The loss of near 0 also indicates that the model is not underfitting neither. The early stopping mechanism was triggered around epoch 220 (max epoch for this dataset was set to 300), confirming that the model had converged to its optimal performance on the validation set.

5. Detailed Performance Results

This section presents the detailed performance metrics for our best-performing MLP and DT/RF models on the held-out test set for each knowledge base. Each test set is 10% of the whole dataset, chosen randomly, but in a way that still includes at least 1 sample from each unique Minimal Conflict Set (MCS), to ensure we test the model on all known MCS.

Direct Prediction Performance

Table 4 summarizes the performance of the models when used for direct prediction of the MCS, without the involvement of QUICKXPLAIN.

Metric Explanations:

- **Exact Match (direct prediction):** We take directly the prediction of the model for a MCS and compare it to the ground-truth MCS. This is a very strict metric that show the percentage of perfectly matched predictions (any prediction that has even just 1 constraint wrong will be considered wrong).
- **F1 / MCC / MAP:** Standard multi-label classification metrics. A score of 1.0 is perfect.
- **Hamming Loss:** The fraction of incorrectly predicted labels (lower is better).
- **ROC AUC:** The Area Under the Receiver Operating Characteristic Curve, measuring the model's ability to distinguish between positive and negative classes. A score of 1.0 is perfect.
- **Cosine Similarity:** Measures the overlap between the predicted and true MCS vectors. A value of 1.0 indicates a perfect match.
- Exact Match (after QX): This again tests how many times we can perfectly compute the preferred MCS that exactly matches the ground-truth MCS. Difference with the other 'exact match' metric is, that this is after we have reordered the constraints and got a MCS that was computed by QuickXplain.

Analysis: The results in Table 4 are exceptionally strong across the board. The Exact Match for direct prediction is above 92% for all models, indicating that these conflict detection problems can be solved with high accuracy by ML models alone. While both model families perform very well, the simpler DT/RF models often show a slight advantage in direct prediction accuracy, particularly on the complex BusyBox dataset. Moreover, the training time for Decision Tree (and also Random Forest) are noticeably faster than MLP, so they would be more beneficial to use for larger dataset. The MLP model, however, achieved near-perfection on the B2C dataset. The high scores across all other metrics (F1, MCC, etc.) further corroborate the models' robustness and reliability.

Runtime and Consistency Check Improvement

Table 5 details the end-to-end system performance, comparing the ML-guided QUICKXPLAIN against a baseline where constraints are ordered randomly.

Analysis: As shown in Table 5, both ML-guided approaches provide a substantial performance improvement over the

Metric	Arcade		BusyBox		B2C	
	MLP	**DT/RF***	**MLP**	**DT/RF***	**MLP**	**DT/RF***
Exact Match (direct prediction)	96.34%	**97.15%**	92.61%	**99.01%**	**99.85%**	97.95%
F1	0.99	0.99	0.98	**0.99**	0.99	0.99
MCC	**0.99**	0.98	0.97	**0.99**	**0.99**	0.98
MAP	0.99	0.99	**0.99**	0.97	**0.99**	0.98
Hamming Loss	0.0010	**0.0007**	0.0004	**0.0003**	**0.00001**	0.0007
ROC AUC	0.99	0.99	0.99	0.99	0.99	0.99
Cosine Similarity	99.70%	**99.86%**	99.75%	**99.83%**	**99.99%**	99.92%
Exact Match (after QX)	99.39%	**99.76%**	99.49%	**99.67%**	**99.99%**	99.68%

*The DT/RF model is a Random Forest for Arcade and a Decision Tree for BusyBox and B2C.

Table 4: Detailed performance metrics for MLP model's prediction on the test set. For each metric, the better-performing model family is shown in bold.

Metric	Arcade			BusyBox			B2C		
	MLP	**DT/RF**	**Random**	**MLP**	**DT/RF**	**Random**	**MLP**	**DT/RF**	**Random**
Runtime (s)	**0.0018**	0.0045	0.0086	**0.0357**	0.0398	0.0667	**0.0045**	0.0068	0.0104
Consistency Checks	**7.115**	9.565	17.272	**10.859**	11.165	18.887	**9.565**	9.936	17.929
Runtime Improvement*	**378.9%**	90.3%	-	**87%**	67.5%	-	**131%**	52.4%	-
CC Reduction	**58.8%**	44.6%	-	**42.5%**	40.9%	-	**46.7%**	44.6%	-

*Runtime improvement is calculated as follow: (Baseline - Model) / Model * 100%. This means, an improvement of 100% is 2 times faster, 200% is 3 times faster, 300% is 4 times faster, and so on.

Table 5: Runtime and Consistency Check (CC) performance comparison. Improvement percentages are relative to the random ordering baseline.

random ordering baseline, confirming that intelligent ordering significantly helps QUICKXPLAIN. A key finding is that the MLP models consistently resulted in a greater reduction in the number of consistency checks, and consequently, a faster average runtime. This demonstrates the MLP's superior ability to rank the constraints in an order that is highly effective for the divide-and-conquer algorithm, making it the preferred choice when raw speed is the primary objective.

Notes on Reproducibility

All source code, trained models, and results are available in https://doi.org/10.5281/zenodo.16739565.