Supervised Fine-Tuning Report

Aniket Chaudhry, Edward Ibarra

October 7, 2025

Data Preprocessing

For the supervised fine-tuning (SFT) phase, we structured the training data into a unified prompt-completion format. Each example from the `mbpp-rkt-correct-executions` dataset was combined into a single text block. This approach combines the problem's description, input format, and output format as a multi-line comment, followed directly by the complete Racket code solution.

This structure was chosen to represent a developer's workflow. By training the model on our format, it learns to associate the prompt with the complete code solution. The final format for each training example is as follows:

```
; <Problem Description>
; Input format: <Input Format Description>
; Output format: <Output Format Description>

<Full Racket Code Solution>
```

Example

```
; Write a function to count the number of vowels in a string...
; Input format: The input consists of two lines...
; Output format: The output is a single integer...

#lang racket

;; Function to count vowels in a string based on a given set of vowels
(define (count-vowels input-string vowels-string)
  (define vowels (string->list vowels-string))
  (define (is-vowel? char)
    (member char vowels))

  (define (count-char char)
    (if (is-vowel? char) 1 0))

  (define (count-vowels-in-string str)
    (foldl (lambda (char count) (+ count (count-char char))) 0 (string->list str)))

  (count-vowels-in-string input-string))

;; Read input from standard input
(define input-string (string-downcase (read-line)))
(define vowels-string (string-downcase (read-line)))

;; Call the function and print the result
(display (count-vowels input-string vowels-string))
```

Experiments and Analysis

We conducted two fine-tuning experiments on the base `Qwen/Qwen3-1.7B-Base` model to observe the impact of different parameter configurations. Both models were trained using the AdamW optimizer.

Hyperparameter Configuration

Model 1 was configured for a brief, single-epoch run with a higher, constant learning rate. Model 2 was designed for a more extensive training process over three epochs, utilizing a lower learning rate and a linear scheduler.

Table 1: Hyperparameter Comparison

Parameter	Model 1	Model 2
Epochs	1	3
Learning Rate	Constant 5×10^{-5}	2×10^{-5}
LR Scheduler	None	Linear with warmup
Warmup Steps	N/A	50
Batch Size	4	4

Evaluation Results

The scheduler for Model 2 gradually warms up the learning rate to its maximum value over the first 50 steps and then linearly decays it to zero by the end of training. This more conservative approach resulted in slightly better performance on the test set.

Table 2: Pass@1 Evaluation Results

Metric	Model 1	Model 2
Pass@1 Rate	38.00%	40.00%
Problems Passed	19 / 50	20 / 50

Analysis

The most effective parameter in our experiments was the learning rate strategy. From our charts, we observed that Model 1, which used a constant learning rate of 5×10^{-5}, reached a low final training loss of approximately 0.03 after just one epoch. In contrast, Model 2 utilized a more complex schedule with a lower learning rate of 2×10^{-5} and required three epochs to achieve a comparable training loss.

This suggests that the higher, simpler learning rate was more efficient for this task, allowing the model to converge much more quickly. However, the more conservative learning rate and scheduler of Model 2 was ultimately the more effective strategy, yielding a higher Pass@1 rate.

Performance Discrepancy

The key discrepancy is the gap between the near-perfect training loss and the 38-40% success rate on the test set. This is an indicator of overfitting. While the models became incredibly familiar with the training dataset, that expertise did not fully translate to solving the problems in the testing dataset. The models learned the specific patterns and examples in the training set so well that it slightly limited their flexibility when faced with the new challenges of the test set.

Appendix

Links

- Project Link: W&B Project
- Report Link: W&B Report

Metrics

Figure 1: Training Loss

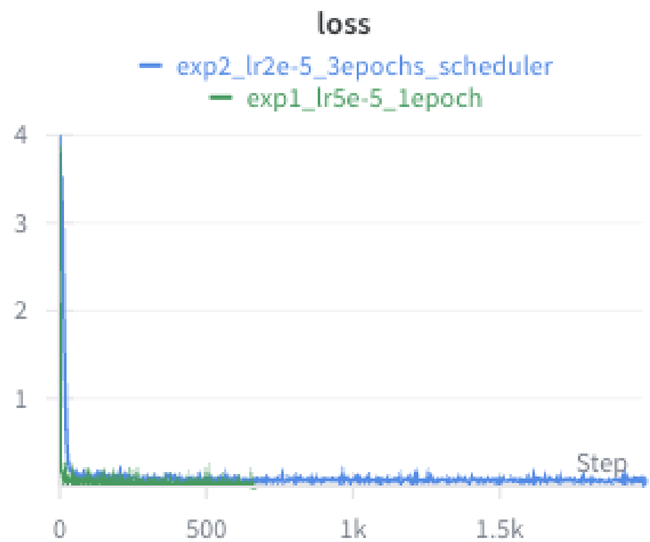

Figure 2: Learning Rate

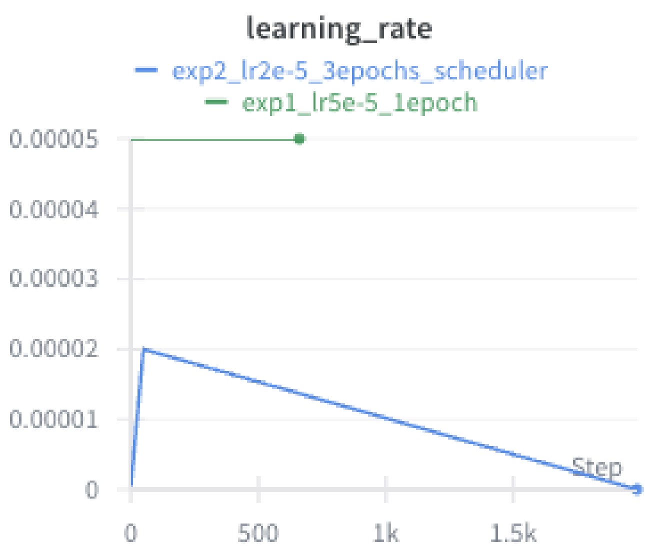

Figure 3: Training Loss

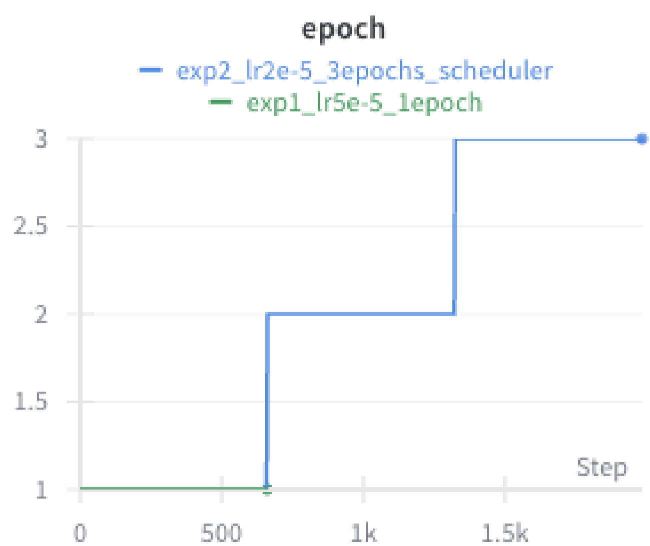

Figure 4: Learning Rate