CARIAD

Master Thesis: AI Usage in CI/CD/CT Pipelines for Compute Platforms in Automotives

We transform automotive mobility

CARIAD
A VOLKSWAGEN GROUP COMPANY

Agenda

// Previous Developments

// Status

// Method

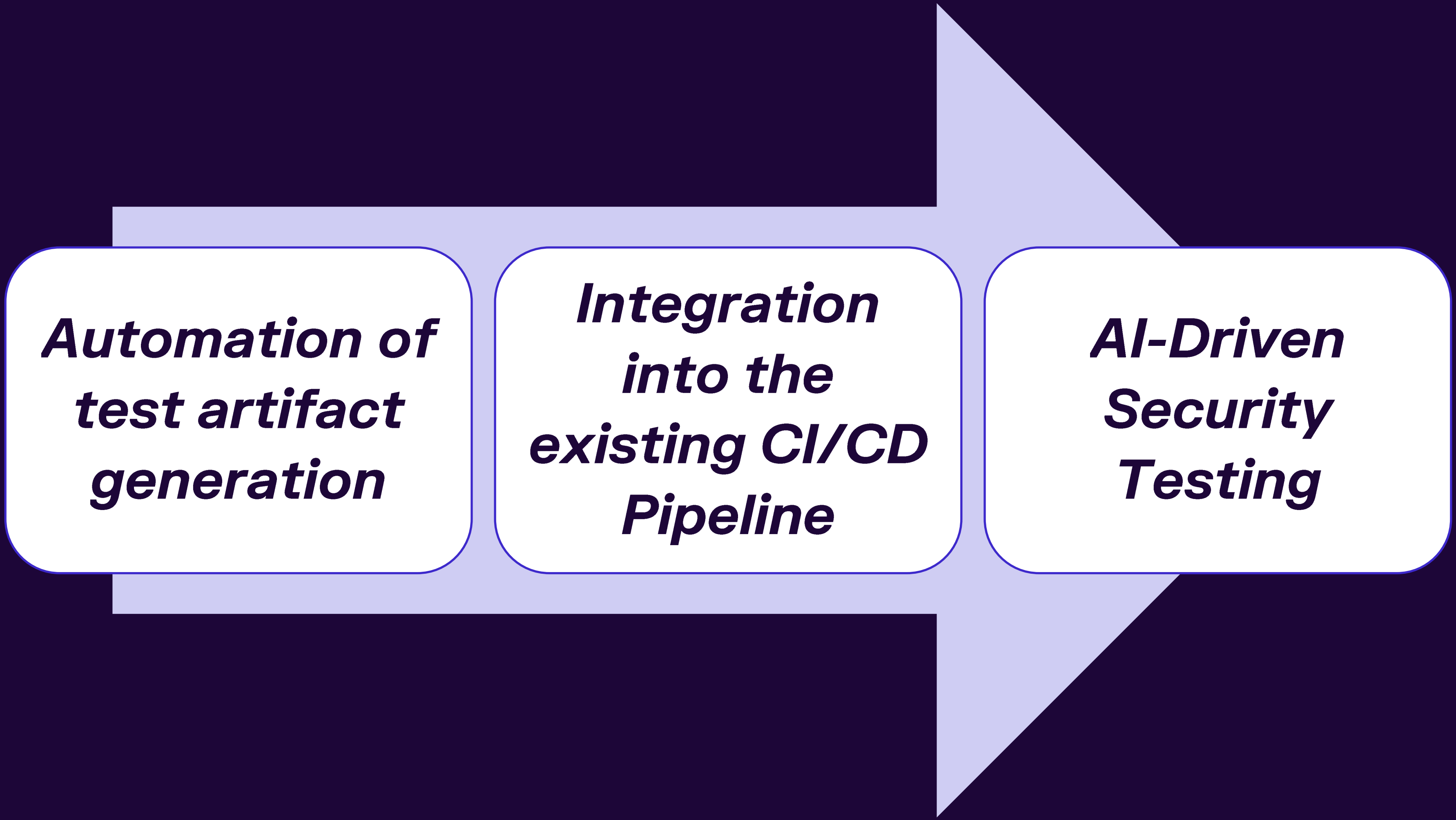
// Results

// Literature Review

// Next Steps

Previous Developments

Expected Outcomes

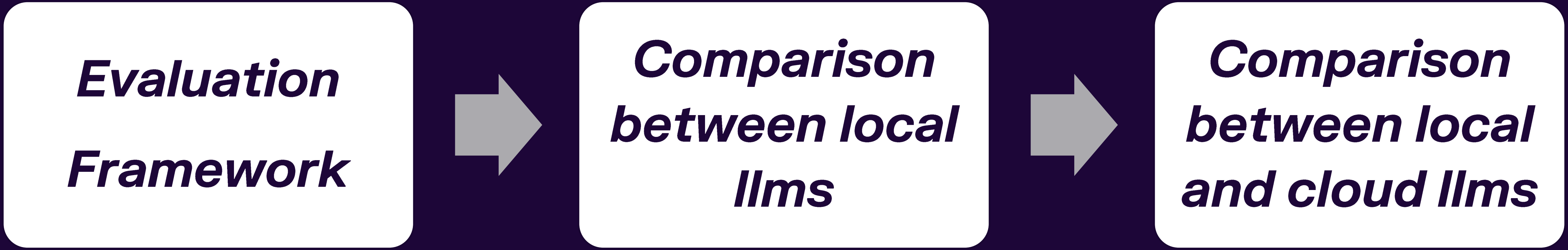


Automation of test artifact generation

Integration into the existing CI/CD Pipeline

AI-Driven Security Testing

Previously



Evaluation Framework

Comparison between local llms

Comparison between local and cloud llms

Evaluation Framework

Metrics:

- *Number of bugs found*
- *Code coverage percentage*
- *Time to generate fuzz tests*
- *Number of tokens*
- *Quality of generated code (measured via static analysis or manual review)*

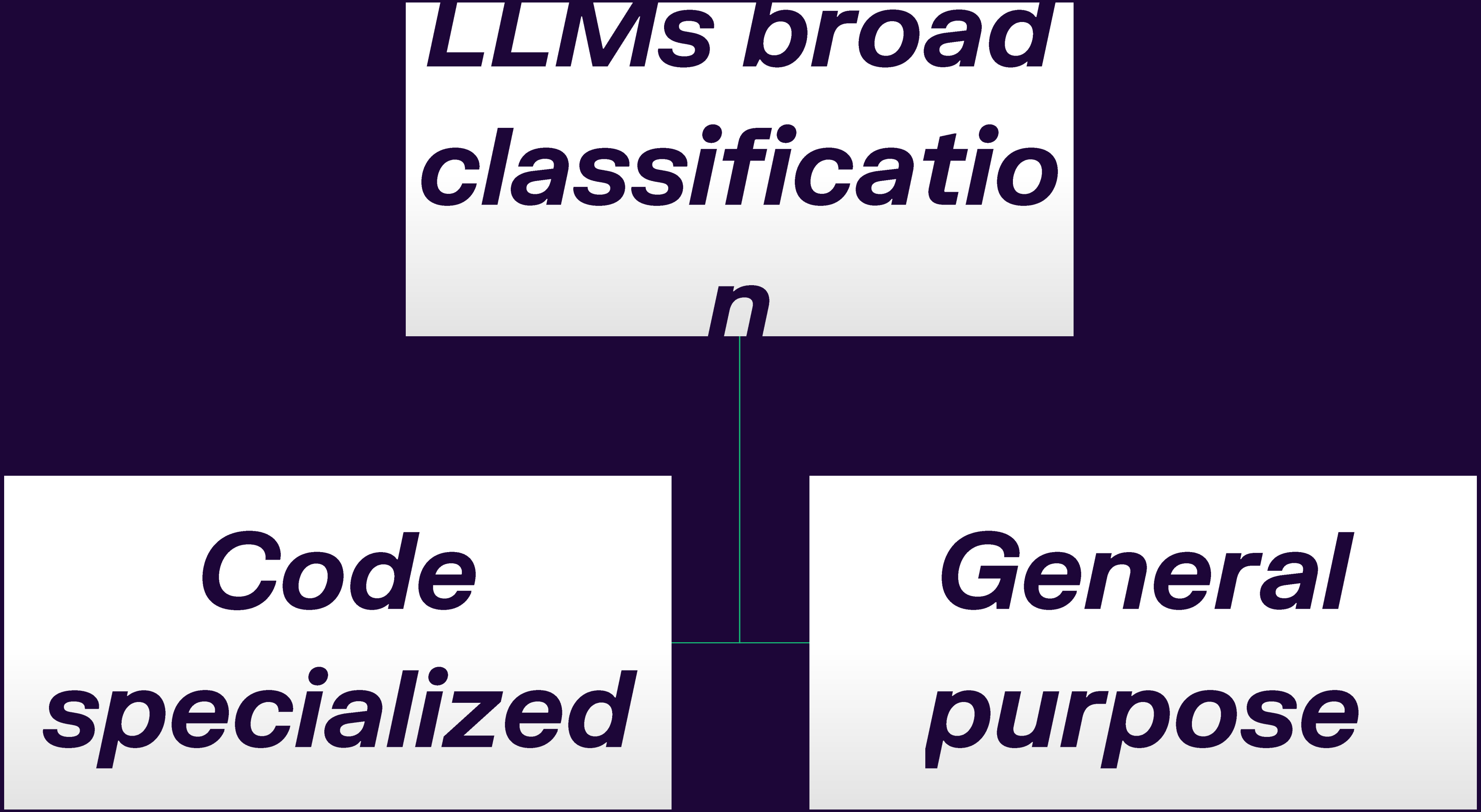
Status

Status

- Different local setups
- Local llms comparison (14 llms)
- Tested on various cpp repositories and libraries (25)
- Automated workflow – half success

Method

Method

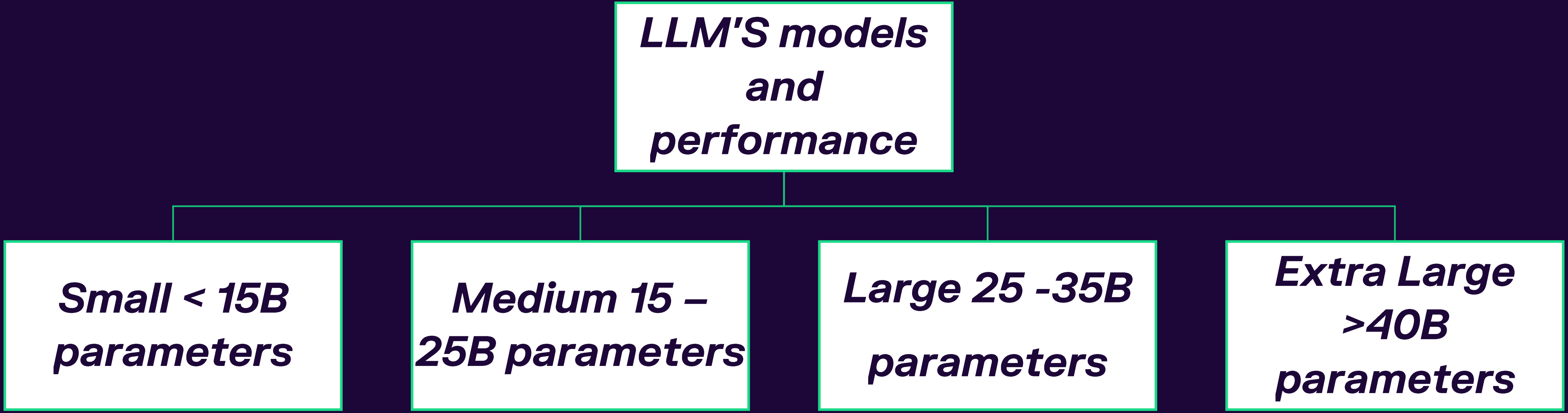


LLMs broad classification

Code specialized

General purpose

Method



LLM'S models and performance

Small < 15B parameters

Medium 15 – 25B parameters

Large 25 -35B parameters

Extra Large >40B parameters

Models

Large (25-35B parameters)

- gemma3:27b (27.4B)
- qwen3:32b (32.8B)
- deepseek-r1:32b (32.8B)
- qwen2.5-coder:32b (32.8B)
- deepseek-coder:33b (33B)
- wizardcoder:33b (33B)
- codellama:34b-instruct (34B)
- yi:34b (34B)

Models

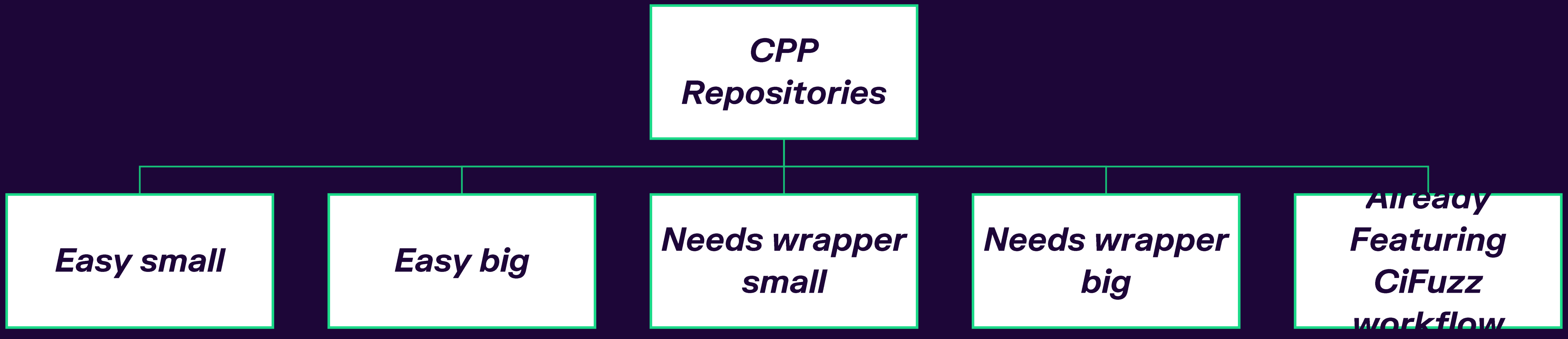
Extra Large (> 40B parameters)

- mixtral:latest (46.7B)

Previously Iterations

- qwen2.5-coder:7b (7B)
- qwen2.5-coder:14b (14B)
- qwen1.5:7b (7B)

Repositories



CPP Repositories

Easy small

Easy big

Needs wrapper small

Needs wrapper big

Already Featuring CiFuzz workflow

repositories

Results

Models - Successful llms

Code intelligence advanced setup

Models	Code Coverage	Time Taken	No of tokens used	Unique test cases	Successfull fuzz tests
Phi 14b	100%	12m 8s	59.8k	0	0
Llama 3	100%	3m 41s	27.6k	0	0
Qwen 1.5 7b	89.47%	6m 9s	50k	0	0

Models - Successful llms

Yaml cpp (35 files, 1061 candidates)

Models	Code Coverage	Time Taken	No of tokens used	Unique test cases	Successfull fuzz tests
Qwen 2.5 coder 32b					
Gemma 3 27b	45.06%	33m 33s	40.2k	2.05k	2
Phi 14b	34.26%	36m 36s	71.5k	2.22k	1

Models - Unsuccessful llms

Yaml cpp (35 files, 1061 candidates)

Models	Code Coverage	Time Taken	No of tokens used	Unique test cases	Successfull fuzz tests
Codellama 32b	0.00%	50m 43s	74k	0	0
Deepseek r1	0.00%	1h 36m 53s	54.9k	0	0
Deepseek code	0.00%	1h 37m 39s	48.8k	0	0
devstral	0.00%	38m 57s	82.6k	0	0
Llama 3 7b	0.00%	27m 40s	268k	0	0
Starcoder 2 15	0.00%	19m 55s	114k	0	0
Wizardcoder	0.00%	32m 15s	32.5k	0	0
Yi 34b	0.00%	2h 39m 53s	74.9k	0	0
Magistral 24b	0.00%	-	-	0	0
Mixtral	0.00%	-	-	0	0

Models - UnSuccessful llms

fmt

Models	Code Coverage	Time Taken	No of tokens used	Unique test cases	Successfull fuzz tests
Qwen 2.5 coder 32b	0.00%	43m 20s	59.8k	0	0
Gemma 3 27b	0.00%	41m 59s	73.9k	0	0

Models - Successful llms

pugixml

Models	Code Coverage	Time Taken	No of tokens used	Unique test cases	Successfull fuzz tests
Qwen 2.5 coder 32b	34.77%	37m 24s	43.1k	2.5k	1
Gemma 3 27b	0.00%	1h 12m 8s	122k	0	0

Models - Successful llms

Qwen 2.5 coder 32b

Repositories	Code Coverage	Time Taken	No of tokens used	Unique test cases	Successfull fuzz tests
jsoncons	45.64%	37m 24s	43.1k	2.5k	1
glm	0.00%	42m 55s	56.8k	0	0
spdlog	0.00%	49m 33s	62.9k	0	0

Literature Review

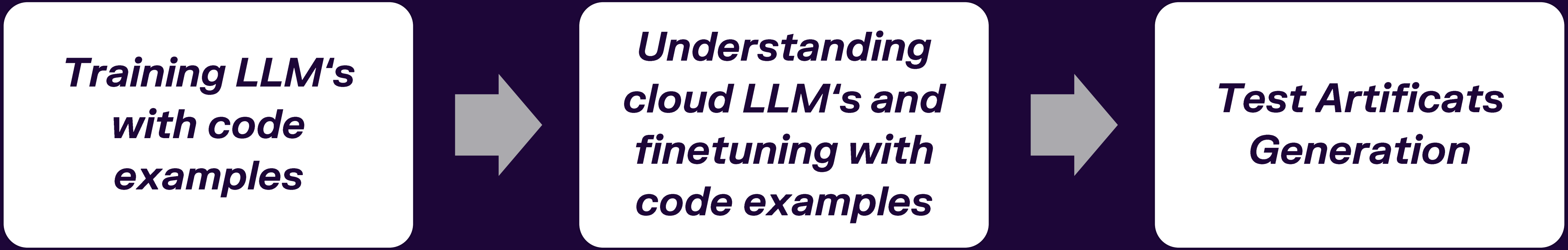
1. ***How effective are they? Exploring large language model based fuzz driver generation***

2. ***On the Challenges of Fuzzing Techniques via Large Language Models***

Literature Review

Next Steps

Next Steps



Training LLM‘s with code examples

Understanding cloud LLM‘s and finetuning with code examples

Test Artificats Generation

Finetuning local llms

Qwen 2.5 coder 32b instruct full model 60gb

LoRA (Low-Rank Adaptation)

- LoRA rank (r): 16 – controls adaptation size
- LoRA alpha: 32 – scaling for adaptation
- Dropout: 0.1 – prevents overfitting
- Target modules: q_proj, v_proj, etc. – efficient fine-tuning
- Device: auto – runs on best available hardware
- Dtype: float16 – faster, less memory
- Efficient model loading and saving (safetensors)

Code examples for training

Jsonl format

{"prompt": "Generate a comprehensive parser fuzzer for structured data\n\nRequirements:\n- Maximize code coverage and edge case detection\n- Generate intelligent, targeted inputs\n- Optimize for vulnerability discovery speed\n- Include comprehensive error handling„

Code examples for training

Qwen 2.5 coder 32b

```
"completion":
"void train_timing_attack(const uint8_t* data, size_t size) {\n    if (size < 20)
return;\n    \n    const char* secret = \"supersecretpassword123\";\n    std::string
user_input(reinterpret_cast<const char*>(data), \n                            std::min(size,
strlen(secret)));\n    \n    auto start =
std::chrono::high_resolution_clock::now();\n    \n    // VULNERABLE: Early return creates
timing difference\n    bool matches = true;\n    for (size_t i = 0; i < strlen(secret) && i <
user_input.length(); i++) {\n        if (secret[i] != user_input[i]) {\n            matches =
false;\n            break;  // Early return reveals timing information\n        }\n        // Simulate
processing time\n    }\n    \n    auto end =
std::chrono::high_resolution_clock::now();\n    auto duration =
std::chrono::duration_cast<std::chrono::microseconds>(end -
start);\n    \n    printf(\"Password check took %ld microseconds, match: %s\\n\",
\n           duration.count(), matches ? \"true\" : \"false\");\n}"}
```

Thank you!

We transform automotive mobility

CARIAD

A VOLKSWAGEN GROUP COMPANY

Any Questions