AGNOMIN - Architecture Agnostic Multi-Label Function Name Prediction

Supplementary Material

A. More on Experimental Design

1) Dataset Statistics: The Diverse Architecture Binary (DAB) dataset, used for training and evaluating AGNOMIN, comprises a total of 9,000 ELF binaries across three architectures: `amd64`, `armel`, and `i386`. Table I summarizes the data statistics of the *DAB* dataset, including the number of packages and binaries for each architecture, as well as the average number of function nodes, edges, and basic block (BB) nodes per binary. These statistics provide insights into the scale and complexity of the dataset, highlighting the diverse range of binaries and architectures covered. The substantial number of function nodes, edges, and basic block nodes underscores the importance of AGNOMIN's ability to effectively represent and analyze these complex structures, further emphasizing the significance of the Feature-Enriched Hierarchical Graph (FEHG) approach.

TABLE I
THE STATISTICS OF DAB DATASET.

Archi	Packages	Binaries	Functions	Edges in FEHG	Basic Blocks	Edges in FECFG
`armel`	351	3000	133341	226252	1733639	2409644
`i386`	351	3000	135925	263099	1555106	2170903
`amd64`	351	3000	126830	222203	1557687	2179100
DAB	1053	9000	396096	711554	4846432	6759647

2) Encoder Training Configuration: Table II depicts the training configuration for different `Pcode` configuration.

TABLE II
EXPERIMENTAL CONFIGURATIONS.

`PCode` Features	32	64	128
none	-	-	-
bb	bb 32	bb 64	bb 128
func	func 32	func 64	func 128
bb-func	bb-func 32	bb-func 64	bb-func 128

3) Training Setup: We conducted all experiments on a server equipped with Intel Core i7-7820X CPUs @ 3.60GHz and 16GB RAM. To leverage the computational power of GPUs for model training and inference, we utilized three different GPU configurations: NVIDIA Tesla V100, A30 Tensor Core GPU, and A100 Tensor Core GPU.

Table III shows the approximate time required for training and inference tasks on each GPU configuration. As can be observed, the training process for AGNOMIN's models can be computationally intensive, taking several hours depending on the GPU configuration. However, once trained, the inference time for function label prediction on a single binary is remarkably fast, taking only a minute or less.

TABLE III
TIME TAKEN FOR TRAINING AND INFERENCE TASKS ON THREE DIFFERENT GPU CONFIGURATIONS AND ON CPU. THE INFERENCE IS MADE ON ONE BINARY.

Model	Tasks	Compute Resource	Runtime
Encoder	Training	V100	46h 03m 16s
		A30	35h 11m 0s
		A100	32h 17m 58s
	Inference	V100	0.575s
		A30	0.445s
		A100	0.6563s
Decoder	Training	V100	13h 26m 38s
		A30	8h 56m 49s
		A100	8h 55m 36s
	Inference	V100	42.881
		A30	41.881
		A100	36.881

This efficient inference time is a crucial factor in ensuring AGNOMIN's practical utility as a reverse engineering tool. When integrated into frameworks like Ghidra, AGNOMIN can provide near real-time function name predictions without introducing significant delays in the reverse engineering workflow. This seamless integration allows reverse engineers to leverage AGNOMIN's capabilities effectively, enhancing their productivity and efficiency in analyzing stripped binaries.

B. Label Distribution

Figure 1 (left) shows the distribution of function labels in the *DAB* dataset when considering the entire label space of 4096 labels. As evident from the figure, there is a significant class imbalance, where some labels appear frequently while others occur only a few times. This imbalance is a common challenge in multi-label classification tasks, as models can tend to focus on predicting the most common labels, potentially neglecting the rare but informative labels.

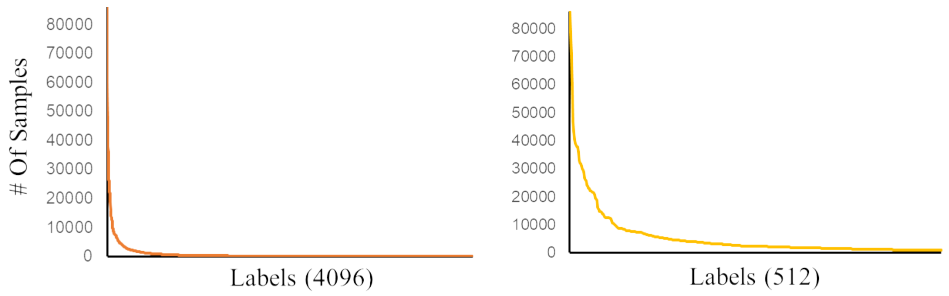

Fig. 1. Label distribution.

To mitigate the label scarcity problem and ensure that the decoder has sufficient examples to learn from, AGNOMIN employs a label space pruning strategy. By retaining only the N most common labels, with N typically set in the range of 512 to 4,096, AGNOMIN focuses on the most frequently occurring labels, reducing the impact of rare labels on the training process. Figure 1 (right) shows the label distribution after pruning the label space to the 512 most common labels. This pruning has lessened the imbalance in the label distribution.

To address the remaining imbalance and the tail-label prediction problem, AGNOMIN employs propensity-based metrics, such as Propensity-Scored Precision (PSP), Propensity-Scored Discounted Cumulative Gain (PSDCG), and Propensity-Scored Normalized Discounted Cumulative Gain (PSnDCG). These metrics incorporate propensity scores that are inversely proportional to the label's frequency in the training dataset, effectively rewarding the model for successfully predicting rare labels.

C. More Experimental Results

1) Effects of Features Extracted from `PCode`: Table IV shows the effect of the number of features extracted from `PCode` on the decoder model's performance. To determine the optimal `PCode` configuration for AGNOMIN's decoder, we compared its performance across different `PCode` feature extraction levels.

As the table demonstrates, extracting 128 features per BB and function call consistently yields higher precision than extracting 32 or 64 features. This suggests that a higher number of features better captures the semantics of BBs and function calls, enabling the decoder to make more accurate predictions.

To determine the best `PCode` configuration for the decoder within AGNOMIN, we compare the performance of the decoder with different `PCode` configurations. As we can see from the table, extracting 128 features per BB and function call consistently yields higher precision than 32 or 64 features. This suggests more features better capture the semantics of BBs and function calls.

TABLE IV
EFFECTS OF FEATURES EXTRACTED FROM `PCODE`.

# BB_Func	P@1	P@5	PSP@5	nDCG@5	R@5
128	92.11%	52.59%	89.63%	90.91%	91.15%
64	91.96%	52.19%	89.20%	90.13%	91.22%
32	90.12%	52.58%	89.06%	88.86%	89.02%

2) Examples of AGNOMIN's Predictions: Some examples of AGNOMIN's function label predictions are depicted in Table V.

TABLE V
SOME EXAMPLES OF AGNOMIN'S PREDICTIONS.*

Predicted[1]					Ground Truth[2]				
multiplication	information	one	nr	resolve	resolve	nr	one	multiplication	information
multiplication	pf				pf	multiplication			
cs	clone	so	main		cs	clone	so		
register	tm	get	test	cs	test	cs	clone		
so	main				so				
pt	wrap	type	res		res	pt	type	wrap	
read	new	quit			read				
ae	weak	vi	buffer	unknown	ae	weak	unknown	vi	buffer
register	tm	get	test	cs	bin	tm	register	get	
register	tm	get	test	cs	tm	register	get		
lib	string	in	hmac	fin	in	string	fin	hmac	lib
key	dumb				dumb	key			

* More examples can be found in the supplementary materials.
[1] Words written in red are predicted by the model but not found in the ground truth.
[2] Words written in blue are found in the ground truth but not predicted by the model.

D. Case Study: Challenge Problem Description

In the challenge problem evaluated as a case study, a simulated rover became trapped while inspecting a rock (see Fig. 2). Its existing preprocessing binary caused its motors to move forward, preventing the rover from maneuvering around the obstacle. The REs were tasked with micro-patching the preprocessing binary and calling the ”sharpen” function to enhance the image, enabling the rover to navigate past the rock.

To gain insights into the binary, the REs leveraged AGNOMIN's function matching and function name prediction capabilities. First, they used a reference binary with source code to identify functions within the target binary that appeared to be related to image processing. Then, using the matched and predicted function names, they could pinpoint the specific function requiring patching to enable the ”sharpen” functionality.

AGNOMIN's ability to match functions across binaries and provide descriptive function name predictions played a crucial role in assisting the REs in understanding the binary's functionality and locating the critical code sections that needed modification. This case study highlights the practical utility of AGNOMIN in real-world reverse engineering scenarios, where efficient and accurate analysis of stripped binaries is essential for addressing critical challenges.

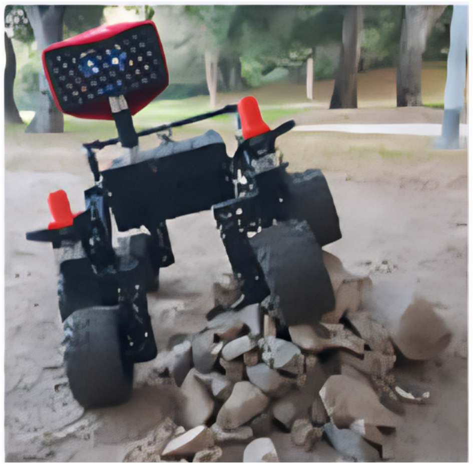
Fig. 2. Rover trapped in a rock.