

IMPATTO: A Static Analyzer for Quantitative Input Data Usage

Denis Mazzucato, Marco Campion, and Caterina Urban

INRIA & ENS | PSL, {denis.mazzucato,marco.campion,caterina.urban}@inria.fr

Abstract. We present IMPATTO, a static analyzer designed for quantitative verification of input data usage properties. IMPATTO combines a backward analysis with an impact calculator to quantify the influence of inputs on the output of a given program. The approach is modular, allowing the end-users to tailor the tool to their specific needs by choosing the most suitable setting. Indeed, IMPATTO is parametric in the choice of the underlying backward analyzer and impact measure. We offer a set of instantiations for both general purpose programs and neural network models. This paper details the architecture, functionality, and applications of IMPATTO.

1 Introduction

Nowadays, machine learning plays a crucial role by helping organizations and individuals make critical decisions based on data-driven insights in a variety of fields, including healthcare, finance, and autonomous driving [3]. However, the increasing reliance on machine learning introduces new challenges, as disastrous outcomes may result from programming errors in these safety-critical settings [6]. Unlike traditional software failures, subtle errors in data-intensive programs may produce seemingly plausible yet erroneous results. Such bugs are hard to spot since they provide no indication that something went wrong. One potential source of such issue arises when an input variable has an unexpected impact on the program computations compared to the developers' expectations.

In this paper, we present IMPATTO, a static analysis tool based on the quantitative framework for input data usage properties proposed by Mazzucato et al. [5]. Our tool leverages an underlying backward analyzer to compute the set of input-output relations of the program under analysis. This backward analyzer is a parameter of the tool, allowing different kind of analyses such as program or neural network analysis. Furthermore, the choice of the impact measure is also a parameter of the tool to better suit several factors, such as the program structure, the environment, and the intuition of the researcher.

Our work outlines IMPATTO¹ and its architecture. As a novel contribution, we extend the general purpose quantitative framework with novel impact measures tailored for neural network models, addressing the limitations of previous impact measures in capturing the irregularities of the input space of neural networks. Overall, this framework can be used to identify input features that have a disproportionate impact on a variety of contexts. Such knowledge could either certify intended behavior or reveal potential flaws, by matching the developers' intuition on the expected impact of their input with the actual result of our quantitative study.

¹ Publicly available at <https://github.com/denismazzucato/impatto>

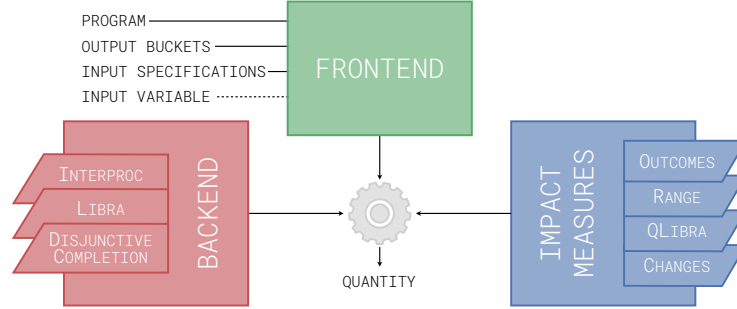


Fig. 1: Overview of IMPATTO.

2 Tool Architecture

IMPATTO is an open-source software implemented in PYTHON 3 and features a modular architecture, depicted in Figure 1, composed of a frontend for parsing input arguments, a backend hosting the backward analyzers, and a module dedicated to impact measures. The tool works starting from a finite set of *postconditions*, the output buckets, and performs a backward analysis of the given program to produce, for each bucket, an over-approximation of the input *preconditions*. Finally, this set of abstract input-output relations is used to measure the influence of the input variable of interest.

More in detail, the frontend handles input arguments, including the program under analysis, the input specification, and the composition of the output buckets. Optionally, the user can specify a single input variable of interest, otherwise the analysis considers all the input variables separately. From the current state of the backend, we support programs written in SPL (from INTERPROC²), and neural network models in PYTHON format (from LIBRA³). The input (resp. output) specifications consist of a set of linear inequalities that describe the input (resp. output) space of the program under analysis.

The backend of IMPATTO is a collection of backward engines. Currently, we support the INTERPROC² abstract interpreter, and two versions from LIBRA³ for verification of neural networks (indicated as LIBRA and DISJUNCTIVECOMPLETION in Figure 1). The first version is a sound forward-backward analysis [7, 4], and the second is a naïve disjunctive completion-based backward analysis. These engines return, for each output bucket, an over-approximation of input values in the form of a disjunction of conjunctions of linear inequalities among input variables.

The impact measures module is a collection of libraries, each implementing a specific impact measure. We currently provide four libraries, namely OUTCOMES, RANGE, QLIBRA, and CHANGES. The first two implement the impact measures proposed by Mazzucato et al. [5, Section 3]: OUTCOMES counts *how many* different output values are reachable solely by changing the value of the input variable of interest, while RANGE determines the *size of the range* of the reachable output values. QLIBRA is a quantitative adaptation of the input data usage proposed by Urban et al. [7]. It returns the *volume* of the input space for which changes to the value of the input variable of interest are

² <https://github.com/jogiet/interproc>

³ <https://github.com/caterinaurban/libra>

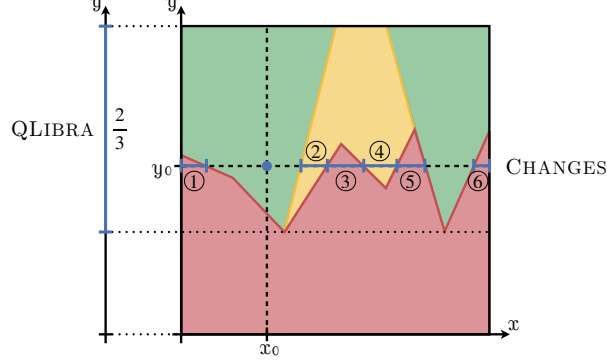


Fig. 2: Input space for a program with two input features (x and y) and three possible outcomes (green, yellow, red), and highlights of the elements determining the impact measures QLIBRA (left) and CHANGES (right).

able to change the output value. Finally, CHANGES is a novel impact measure that we propose in this paper: it counts how many input variable perturbations are able to change outcomes with repetitions, meaning that if a perturbation is able to change twice to the same outcomes in different contiguous regions they are counted separately. Graphically, the two latter impact measures are depicted in Figure 2, which shows the input space of a program with two input features x and y partitioned according to the corresponding output bucket (green, yellow, or red). If x is the input variable of interest, QLIBRA is the volume of the input space in which a perturbation of the value of x of a 2-dimensional point (e.g., (x_0, y_0)) is able to change the output bucket, which is roughly $\frac{2}{3}$ of the total input space. On the other hand, CHANGES is the maximum number of perturbations of the value of x that are able to change the outcome. In our example, depending on the chosen point, we are able to change from 0 to 6 outcomes with repetitions. For instance, the point (x_0, y_0) , leading to the green classification, achieves the maximum of CHANGES since perturbing the value of x leads to 4 red (①, ③, ⑤, and ⑥) and 2 yellow (② and ④) regions, for a total of 6 outcomes.

3 Evaluation

We showcase IMPATTO in the analysis of general purpose programs and neural networks.

Excel Spreadsheets. Program 1 is a simplified version of the Excel spreadsheet from the CHECKCHELL benchmark suite [1]. The program computes the weighted average for the final grade of a student given the grades of the homework assignments, quizzes and exams. The analysis results are shown in Table 1.

In the original work, Barowy et al. [1] proposed a stochastic analysis to discover possible transposition errors in this spreadsheet function, errors where digits are swapped within cells. The student grades are provided in advance for this analysis, [84,77,92,93,87,90,85,91,84,78]. We can encode such property by defining the input bounds as $v_i = x_{i,0}x_{i,1} \vee v_i = x_{i,1}x_{i,0}$, where v_i is the input variable and $x_{i,0}$ and $x_{i,1}$ are the two digits of the value of v_i . Furthermore, a student is considered to pass the

```

1 def final_grade(
2     HW1, HW2, HW3, HW4,
3     QZ1, QZ2, QZ3, QZ4,
4     EX1, EX2):
5     HW_coeff = 0.2
6     QZ_coeff = 0.3
7     EX_coeff = 0.5
8     HW_avg = HW_coeff * (HW1 + HW2 + HW3 + HW4) / 4
9     QZ_avg = QZ_coeff * (QZ1 + QZ2 + QZ3 + QZ4) / 4
10    EX_avg = EX_coeff * (EX1 + EX2) / 2
11    avg = HW_avg + QZ_avg + EX_avg

```

Program 1: Program computing the landing risk of an aircraft.

Table 1: Analysis results for Program 1.

Impact	Input space	Output buckets	HW1	HW2	HW3	HW4	QZ1	QZ2	QZ3	QZ4	EX1	EX2
OUTCOMES	SINGLE	BOOLEAN	0	0	1	1	0	1	1	1	1	1
OUTCOMES	FULL	ROUNDED	1	1	1	1	1	1	1	1	3	3
RANGE	FULL	ROUNDED	24	24	24	24	24	24	24	24	44	44

course if the final grade is greater than or equal to 85. The output buckets to encode this property are $\{\text{avg} \geq 85\}$ and $\{\text{avg} < 85\}$, respectively the student passes or fails the course. With the encoding presented above, OUTCOMES shows that single transposition errors (called SINGLE) could not affect the student’s graduation (called BOOLEAN) whenever they occur in the homework assignments HW1, HW2, or QZ1; as perturbations in these three variables do not affect the evaluation outcome, first row of Table 1.

Furthermore, we consider the input bounds to cover the full input space for the student grades, $0 \leq v_i \leq 100$, and the output buckets to encode the student final grade rounded up: $\{0 \leq \text{avg} < 15\}, \{15 \leq \text{avg} < 25\}, \dots, \{95 \leq \text{avg} \leq 100\}$. The analysis results are shown in the second and third row of Table 1, where the input space is called FULL and the output buckets ROUNDED. In particular, RANGE shows that changes in the homework assignments and quizzes can affect the final grade up to 24 units, while changes in the exam grades can affect the final grade up to 44 units (considering the final grade units between 0 and 100). The takeaway of the analysis results is that the final grade is partially affected by homework assignments and quizzes, while exam grades are far more impactful.

Feature Importance Metrics. In this section, we evaluate IMPATTO against *feature importance metrics*, i.e., stochastic quantitative measures of the influence of input variables in machine learning models. Specifically, we compare to:

- RFE A naïve feature importance metric that evaluates the changes in performance of a model when retrained without the feature of interest; in the following we call this metric *Retraining Feature Elimination*.
- PFI *Permutation Feature Importance* [2], one of the most popular model-agnostic feature importance metrics, which monitors changes in performance when the values of the feature of interest are randomly shuffled.

For the evaluation setup, we used public datasets from the Kaggleplatform to train several neural network models. We focused on four datasets: “Red Wine Quality”⁴, “Prima Indians Diabetes”⁵, “Rain in Australia”⁶, and “Cure the Princess”⁷. We pre-processed the “Rain in Australia” database and removed non-continuous input features, as our tool does not support discrete input features for neural networks yet. To preserve data consistency, since the majority of daily weather observations were collected in Canberra, we eliminated the observations from other stations. As a result of this pre-processing step, we retained approximately 2000 entries, aligning with the sizes of other datasets. We trained a total of about 700 networks by permuting the network structure and number of input features to obtain a uniform benchmark. The number of input features ranges from at least 3, to the number of attribute of the databases (after pre-processing), respectively, 10 attributes for “Red Wine Quality”, 8 for “Prima Indians Diabetes”, 17 for “Rain in Australia”, and 13 for “Cure the Princess”. The model size ranges from 2 to 6 hidden layers, each with 3 to 6 nodes, for a total of 6 to 36 nodes for each network model. All models were trained with Keras, using the Adam optimizer with the default learning rate, and binary crossentropy as the loss function. Each model was trained for 150 iterations. The obtained neural network accuracy usually depends on the chosen subset of input features which is usually lower than the accuracy achieved in the literature. However, we remark that our study focuses on the impact analysis, therefore high accuracy is not needed in our benchmarks. All models used in our experiments are open source as part of IMPATTO.

In order to evaluate IMPATTO, we first run it on each input feature separately, obtaining a sequence of impact quantities. Then, we compare the results for similarity with respect to the other feature importance metrics RFE and PFI. To this end, we employ the *maximum common prefix length* (MCPL), which is the length of the longest prefix shared among the indices of the two sequences sorted in descending order, formally:

$$\text{MCPL}(I, J) \stackrel{\text{def}}{=} \begin{cases} 1 + \text{MCPL}(\langle i_2, \dots \rangle, \langle j_2, \dots \rangle) & \text{if } I = \langle i_1, \dots, i_m \rangle \wedge J = \langle j_1, \dots, j_m \rangle \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $I = \text{argsort } X$ and $J = \text{argsort } Y$ are the indices of the two sequences X and Y sorted in descending order. The operator argsort returns the corresponding indices of the sorted sequence, e.g., $\text{argsort}(30, 65, 2, 60) = \langle 3, 1, 4, 2 \rangle$. The MCPL is a measure of similarity between two sequences, where we prioritize the elements with higher impact (cf., through the argsort operation).

Figure 3 shows the similarity results of the comparison between IMPATTO, RFE, and PFI with respect to the four datasets together. The x-axis represents the length of the longest prefix, while the y-axis represents the compound frequency of test cases with a prefix of at least the length of the x-axis. In summary, we notice that our static analyzer IMPATTO observe a higher degree of similarity with PFI than RFE. This can be noticed by the higher frequency of the bars in Figure 3a compared to Figure 3b. Furthermore, by inspecting the data we noted 75 cases in which IMPATTO and PFI obtained a complete match from the most impactful feature to the least impactful one, while RFE and PFI obtained a full match in 25 cases only. Moreover, IMPATTO and PFI did not match even the first most impactful feature in 323 cases, while RFE and PFI did not match in 480 cases. Notably, our approach is more similar to PFI, a state-of-the-art

⁴ <https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009>

⁵ <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

⁶ <https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package>

⁷ <https://www.kaggle.com/datasets/unmoved/cure-the-princess>

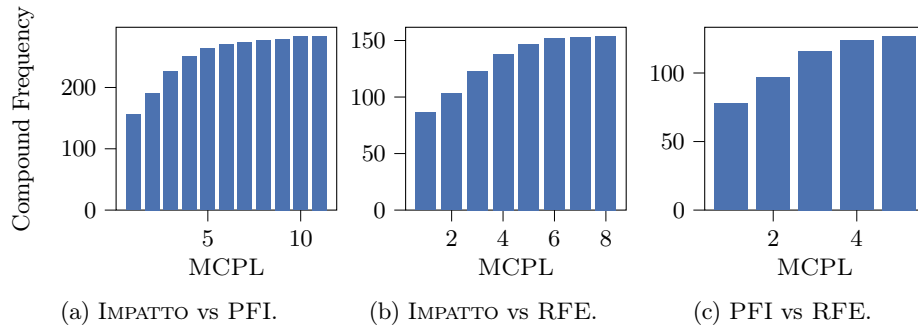


Fig. 3: Comparison between IMPATTO, RFE, and PFI by using the MCPL metric.

among feature importance metrics. Furthermore, IMPATTO stands out for its capability of addressing a formally defined impact property, e.g. *CHANGES*, providing a flexible framework that surpasses the hardcoded intuitions of the other two metrics. We show further comparisons with variations on similarity metrics and datasets in Appendix B.

4 Future Work

As future work, we plan to introduce heuristics able to automatically infer the output buckets, as their choice is essential for a useful analysis. Additionally, support for non-determinism at transition level is currently missing, and we plan to address this limitation in the future. Finally, new abstract domain implementations for the backward analysis could improve the precision of the analysis by focusing on (even non-linear) relations of input-output variables, rather than on the full state space.

ts at

References

- [1] D. W. Barowy, D. Gochev, and E. D. Berger. Checkcell: data debugging for spreadsheets. *OOPSLA 2014*. <https://doi.org/10.1145/2660193.2660207>.
- [2] L. Breiman. Random forests. *Machine Learning 2001*. <https://doi.org/10.1023/A:1010933404324>.
- [3] P. Kohli and A. Chadha. Enabling pedestrian safety using computer vision techniques: A case study of the 2018 uber inc. self-driving car crash. 2018. URL <http://arxiv.org/abs/1805.11815>.
- [4] D. Mazzucato and C. Urban. Reduced products of abstract domains for fairness certification of neural networks. *SAS 2021*. https://doi.org/10.1007/978-3-030-88806-0_15.
- [5] D. Mazzucato, M. Campion, and C. Urban. Quantitative Input Usage Static Analysis. 2023. URL <https://hal.science/hal-04339001>.
- [6] C. M. Reinhart and K. S. Rogoff. Growth in a time of debt. *American Economic Review 2010*. <https://doi.org/10.1257/AER.100.2.573>.
- [7] C. Urban, M. Christakis, V. Wüstholtz, and F. Zhang. Perfectly parallel fairness certification of neural networks. *OOPSLA 2020*. <https://doi.org/10.1145/3428253>.

A Demo Description

URL to the screencast of IMPATTO:

<https://github.com/denismazzucato/impatto#demo-screencast>

In this section, we present a demo of IMPATTO. Figure 4 shows the screenshots of IMPATTO presented in the demo. In particular, we show the helper menu, the analysis OUTCOMES for Program 1 with respect to both the single input variable **EX1**, and for all the input variables. Figure 5 shows the analysis of Program 1 with **-debug** mode on, this mode is useful to understand the behavior of IMPATTO. Finally, Figure 6 shows the analysis of a neural network model using **CHANGES**.

```

> python impatto.py --help
usage: impatto.py [-h] [-d] [-a IMPACT] [-e ENGINE] [-i VARIABLE] [--progress-bar]
                PROGRAM INPUTS.json BUCKETS.json

Impatto, A Static Analyzer for Quantitative Input Data Usage

positional arguments:
  PROGRAM              path to the input program PROGRAM
  INPUTS.json          path to the INPUTS.json file for input specifications
  BUCKETS.json         path to the BUCKETS.json file for output buckets

options:
  -h, --help            show this help message and exit
  -d, --debug           activate debug mode
  -a IMPACT, --analysis IMPACT
                        impact analysis: outcomes, range, unused, changes
                        default: outcomes
  -e ENGINE, --engine ENGINE
                        backward engines: interproc, interproc-fast, interproc-strong, libra,
                        disjunctive-completion
                        default: interproc
  -i VARIABLE, --interest VARIABLE
                        variable of interest
                        default: all
  --progress-bar        show progress bar

```

```

> python impatto.py samples/examples/excel.spl samples/inputs/excel-transposition.json \
samples/buckets/excel-boolean.json --analysis outcomes --engine interproc --interest EX1
INFO: Computing backward analysis...
INFO: Analysis results:
Running "OutcomesAnalysis" for variable "EX1":
1

```

```

> python impatto.py samples/examples/excel.spl samples/inputs/excel-transposition.json \
samples/buckets/excel-boolean.json --analysis outcomes --engine interproc
INFO: Computing backward analysis...
INFO: Analysis results:
Running "OutcomesAnalysis" for variable "HW1":
0
Running "OutcomesAnalysis" for variable "HW2":
0
Running "OutcomesAnalysis" for variable "HW3":
1
Running "OutcomesAnalysis" for variable "HW4":
1
Running "OutcomesAnalysis" for variable "QZ1":
0
Running "OutcomesAnalysis" for variable "QZ2":
1
Running "OutcomesAnalysis" for variable "QZ3":
1
Running "OutcomesAnalysis" for variable "QZ4":
1
Running "OutcomesAnalysis" for variable "EX1":
1
Running "OutcomesAnalysis" for variable "EX2":
1
Running "OutcomesAnalysis" for variable "avg":
0

```

Fig. 4: Some screenshots of IMPATTO in action.

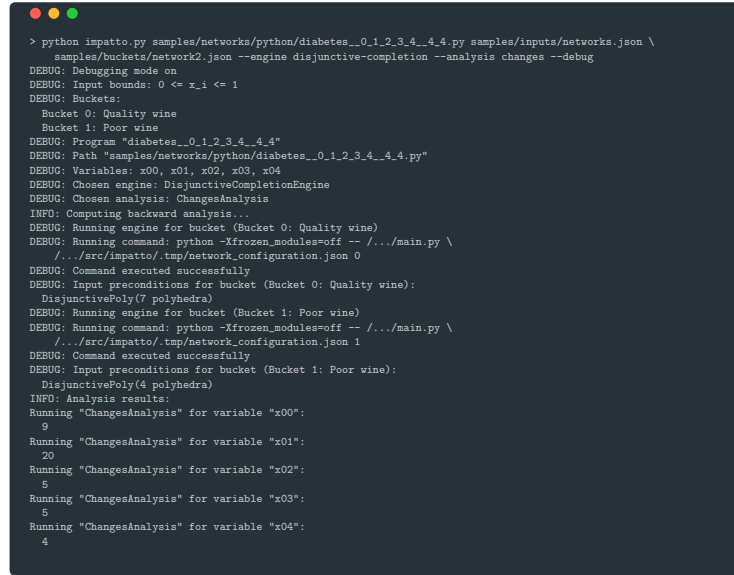
```

> python impatto.py samples/examples/excel.spl samples/inputs/excel-transposition.json \
  samples/buckets/excel-boolean.json --analysis outcomes --engine interproc --debug
DEBUG: Debugging mode on
DEBUG: Input bounds:
HW1 == 84 or HW1 == 48
HW2 == 77 or HW2 == 77
HW3 == 92 or HW3 == 29
HW4 == 93 or HW4 == 39
QZ1 == 87 or QZ1 == 78
QZ2 == 90 or QZ2 == 09
QZ3 == 85 or QZ3 == 88
QZ4 == 91 or QZ4 == 19
EX1 == 84 or EX1 == 48
EX2 == 78 or EX2 == 87
DEBUG: Buckets:
Bucket 0: avg < 85
Bucket 1: avg >= 85
DEBUG: Program "excel"
DEBUG: Path "samples/examples/excel.spl"
DEBUG: Variables: HW1, HW2, HW3, HW4, QZ1, QZ2, QZ3, QZ4, EX1, EX2, avg
DEBUG: Chosen engine: InterprocEngine
DEBUG: Chosen analysis: OutcomesAnalysis
DEBUG: No loop to unroll
INFO: Computing backward analysis...
DEBUG: Running engine for bucket (Bucket 0: avg < 85)
DEBUG: Running interproc:
> /.../interproc-analysis fbf -domain polkastrict -widening 5 5 -display text \
  /.../impatto/tmp/excel_inputbounds_bucket0.spl
DEBUG: Command executed successfully
DEBUG: Input preconditions for bucket (Bucket 0: avg < 85):
HW2 <= 77 &&
-HW2 <= -77 &&
504*QZ1 + 56*QZ2 + 63*QZ4 + 126*EX1 + 504*EX2 + 84*HW4 + 126*HW1 + 72*HW3 <= 129537 &&
56*QZ2 + 168*QZ3 + 63*QZ4 + 126*EX1 + 504*EX2 + 84*HW4 + 72*HW3 <= 89385 &&
252*QZ1 + 56*QZ2 + 84*QZ3 + 63*QZ4 + 126*EX1 + 252*EX2 + 84*HW4 + 63*HW1 + 72*HW3 <= 87537 &&
504*QZ1 + 56*QZ2 + 168*QZ3 + 63*QZ4 + 126*EX1 + 84*HW4 + 126*HW1 + 72*HW3 <= 99969 &&
EX1 <= 84 &&
EX2 <= 87 &&
HW1 <= 84 &&
HW3 <= 92 &&
HW4 <= 39 &&
QZ1 <= 87 &&
QZ2 <= 90 &&
QZ3 <= 85 &&
QZ4 <= 91 &&
-QZ4 <= -19 &&
-QZ3 <= -58 &&
-QZ2 <= -9 &&
-QZ1 <= -78 &&
-HW4 <= -39 &&
-HW3 <= -29 &&
-HW1 <= -48 &&
-EX2 <= -78 &&
-EX1 <= -48
DEBUG: Running engine for bucket (Bucket 1: avg >= 85)
DEBUG: Running interproc:
> /.../interproc-analysis fbf -domain polkastrict -widening 5 5 -display text \
  /.../impatto/tmp/excel_inputbounds_bucket1.spl
DEBUG: Command executed successfully
DEBUG: Input preconditions for bucket (Bucket 1: avg >= 85):
QZ4 <= 91 &&
-QZ4 <= -91 &&
QZ3 <= 85 &&
-QZ3 <= -85 &&
QZ2 <= 90 &&
-QZ2 <= -90 &&
HW4 <= 93 &&
-HW4 <= -93 &&
HW3 <= 92 &&
-HW3 <= -92 &&
HW2 <= 77 &&
-HW2 <= -77 &&
HW1 <= 84 &&
-HW1 <= -84 &&
EX2 <= 87 &&
-EX2 <= -87 &&
EX1 <= 84 &&
-EX1 <= -84 &&
QZ1 <= 87 &&
-QZ1 <= -78
INFO: Analysis results:
Running "OutcomesAnalysis" for variable "HW1":
0
Running "OutcomesAnalysis" for variable "HW2":
0
Running "OutcomesAnalysis" for variable "HW3":
1
Running "OutcomesAnalysis" for variable "HW4":
1
Running "OutcomesAnalysis" for variable "QZ1":
0
Running "OutcomesAnalysis" for variable "QZ2":
1
Running "OutcomesAnalysis" for variable "QZ3":
1
Running "OutcomesAnalysis" for variable "QZ4":
1
Running "OutcomesAnalysis" for variable "EX1":
1
Running "OutcomesAnalysis" for variable "EX2":
1
Running "OutcomesAnalysis" for variable "avg":
0

```

Fig. 5: Analysis run of Program 1 with --debug mode on.

It is interesting to note the debugging information provided by IMPATTO in Figure 5. In particular, IMPATTO starts by showing the chosen setup for the current execution, i.e., the input specifications, the composition of output buckets, the program under analysis, the variable of interest, the chosen engine, and the chosen analysis. Then, IMPATTO performs the backward analysis for each bucket, logging each time the executed command and the analysis result. The command executed to run the engine can be useful to load the same engine configuration for external use. The backward analysis result is shown entirely whenever the disjunction of conjunction of linear inequalities is composed by a single conjunction, as in the screenshot above. Finally, IMPATTO outputs the analysis results for each variable of the program under analysis (or only the single variable of interest if specified). Note that, there is no discrimination between input variables and internal variable as they all belong to the memory state.



```
> python impatto.py samples/networks/python/diabetes__0_1_2_3_4__4.py samples/inputs/networks.json \
samples/buckets/network2.json --engine disjunctive-completion --analysis changes --debug
DEBUG: Debugging mode on
DEBUG: Input bounds: 0 <= x_i <= 1
DEBUG: Buckets:
  Bucket 0: Quality wine
  Bucket 1: Poor wine
DEBUG: Program "diabetes__0_1_2_3_4__4.py"
DEBUG: Path "samples/networks/python/diabetes__0_1_2_3_4__4.py"
DEBUG: Variables: x00, x01, x02, x03, x04
DEBUG: Chosen engine: DisjunctiveCompletionEngine
DEBUG: Chosen analysis: ChangesAnalysis
INFO: Computing backward analysis...
DEBUG: Running engine for bucket (Bucket 0: Quality wine)
DEBUG: Running command: python -Ifrozen_modules=off -- ./../main.py \
./../src/impatto/tmp/network_configuration.json 0
DEBUG: Command executed successfully
DEBUG: Input preconditions for bucket (Bucket 0: Quality wine):
  DisjunctivePoly(7 polyhedra)
DEBUG: Running engine for bucket (Bucket 1: Poor wine)
DEBUG: Running command: python -Ifrozen_modules=off -- ./../main.py \
./../src/impatto/tmp/network_configuration.json 1
DEBUG: Command executed successfully
DEBUG: Input preconditions for bucket (Bucket 1: Poor wine):
  DisjunctivePoly(4 polyhedra)
INFO: Analysis results:
Running "ChangesAnalysis" for variable "x00":
9
Running "ChangesAnalysis" for variable "x01":
20
Running "ChangesAnalysis" for variable "x02":
5
Running "ChangesAnalysis" for variable "x03":
5
Running "ChangesAnalysis" for variable "x04":
4
```

Fig. 6: Analysis of a neural network model using CHANGES.

Figure 6 shows the analysis of a neural network model using CHANGES with the debugging logs enabled. The neural network under analysis is composed of 5 input features, 4 hidden layers with 4 neurons each, and 2 output neurons. We used the “Red Wine Quality” dataset to train it, thus the two buckets correspond to either quality or poor wine.

There are two main differences from the screenshot of IMPATTO for program analysis. First, input variables are distinguished from internal variables as neural networks have a rigid model structure. Second, the analysis results are, in this instance, disjunctions of more than one conjunction of inequalities, thus we decided to only shows the size of such set.

B Full Experimental Overview

This section contains the full overview of our experiment. All the figures are organized in a 4 rows by 3 columns setup, we dedicated one page for the experiments of each dataset. Figure 7 refers to “Prima Indians Diabetes” dataset, 8 to “Red Wine Quality”, 9 to “Rain in Australia”, and 10 to “Cure the Princess”. Each column corresponds to a different comparison: from left to right, we show the comparison between IMPATTO vs PFI, IMPATTO vs RFE, and PFI vs RFE.

The row refers to a different similarity metric, namely, from top to bottom, maximum common prefix length (MCPL, already defined in Eq. (1)), relaxed maximum common prefix length (RMCPL), Euclidean distance (ED), and Manhattan distance (MD). Given two analyses F, F' (e.g., IMPATTO and PFI), any similarity metric (e.g., MCPL) first sorts the result of the analyses F, F' applied to all the input features, by decreasing order. Formally, it computes $I = \text{argsort } F(M)$ and $J = \text{argsort } F'(M)$ for the two analyses respectively, where M is the neural network under consideration and argsort returns the corresponding indices of the sorted list (by decreasing order), e.g., $\text{argsort}\langle 30, 65, 2, 60 \rangle = \langle 3, 1, 4, 2 \rangle$. Afterwards, (MCPL) retrieves the length of the maximal common prefix between I and J , as shown in Eq. (1). For instance, assuming $I = \langle 4, 1, 2, 3, 5 \rangle$ and $J = \langle 4, 1, 2, 5, 3 \rangle$, $\text{MCPL}(I, J) = 3$ since the maximum common prefix is $\langle 4, 1, 2 \rangle$ of length 3. The relaxed variation (RMCPL) allows a 10% of overlap among quantity values, e.g., given $[F(M) \mid i \in \Delta] = \langle 0.4, 0.95, 0.6, 1 \rangle$ and $[F'(M) \mid i \in \Delta] = \langle 30, 65, 2, 60 \rangle$, we obtain $I = \text{argsort}\langle 0.4, 0.95, 0.6, 1 \rangle = \langle 4, 2, 3, 1 \rangle$ and $J = \text{argsort}\langle 30, 65, 2, 60 \rangle = \langle 2, 4, 1, 3 \rangle$, hence $\text{MCPL}(I, J)$ would return 0. However, a 10% of margin of error permits to swap the indices of values 0.95 and 1 in the first list obtaining $\langle 2, 4, 3, 1 \rangle$, we say that I is equivalent to $\langle 2, 4, 3, 1 \rangle$ (written $I \simeq \langle 2, 4, 3, 1 \rangle$), hence $\text{MCPL}(\langle 2, 4, 3, 1 \rangle, J) = 2$. Formally, we define RMCPL as the maximum of MCPL of all the possible equivalences:

$$\text{RMCPL}(I, J) \stackrel{\text{def}}{=} \max\{\text{MCPL}(I', J') \mid I' \simeq I \wedge J' \simeq J\}$$

The Euclidean distance is defined as $\text{ED}(I, J) \stackrel{\text{def}}{=} \sqrt{\sum_k (I_k - J_k)^2}$, and the Manhattan distance as $\text{MD}(I, J) \stackrel{\text{def}}{=} \sum_k |I_k - J_k|$.

Similarly to the evaluation section, the x-axis represents the length of the longest prefix, while the y-axis represents the frequency of test cases with a prefix of *exactly* the length of the x-axis.

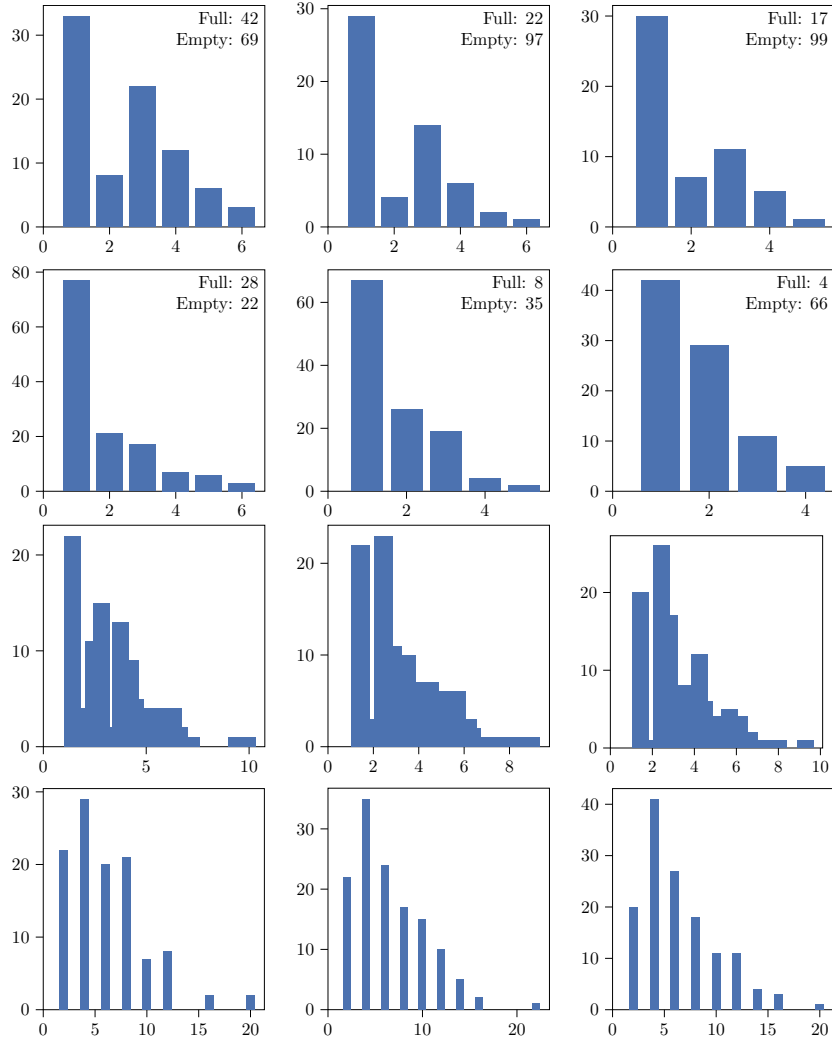


Fig. 7: Experimental overview regarding the “Prima Indians Diabetes” dataset.

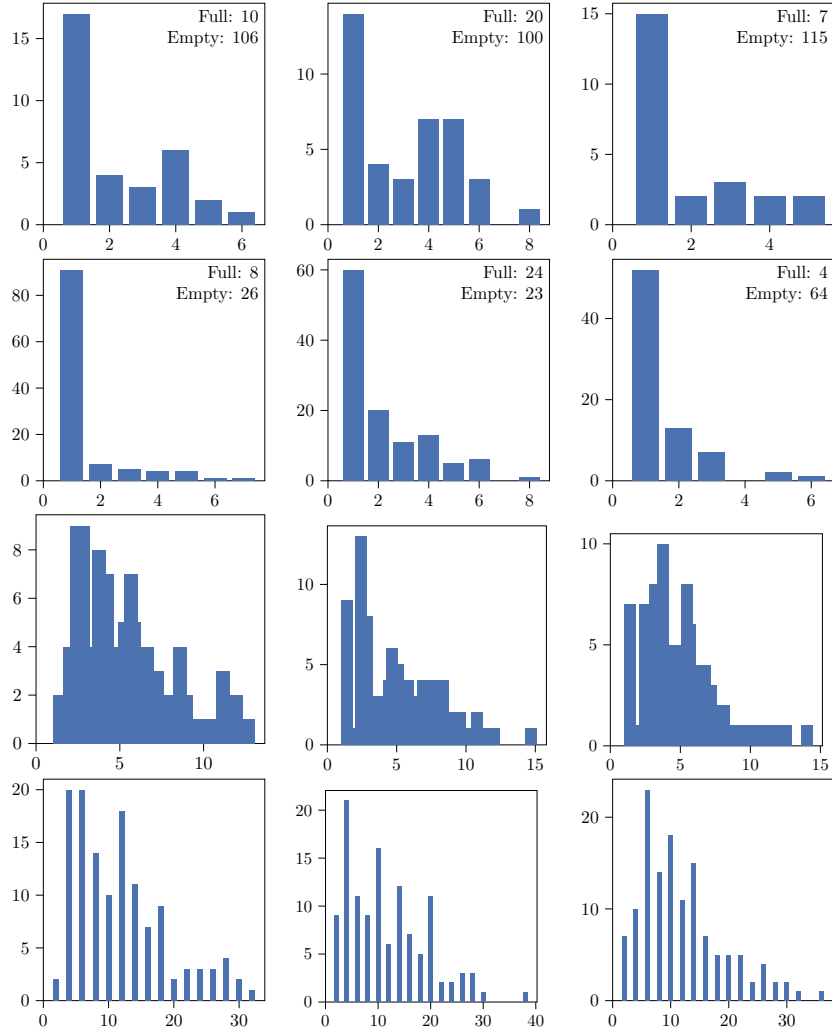


Fig. 8: Experimental overview regarding the “Red Wine Quality” dataset.

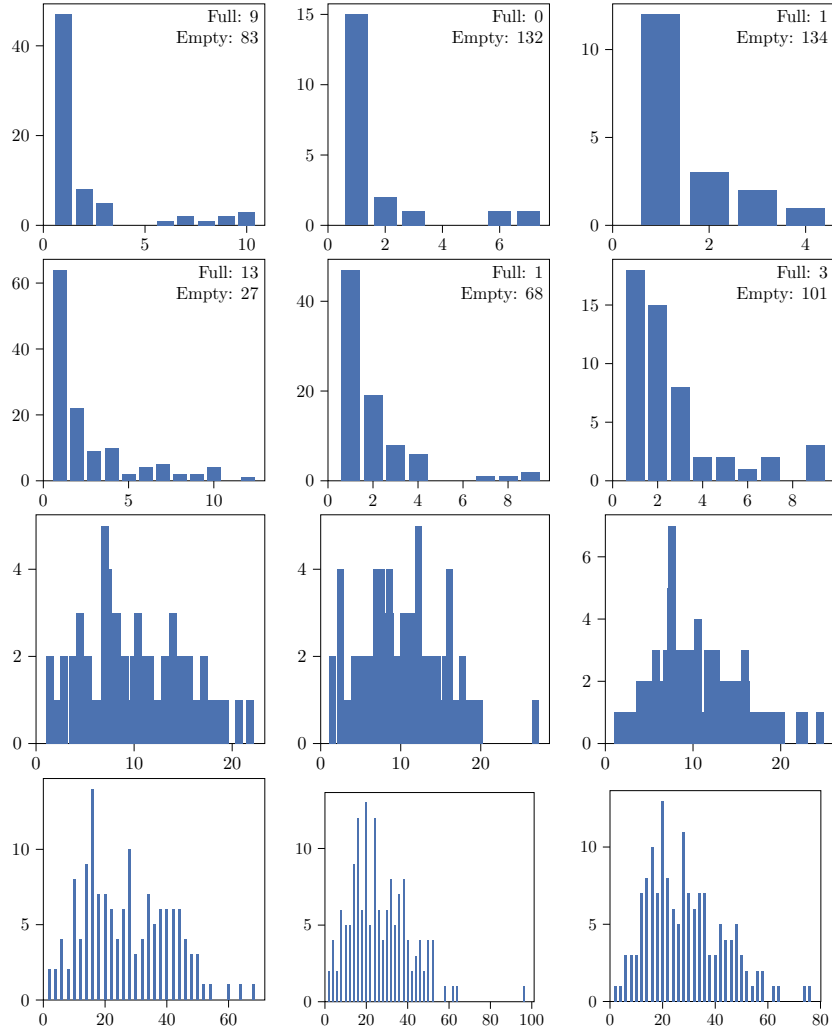


Fig. 9: Experimental overview regarding the “Rain in Australia” dataset.

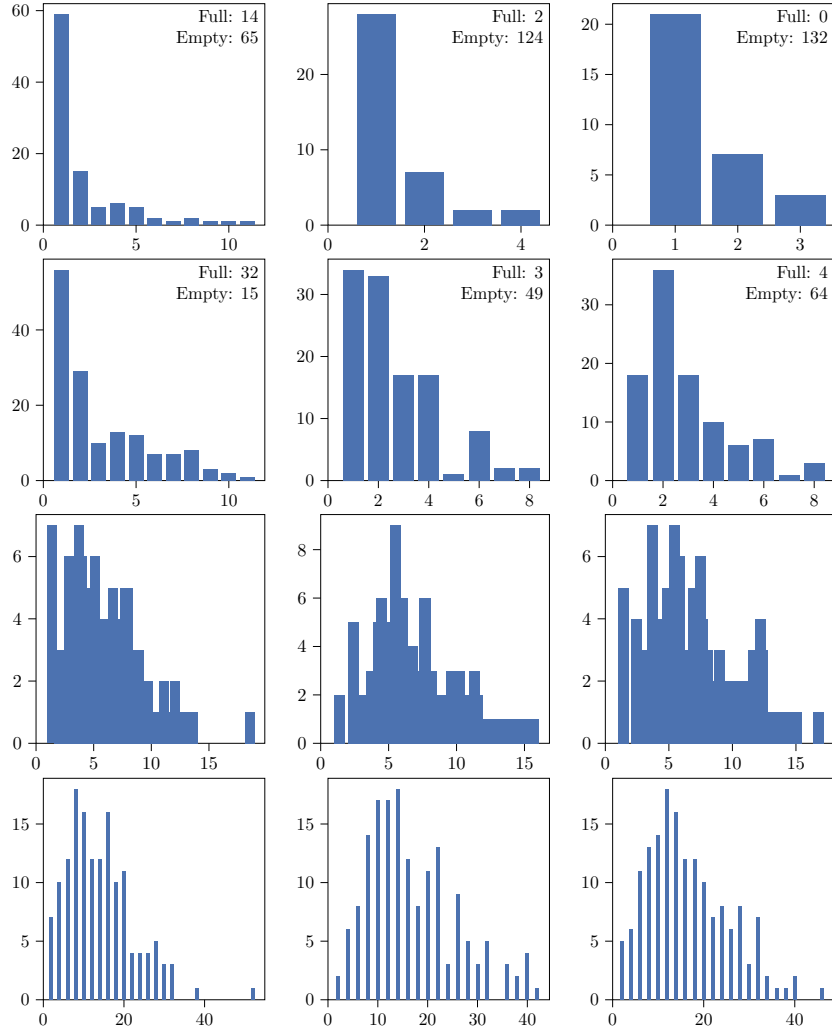


Fig. 10: Experimental overview regarding the “Cure the Princess” dataset.