A ALGORITHM

A.1 VERIX+

In this enhanced BINARYSEARCH algorithm, the solver (e.g., Marabou or Lirpa) is passed as an explicit parameter to enable the CHECK function, which performs the core verification queries.

Algorithm 2 BINARYSEARCH(f, x_Θ, solver)

function BINARYSEARCH(f, x_Θ, solver)
  if $|x_\Theta| = 1$ **then**
    if CHECK($f, x_B \cup x_\Theta$, solver) **then**
      $x_B \leftarrow x_B \cup x_\Theta$
      return
    else
      $x_A \leftarrow x_A \cup x_\Theta$
      return
    end if
  end if
  $x_\Phi, x_\Psi = \text{split}(x_\Theta, 2)$
  if CHECK($f, x_B \cup x_\Phi$, solver) **then**
    $x_B \leftarrow x_B \cup x_\Phi$
    if CHECK($f, x_B \cup x_\Psi$, solver) **then**
      $x_B \leftarrow x_B \cup x_\Psi$
    else
      if $|x_\Psi| = 1$ **then**
        $x_A \leftarrow x_A \cup x_\Psi$
      else
        BINARYSEARCH(f, x_Ψ, solver)
      end if
    end if
  else
    if $|x_\Phi| = 1$ **then**
      $x_A \leftarrow x_A \cup x_\Phi$
    else
      BINARYSEARCH(f, x_Φ, solver)
    end if
  end if
end function

A.2 SIMULTANEOUS ADD

Algorithm 3 Simultaneous Add

Input: model f, input x, candidate set $\mathcal{F}$, current free set $\mathcal{A}$, adversarial procedure ATTACK(,) property $\mathcal{P}$
Initialize: $\mathcal{E} \leftarrow \emptyset$ ▷ set of necessary features
for $i \in \mathcal{F} \setminus \mathcal{A}$ **do**
  $\mathcal{F}' \leftarrow \mathcal{F} \setminus \{i\}$
  if ATTACK($f, \Omega(x, \mathcal{F}'), \mathcal{P}$) succeeds **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{i\}$ ▷ i must remain fixed
  end if
end for
Return: $\mathcal{E}$

A.3 ITERATIVE SINGLETON FREEING

Algorithm 4 Iterative Singleton Free

Input: model f, input x, candidate set $\mathcal{F}$, free set $\mathcal{A}$, certificate method LIRPA(,) traversal order π, property $\mathcal{P}$
repeat
  `found` $\leftarrow$ **false**
  for $i \in \pi$ with $i \in \mathcal{F} \setminus \mathcal{A}$ **do**
    if LIRPA$(f, \Omega(x, \mathcal{A} \cup \{i\}), \mathcal{P})$ succeeds **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{i\}$
      `found` $\leftarrow$ **true**
      break ▷ restart scan from beginning of π
    end if
  end for
until `found` $=$ **false**
Return: $\mathcal{A}$

A.4 RECURSIVE SIMULTANEOUS FREE

Algorithm 5 Recursive Abstract Batch Freeing

Input: model f, input x, candidate set $\mathcal{F}$
Initialize: $\mathcal{A} \leftarrow \emptyset$ ▷ certified free set
repeat
  $\mathcal{A}_{best} \leftarrow \emptyset$
  for $m = 1 \ldots |\mathcal{F} \setminus \mathcal{A}|$ **do**
    $\mathcal{A}_m \leftarrow$ GREEDYABSTRACTBATCHFREEING$(f, \Omega^m(x; \mathcal{A}), \mathcal{F} \setminus \mathcal{A})$
    if $|\mathcal{A}_m| > |\mathcal{A}_{best}|$ **then**
      $\mathcal{A}_{best} \leftarrow \mathcal{A}_m$
    end if
  end for
  $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_{best}$
until $\mathcal{A}_{best} = \emptyset$
$\mathcal{A}$ = ITERATIVE SINGLETON FREE(f, x, $\mathcal{F}$, $\mathcal{A}$) ▷ refine by testing remaining features
Return: $\mathcal{A}$

B PROOF

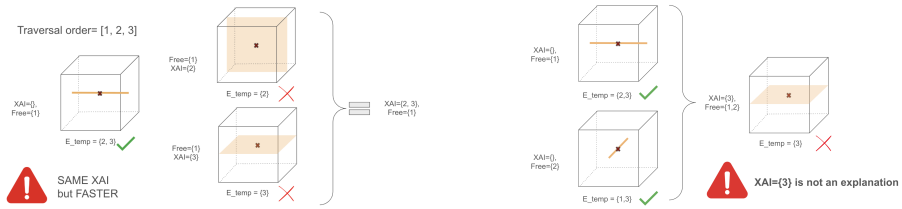
THE ASYMMETRY OF PARALLEL FEATURE SELECTION

Proposition B.1 (**Simultaneous Addition**)**.** Any number of essential features can be added to the explanation **simultaneously**. This property allows us to leverage solvers capable of assessing *multiple verification queries in parallel*, leading to a substantial reduction in runtime.

Proposition B.2 (**Simultaneous Freeing**)**.** The converse does *not* hold: it is unsound to free multiple features at once based only on individual verification as two features may be individually irrelevant yet jointly critical.

Simultaneous Addition B.1. Let $\mathcal{X}$ be the current explanation candidate, and let $\mathcal{R} = \{r_1, \ldots, r_k\}$ be a set of features not in $\mathcal{X}$. If, for every $r_i \in \mathcal{R}$, removing the single feature r_i from the set $\mathcal{F} \setminus (\mathcal{X} \cup \{r_i\})$ produces a counterexample, then all features in R are necessary and can be added to the explanation at once. □

Simultaneous freeing B.2. If removing any feature from a set $\mathcal{R} \subseteq \mathcal{F} \setminus \mathcal{X}$ individually causes the explanation to fail (i.e., produces a counterexample), then all features in $\mathcal{R}$ can be added to the explanation $\mathcal{X}$ simultaneously. □



(a) Adding several features at once is sound.

(b) Freeing several features at once is unsound.

Figure 5: Toy example illustrating the asymmetry between adding and freeing features.

Batch-Certifiable Freeing 4.1. For any $i \neq c$ and $x' \in \Omega(x)$, lirpa bounds give $f_i(x') - f_c(x') \leq \overline{b}^i(x) + \sum_{j \in \mathcal{A}} \Delta_{i,j}(x')$ with $\Delta_{i,j}(x') \leq c_{i,j}$. Taking the worst case over x' and i yields $f_i(x') - f_c(x') \leq \Phi(\mathcal{A}) \leq 0$, precluding a label flip.

□

PROPOSITION (CORRECTNESS OF THE RECURSIVE PROCEDURE)

Let $\mathcal{A}$ be the set returned by Algorithm 5 augmented with the final singleton refinement step that tests each remaining feature individually with the LiRPA certificate $\Phi(\cdot)$. Then:

(i) (No singleton extension) For every feature $j \in \mathcal{F} \setminus \mathcal{A}$ we have

$$\Phi(\mathcal{A} \cup \{j\}) > 0,$$

i.e. no single feature can be added to $\mathcal{A}$ while preserving the certificate. Hence $\mathcal{A}$ is *singleton-maximal* with respect to the LiRPA certificate.

(ii) (Termination) Algorithm 5 terminates in at most $|\mathcal{F}|$ outer iterations (and finitely many inner steps).

(iii) (Full abstract minimality — conditional) If the inner batch solver called by Algorithm 5 returns, for each tested budget p, a *globally optimal* certified free set (i.e., for the current domain it finds a maximum-cardinality $\mathcal{A}_p$ satisfying $\Phi(\mathcal{A}_p) \leq 0$), then the final $\mathcal{A}$ is a globally maximal certified free set: there is no $\mathcal{A}' \supsetneq \mathcal{A}$ with $\Phi(\mathcal{A}') \leq 0$. In this case $\mathcal{A}$ is a true minimal abstract explanation (with respect to the chosen LiRPA relaxation).

Proof. **(i) No singleton extension.** By construction, the algorithm performs a final singleton refinement: it tests every feature $j \in \mathcal{F} \setminus \mathcal{A}$ by evaluating the certificate on $\mathcal{A} \cup \{j\}$. The algorithm only adds j to $\mathcal{A}$ if $\Phi(\mathcal{A} \cup \{j\}) \leq 0$. Since the refinement ends with no further additions, it follows that for every remaining j we have $\Phi(\mathcal{A} \cup \{j\}) > 0$. This is exactly the stated property.

(ii) Termination. Each time the algorithm adds at least one feature to $\mathcal{A}$, the cardinality $|\mathcal{A}|$ strictly increases and cannot exceed $|\mathcal{F}|$. The outer loop therefore performs at most $|\mathcal{F}|$ successful additions. If an outer iteration yields no new features, the loop stops. Inner loops (scanning budgets p or performing singleton checks) are finite since they iterate over finite sets. Hence the algorithm terminates in finite time.

(iii) Full abstract minimality under optimal inner solver. Suppose that for every domain tested, the inner routine (called for each p) returns a certified free set of maximum possible cardinality among all subsets that satisfy $\Phi(\cdot) \leq 0$ on that domain. During each outer iteration the algorithm enumerates budgets p (or otherwise explores the space of allowed cardinalities) and selects the largest $\mathcal{A}_p$ found; then $\mathcal{A}$ is augmented by that largest globally-feasible batch. If no nonempty globally-feasible batch exists for any tested p, then no superset of the current $\mathcal{A}$ can be certified (because any superset would have some cardinality p' tested and the solver would have returned it). After the final singleton checks (which also use the optimal verifier on singletons), there remains no

single feature that can be added. Combining these facts yields that no superset of $\mathcal{A}$ is certifiable, i.e. $\mathcal{A}$ is a globally maximal certified free set, as claimed.

□

Abstract Minimal Explanation

Correctness of Iterative Singleton Freeing. Let $\mathcal{F}$ be the candidate feature set and let $\mathcal{A}_0 \subseteq \mathcal{F}$ be an initial free set such that the LiRPA certificate verifies $\mathcal{A}_0$ (i.e. $\Phi(\mathcal{A}_0) \leq 0$). Run the Iterative Singleton Freeing procedure (Algorithm 4) with traversal order π. The algorithm returns a set $\mathcal{A}$ with the following properties:

1. **(Soundness)** The final set $\mathcal{A}$ satisfies $\Phi(\mathcal{A}) \leq 0$ (every added singleton was certified).
2. **(Termination)** The algorithm terminates after at most $|\mathcal{F}| - |\mathcal{A}_0|$ successful additions (hence in finite time).
3. **(Singleton-maximality)** For every $j \in \mathcal{F} \setminus \mathcal{A}$ we have $\Phi(\mathcal{A} \cup \{j\}) > 0$, i.e. no remaining single feature can be certified as free.

□

Proof. **Soundness (invariant).** By assumption $\Phi(\mathcal{A}_0) \leq 0$. The algorithm only appends a feature i to the current free set after a LiRPA call returns success on $\mathcal{A} \cup \{i\}$, i.e. $\Phi(\mathcal{A} \cup \{i\}) \leq 0$. Since LiRPA certificates are sound, every update preserves the invariant "current $\mathcal{A}$ is certified". Therefore the final $\mathcal{A}$ satisfies $\Phi(\mathcal{A}) \leq 0$.

Termination. Each successful iteration increases $|\mathcal{A}|$ by one and $|\mathcal{A}| \leq |\mathcal{F}|$. Thus there can be at most $|\mathcal{F}| - |\mathcal{A}_0|$ successful additions. The algorithm halts when a full scan yields no addition; since scans iterate over a finite set ordered by π, the procedure terminates in finite time.

Singleton-maximality. Assume by contradiction that after termination there exists $j \in \mathcal{F} \setminus \mathcal{A}$ with $\Phi(\mathcal{A} \cup \{j\}) \leq 0$. The final scan that caused termination necessarily tested j (traversal order covers all remaining indices), so the algorithm would have added j, contradicting termination. Hence for every $j \in \mathcal{F} \setminus \mathcal{A}$ we must have $\Phi(\mathcal{A} \cup \{j\}) > 0$, proving singleton-maximality. □

Worked counterexample (illustrating joint freeing). Consider a toy binary classifier with two input features x_1, x_2 and property $\mathcal{P}$: the label remains class 0 iff $f_0(x') - f_1(x') \geq 0$. Suppose the LiRPA relaxation yields conservative linear contributions such that

$$\bar{b} + c_1 > 0, \qquad \bar{b} + c_2 > 0, \quad \text{but} \quad \bar{b} + c_1 + c_2 \leq 0,$$

where c_i is the worst-case contribution of feature i and $\bar{b}$ is the baseline margin. Then neither singleton $\{1\}$ nor $\{2\}$ is certifiable (each violates the certificate), but the joint set $\{1, 2\}$ is certifiable. The iterative singleton procedure terminates without adding either feature, while a batch routine (or an optimal MKP solver) would free both. This demonstrates the algorithm's limitation: it guarantees only singleton-maximality, not global maximality over multi-feature batches.

Complexity and practical cost. Let $n = |\mathcal{F}|$. In the worst case the algorithm may attempt a LiRPA call for every remaining feature on each outer iteration. If r features are eventually added, the total number of LiRPA calls is bounded by

$$(n) + (n-1) + \cdots + (n-r+1) \;=\; r \cdot n - \frac{r(r-1)}{2} \;\leq\; \frac{n(n+1)}{2} = \mathcal{O}(n^2).$$

Thus worst-case LiRPA call complexity is quadratic in n. In practice, however, each successful addition reduces the candidate set and often many iterations terminate early; empirical behavior tends to be much closer to linear in n for structured data because (i) many features are certified in early passes and (ii) LiRPA calls are highly parallelizable across features and can exploit GPU acceleration. Finally, the dominant runtime factor is the per-call cost of LiRPA (forward/backward bound propagation); therefore hybrid strategies (batch pre-filtering, prioritized traversal orders, occasional exact-solver checks on promising subsets) are useful to reduce the number of expensive LiRPA evaluations.

C EXAMPLES

Illustration of the Knapsack Formulation This is an example demonstrating how the greedy heuristic described in Algorithm 1 works. Given a multi-class classification problem with three classes: 0, 1, and 2. The model correctly predicts class 0 for a given input. We want to free features from the irrelevant set $\mathcal{A}$ based on the abstract batch certificate. We have three candidate features to free: j_1, j_2, and j_3. The baseline budgets for the non-ground-truth classes are:

- Class 1: $-\bar{b}^1 = 10$
- Class 2: $-\bar{b}^2 = 20$

The normalized costs for each feature are calculated as $c_{i,j}/(-\bar{b}^i)$:

Table 2: Example of Greedy Heuristic Decision Making

Feature (j)	**Normalized Cost for Class 1** ($c_{1,j}/(-\bar{b}^1)$)	**Normalized Cost for Class 2** ($c_{2,j}/(-\bar{b}^2)$)	**Maximum Normalized Cost** ($\max_i$)
j_1	$2/10 = 0.2$	$8/20 = 0.4$	0.4
j_2	$7/10 = 0.7$	$4/20 = 0.2$	0.7
j_3	$3/10 = 0.3$	$3/20 = 0.15$	0.3

The algorithm's objective is to minimize the maximum normalized cost across all non-ground-truth classes. As shown in the table, the minimum value in the "Maximum Normalized Cost" column is 0.3, which corresponds to feature j_3. Therefore, the greedy heuristic selects feature j_3 to be added to the free set in this step, as it represents the safest choice.

D EXPERIMENTS

D.1 MODEL SPECIFICATION

We evaluated our framework on standard image benchmarks including the MNIST(43) and GTSRB(38) datasets. We used both fully connected and convolutional models trained in a prior state-of-the-art VERIX+(40) to perform our analysis.

The MNIST dataset consists of $28 \times 28 \times 1$ grayscale handwritten images. The architectures of the fully connected and convolutional neural networks trained on this dataset are detailed in Table 3 and Table 4, respectively. These models achieved prediction accuracies of 93.76% for the fully connected model and 96.29% for the convolutional model.

Table 3: Architecture of the MNIST-FC model.

Layer Type	**Input Shape**	**Output Shape**	**Activation**
Flatten	$28 \times 28 \times 1$	784	-
Fully Connected	784	10	ReLU
Fully Connected	10	10	ReLU
Output	10	10	-

Table 4: Architecture of the MNIST-CNN model.

Layer Type	**Input Shape**	**Output Shape**	**Activation**
Convolution 2D	$28 \times 28 \times 1$	$13 \times 13 \times 4$	-
Convolution 2D	$13 \times 13 \times 4$	$6 \times 6 \times 4$	-
Flatten	$6 \times 6 \times 4$	144	-
Fully Connected	144	20	ReLU
Output	20	10	-

The GTSRB dataset contains colored images of traffic signs with a shape of 32×32×3 and includes 43 distinct categories. In the models used for our experiments, which were trained by the authors of VERIX+, only the 10 most frequent categories were used to mitigate potential distribution shift and obtain higher prediction accuracies. The architectures of the fully connected and convolutional models trained on GTSRB are presented in Table 5 and Table 6, respectively. These networks achieved prediction accuracies of 85.93% and 90.32%, respectively.

Table 5: Architecture of the GTSRB-FC model.

Layer Type	**Input Shape**	**Output Shape**	**Activation**
Flatten	$32 \times 32 \times 3$	3072	-
Fully Connected	3072	10	ReLU
Fully Connected	10	10	ReLU
Output	10	10	-

Table 6: Architecture of the GTSRB-CNN model.

Layer Type	**Input Shape**	**Output Shape**	**Activation**
Convolution 2D	$32 \times 32 \times 3$	$15 \times 15 \times 4$	-
Convolution 2D	$15 \times 15 \times 4$	$7 \times 7 \times 4$	-
Flatten	$7 \times 7 \times 4$	196	-
Fully Connected	196	20	ReLU
Output	20	10	-

D.2 Detailed Experimental Setup

We configured the VERIX+ implementation with the following settings: binary_search=true, logit_ranking=true, and traversal_order=bounds. To identify necessary features, we used the Fast Gradient Sign (FGS) technique for singleton attack addition, though the Projected Gradient Descent (PGD) is also available for this purpose.

D.3 Supplementary Experimental Results

PERFORMANCE WITH ITERATIVE REFINEMENT The three plots compare the performance of a greedy heuristic with an exact MILP solver for an iterative refinement task. The central finding across all three visualizations is that the greedy heuristic provides a strong trade-off between speed and solution quality, making it a more practical approach for large-scale problems.

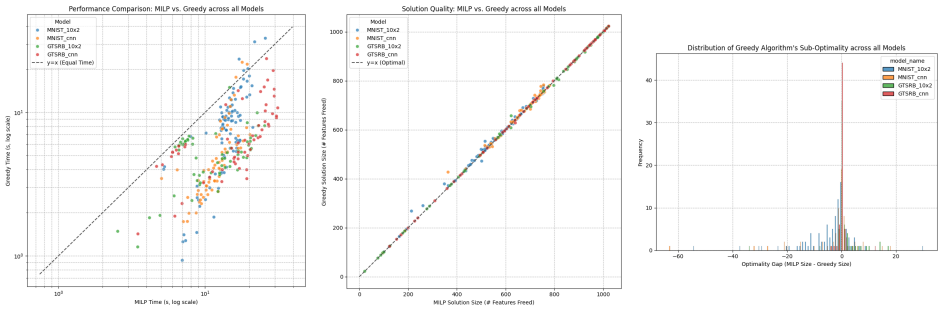


Figure 6: **Performance Comparison of FAME's Abstract Batch Freeing Methods.** These three plots compare the greedy heuristic against the exact MILP solver for the **iterative refinement** task for all the models. The first plot shows the runtime comparison of the two methods on a log-log scale. The second plot compares the size of the freed feature set for both methods. The third plot illustrates the distribution of the optimality gap (MILP size - Greedy size).

Analysis of FAME's Abstract Batch Freeing The visualizations demonstrate that the greedy heuristic provides a strong trade-off between speed and solution quality for the iterative refinement task.

- **Runtime Performance:** As shown in the first plot, the greedy algorithm is consistently faster than the MILP solver. This is evidenced by the data points for all models lying significantly below the diagonal line, confirming a substantial gain in runtime.
- **Solution Quality:** The second plot shows that the greedy algorithm produces solutions of comparable quality to the optimal MILP solver. The tight clustering of data points along the diagonal line for all models indicates a strong correlation between the sizes of the freed feature sets.
- **Optimality Gap:** The histogram of the final plot reinforces these findings by showing that the greedy heuristic frequently achieves the optimal solution, with the highest frequency of samples occurring at a gap of zero. The distribution further confirms that any sub-optimality is typically minimal.