Network Management Automation Using Large Language Models

Illia Nasiri, Gwen Yang, Kangyi Zhang
Simon Fraser University
Supervised by: Dr. Ouldooz Baghban Karimi

Motivation

- Network management has become increasingly complex.
- Intent-Based Networking (IBN) and network programmability (P4) provide a more flexible, high-level approach to network control.
- Large Language Models (LLMs) make it easier to translate natural-language instructions into programmable network behavior [1,2].

Problem

P4 (language for programming the network data plane) is underrepresented in LLM training data, and natural-language intents are highly variable, leading to poor P4 code generation [3].

Research Question:
How can an AI system help automate the generation of complex network programs from a high-level, natural-language intent?

Methodology

Experimental Setup

Base model: DeepSeek-Coder-V2 Instruct 16B [3]
Dataset size: 400+ P4 code samples
Prompting: Zero-Shot · Few-Shot · Chain-of-Thought
Metrics: Compilation rate (p4c) · pass@k (p4c + P4Testgen)
Compared: Qwen 3 Max · Llama 4 Maverick · GPT-5 Mini

Compilation Rate

Does the generated P4 code compile?

Validated using the p4c reference compiler.

Necessary but not sufficient for functional correctness.

pass@k

Is at least 1 of k generated programs correct?

Generate n=100 programs per prompt.

Validated with **p4c** + **P4Testgen**.
Report pass@1 and pass@10.

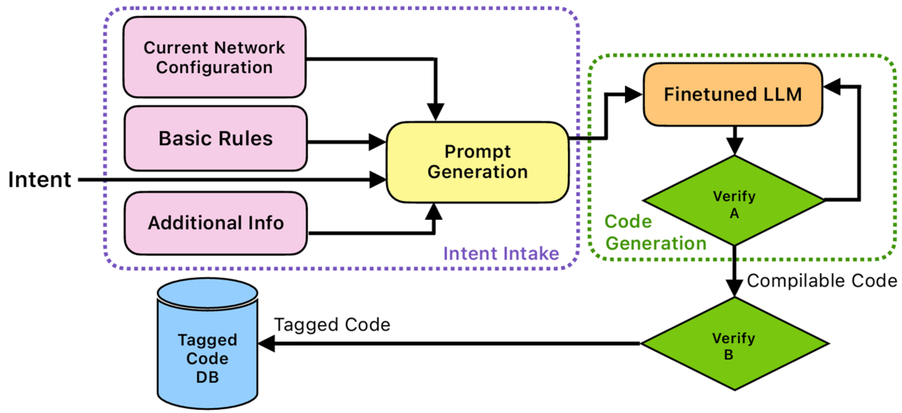
Evaluation Pipeline

GENERATE Code (n=100 programs) → COMPILE (p4c compiler) → Generate TESTS (P4Testgen) → PASS@k (k = 1 or 10)

A sample **passes if it compiles (p4c) and passes all functional tests (P4Testgen).**

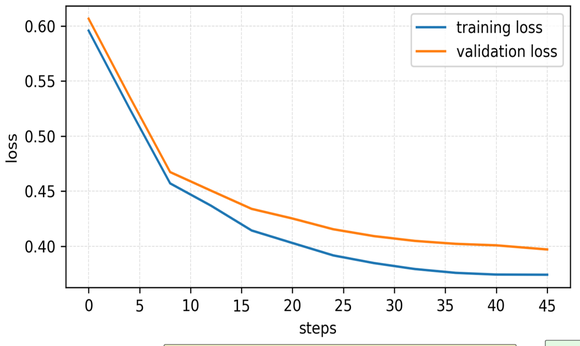
Design & Implementation

System:



Fine-tuning:

- **QLoRA** fine-tuning on **P4** examples
- Trained only **LoRA adapters**; base model mostly frozen
- **Gradient accumulation** for memory efficiency
- **Warmup + cosine decay** for stability
- **Flash Attention 2** for speed **~40 min** training on one GPU



Parameter	Value
QLoRA Params	
r	16
α	32
Target Modules	q_proj, v_proj, k_proj, o_proj
Dropout	0.05
Fine-tuning Parameters	
Optimizer Steps	45
Learning Rate	min: 10^{-5}, max: 4.0×10^{-4}
Batch Size	131,072 tokens

* Cosine decay with 10 linear warm-up steps
** 4096 *tokens* × 32 *examples*

Intent
Implement a basic Ethernet switch with source MAC learning and destination MAC forwarding.

Prompt with Wrapper

```
<|begin_of_sentence|>You are a P4 code generator. Respond with one compilable P4_16 program wrapped in <p4>...</p4> tags...
Target BMv2 with v1model and ensure the output works with p4c.
User:
You are generating one compilable P4_16 program for BMv2 v1model.
TASK:
Implement the network intent: Implement a basic Ethernet switch with source MAC learning and destination MAC forwarding.
YANG MODEL SCHEMA:
The following YANG model defines the data structure and constraints:
```yang
<port-number, interface-name, rule-id, interfaces configuration,firewall rules>
```
CURRENT NETWORK CONFIGURATION:
```json
<interfaces, firewall>
```
EXAMPLES:
Here are examples of similar network intents and their P4 implementations:
Example 1:
Intent: Implement a basic stateful firewall.
Implementation:
```p4
<Full P4 code implementation>
```
Example 2:
Intent: Implement basic L3 forwarding with Explicit Congestion Notification (ECN).
Implementation:
```p4
<Full P4 code implementation>
```
REQUIREMENTS:
- Target: BMv2 with v1model architecture
- Use the YANG model schema to understand data structures
Generate the P4 program now:
```

Generated P4 Code

```
#include <core.p4>
#include <v1model.p4>
...
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
    action drop() {
        mark_to_drop(standard_metadata);
    }
    action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
        standard_metadata.egress_spec = port;
        hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
        hdr.ethernet.dstAddr = dstAddr;
        hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
    }
    table mac_learning {
        key = {
            hdr.ethernet.dstAddr: exact;
        }
        actions = {
            ipv4_forward;
            drop;
        }
        size = 1024;
        default_action = drop();
    }
...
    apply {
        if (hdr.ipv4.isValid()) {
            mac_table.apply();
        } else if (hdr.ethernet.isValid()) {
            mac_learning.apply();
        }
    }
}
...
V1Switch(
MyParser(),
MyVerifyChecksum(),
MyIngress(),
MyEgress(),
MyComputeChecksum(),
MyDeparser()
) main;
```

Results

98% pass@10 (Fine-Tuned)	**88%** pass@1 (Fine-Tuned)
76% Compilation Rate (FS)	**2.1×** vs Standard LLMs

Fine-Tuning

- Fine-tuning on 400+ P4 samples improves compilation from 30% → 76% and Pass@1 from 40% → 88%.
- Fine-tuned 16B model outperforms Qwen 3 Max, Llama 4 Maverick, and GPT-5 Mini on pass@1.

Prompting Strategies

- Zero-shot fails completely (0% compilation for base model) — LLMs need domain examples for P4
- Few-shot prompting boosts compilation across all models — up to +40 pp over zero-shot.
- Chain-of-Thought helps (15% → 60% for fine-tuned) but few-shot remains strongest overall.

Intent Classification

- TF-IDF + SVM classifier retrieves top-2 category examples via few-shot retrieval, improving code generation by 10% over random selection.

Conclusion & Future Work

LLMs can support intent-driven network management automation:

We achieved up to 98% compilation success and 48-100% pass@k success.

Future work:

- Improve generation & verification quality
- Add runtime verification and feedback
- Test on broader network scenarios and policies
- Expand classifier-guided retrieval to support validation and automated retraining

[1] Zhiyuan He, Aashish Gottipati, Lili Qiu, Xufang Luo, Kenuo Xu, Yuqing Yang, and Francis Y. Yan. Designing network algorithms via large language models. In Proceedings of the 23rd ACM Workshop on Hot Topics in Networks, HotNets '24, page 205–212, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400712722. URL https://doi.org/10.1145/3696348.3696868.
[2] Oscar G. Lira, Oscar M. Caicedo, and Nelson L. S. da Fonseca. Large language models for zero touch network configuration management. IEEE Communications Magazine, 63(7):146–153, 2025. doi: 10.1109/MCOM.001.2400368.
[3] Mohammad Riftadi and Fernando Kuipers. P4i/o: Intent-based networking with p4. In 2019 IEEE Conference on Network Softwarization, pages 438–443, 2019.
[4] Zhu, Q., et al. (2024). DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence. arXiv:2406.11931. https://arxiv.org/abs/2406.11931