# Progress Report: EPFLLaMA - A Lightweight LLM Finetuned on EPFL Curriculum

EPFLLaMA

Ali Bakly | 383057 | ali.bakly@epfl.ch
Elias Ulf Hörnberg | 384928 | elias.hornberg@epfl.ch
Othmane Sqalli Houssaini | 246132 | othmane.sqallihoussaini@epfl.ch
AB-EH-ME

## 1 Introduction

This progress report presents the work conducted on fine-tuning and evaluating a language model for specific tasks using various datasets. Our approach involves two key methodologies: Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO). We describe the construction and processing of three datasets (SFT, DPO, and MCQA), the model utilized, and the preliminary training results. Additionally, we explore the quantization specialization to enhance model efficiency.

## 2 Dataset

Our finetuning pipeline includes Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) regimes, using three main datasets: SFT, DPO, and MCQA.

**Supervised Fine-Tuning (SFT):** The SFT dataset is derived from the H4 Stack Exchange Preferences Dataset (Lambert et al., 2023), focusing on Data Science, Computer Science, Physics and Mathematics forums. Downloaded from Huggingface, the data was processed, and questions with their best answer scoring $\geq 4$ (around 10 upvotes) were selected. If the questions did not have an answer with that score, it was rejected. We cleaned the data using Python's BeautifulSoup and applied the Zephyr-7B-$\beta$ chat template (Tunstall et al., 2023) for training. Data points exceeding 2000 tokens, after concatenation with the chat template, were removed. This resulted in approximately 11,000 datapoints, split 90% for training and 10% for validation.

**Direct Preference Optimization (DPO):** The DPO dataset includes student-annotated preference data and Stack Exchange data. Non-alphanumeric chosen or rejected answers and MCQA questions were removed. Best and worst answers from Stack Exchange with scores $\geq 2$ were used. Data points exceeding 2000 tokens, including the chat template, were discarded. This yielded around 30,000 Stack Exchange preference points and 11,000 student-annotated preferences. We sampled 10,000 from each, resulting in 20,000 datapoints, with 18,000 for training and 2,000 for testing.

**MCQA:** For the MCQA dataset, the original student-annotated data was modified so the chosen output is a correct one-letter option. ChatGPT was used to map chosen answers to one letter. The most frequently occurring correct option per question was selected. This resulted in approximately 14,000 data points, plus a deduplicated dataset of 800 correct answers, possibly for SFT training.

## 3 Model

The base model used for fine-tuning is the `TinyLlama-1.1B-intermediate-step-1431k-3T` (Zhang et al., 2024). TinyLlama is a variant of the LLaMA (Large Language Model Meta AI) architecture, designed to be more efficient in terms of computational resources while maintaining competitive performance. This particular model has 1.1 billion parameters, making it a smaller yet capable variant suitable for various NLP tasks.

The TinyLlama model was pre-trained on diverse datasets, including SlimPajama (Soboleva et al., 2023) and StarcoderData (Li et al., 2023), to ensure broad language understanding and generation capabilities.

One should also mention that TinyLlama offers an already finetuned version, `TinyLlama-1.1B-Chat-v1.0`, but we choose to not use an already finetuned model in order to see how good results we can reach with our finetuning only. Also, TinyLlama does not come with a pad token, so we added it ourselves and accounted for it in the embedding dimensions

### 3.1 DPO Loss

In the Direct Preference Optimization (DPO) phase (Rafailov et al., 2023), a preference-based loss function was used to fine-tune the model based on human preferences. We use a variant of the DPO loss function called cDPO (Mitchell, 2023), which adjusts for noisy preference labels by assuming a probability of noise $\epsilon$ such that $0 < \epsilon < 0.5$:

$$\mathscr{L}^{\epsilon}_{\text{DPO}}(\theta, y_w, y_l) = (1-\epsilon) \cdot \mathscr{L}_{\text{DPO}}(\theta, y_w, y_l) + \epsilon \cdot \mathscr{L}_{\text{DPO}}(\theta, y_l, y_w) \quad (1)$$

See appendix D for details on $\mathscr{L}_{\text{DPO}}$.

## 4 Preliminary training results

For training, we used the Unsloth ecosystem and LoRA, targeting all projection modules. Although Unsloth supports 4-bit precision, we loaded TinyLlama in its original BF16 since we will quantize it later. Due to budget and time constraints (using Colab), extensive hyperparameter tuning was not possible.

### 4.1 Experimental Setup

**Supervised Fine-Tuning (SFT):** For SFT, we utilized the `SFTTrainer` class from the `trl` library. The training process employed the previously mentioned base model, tokenizer, and dataset, with a cross-entropy loss function. Here we used a LoRA attention dimension of $r = 32$ and a LoRA scaling parameter of $\alpha = 32$.

**Direct Preference Optimization (DPO):** For DPO, we observed that the data contained many incorrectly labeled preferences, resulting in considerable noise. To address this, we employed cDPO, as described in 1, with a label smoothing parameter $\epsilon = 0.2$. We used a learning rate of $5 \cdot 10^{-7}$, a cosine learning rate scheduler, and a warmup ratio of 0.1. The training was conducted with a batch size of 4, and spanned 2 epochs. Here we used a LoRA attention dimension of $r = 64$ and a LoRA scaling parameter of $\alpha = 64$.

### 4.2 Training and Evaluation Performance

For now, our evaluation metrics have been loss and reward accuracies. With reward accuracies we mean how often the chosen rewards are grater than than the corresponding rejected rewards. Note that our validation data is essentially the same as our test data, since little to none parameter tuning was performed.

**Supervised Fine-Tuning (SFT):** During the SFT phase, the training loss showed a general downward trend with occasional fluctuations, as illustrated in Figure 1. The validation loss similarly decreased over time, indicating effective learning by the model without much overfitting (see Figure 2).

**Direct Preference Optimization (DPO):** In the DPO phase, the training loss steadily declined with minor fluctuations, (refer to Figure 3). The validation loss also showed a consistent decrease, reflecting an improved generalization (Figure 5). Moreover, the validation reward accuracy kept improving over the steps, suggesting that the model's preferences were increasingly aligning with the desired outcomes (Figure 6).

You can refer to the appendix for the detailed figures on training and validation performance.

### 4.3 Hyperparameter Impact

- Learning Rate: For DPO, we used a learning rate of $5 \times 10^{-7}$, which struck a balance between stable training and effective preference alignment.
- $\beta$ Parameter: For DPO, we experimented with the $\beta$ parameter. We primarily used $\beta = 0.1$, but also tried $\beta = 0.4$. The change in $\beta$ did not significantly affect the model's performance, indicating a potential insensitivity to this hyperparameter within the tested range.

### 4.4 Interesting Observations

Despite various hyperparameter adjustments and different amounts of training data, the validation reward accuracy for the DPO dataset did not exceed around 67%, as seen in figure 6. This indicates potential limitations in the dataset quality that need further investigation.

## 5 Quantization Specialization

The quantization will be applied directly to the TinyLlama model that we have fine-tuned. We do not plan to switch to another architecture for the quantization process. For this purpose we are planning to utilize the `Bitsandbytes` library

### 5.1 Evaluation and data

We will use the same data to evaluate both the original 16-bit model and the quantized 8-bit model, ensuring a consistent basis for comparison and allowing us to directly measure the impact of quantization on model performance. This includes measuring the reward accuracy, assessing improvements in computational efficiency and speed, and comparing the reduction in memory footprint between the two models.

## References

Nathan Lambert, Lewis Tunstall, Nazneen Rajani, and Tristan Thrush. 2023. Huggingface h4 stack exchange preference dataset.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you!

Eric Mitchell. 2023. A note on dpo with noisy preferences & relationship to ipo.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplica

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model.

## A Chat Template

**Chat Template**

<|system|>
You are an experienced teacher who answers the STEM-related question asked by a student below.</s>
<|user|>
What is the time complexity of merge sort?</s>
<|assistant|>
...

## B DPO Loss

Below, you find the standard DPO loss function (Mitchell, 2023):

$$\mathscr{L}_{\text{DPO}}(\theta, y_w, y_l) = -\log \sigma \left( \beta \, h_{\pi_\theta}^{y_w, y_l} \right)$$
$$h_{\pi_\theta}^{y_w, y_l} = \log \frac{\pi_\theta(y_w)}{\pi_{\text{ref}}(y_w)} - \log \frac{\pi_\theta(y_l)}{\pi_{\text{ref}}(y_l)}$$

In this equation, $\theta$ are the weights of the policy model being optimized, and variables $y_w$ and $y_l$ denote the preferred and less preferred responses, respectively. The parameter $\beta$ controls the strength of the KL-divergence constraint, and $\sigma$ denotes the logistic sigmoid function. $\pi_\theta$ is the policy model being optimized, and $\pi_{\text{ref}}$ is the reference model, which is often the same as the initial model before DPO.

## C SFT Training

Below you find the loss plots for the SFT training.

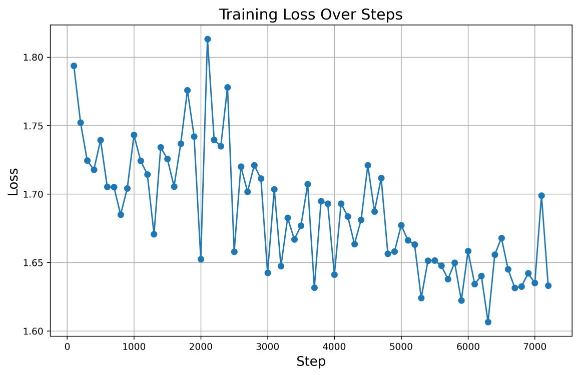

Figure 1: Training loss during SFT.

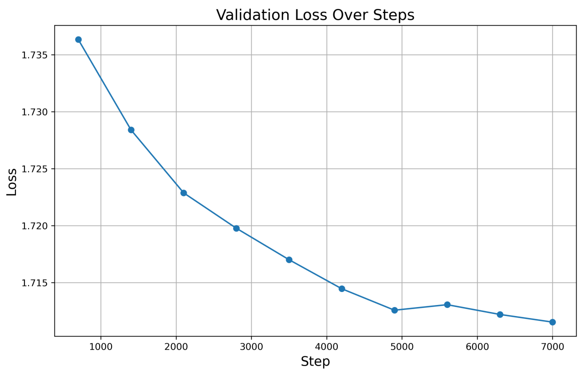

Figure 2: Validation loss during SFT.

## D DPO Training

Below you find the loss and reward accuracy plots for DPO training.

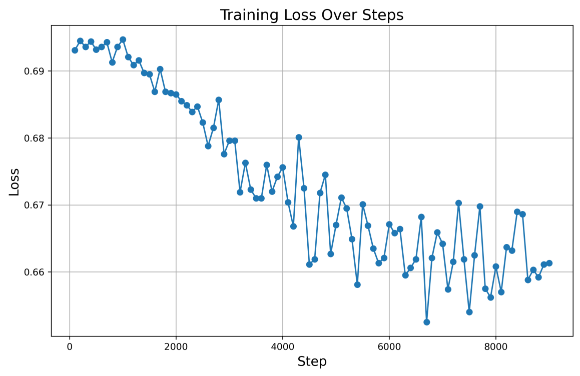

Figure 3: Training loss during DPO.

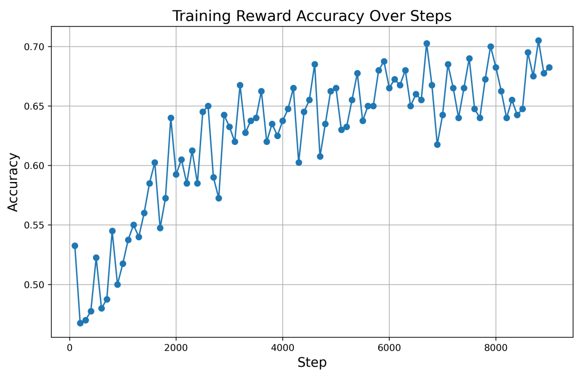

Figure 4: Training reward accuracy during DPO.

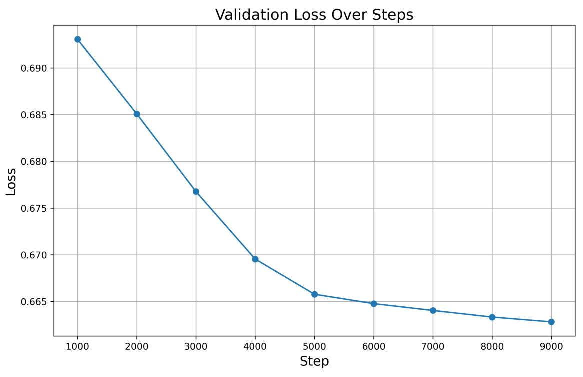

Figure 5: Validation loss during DPO.

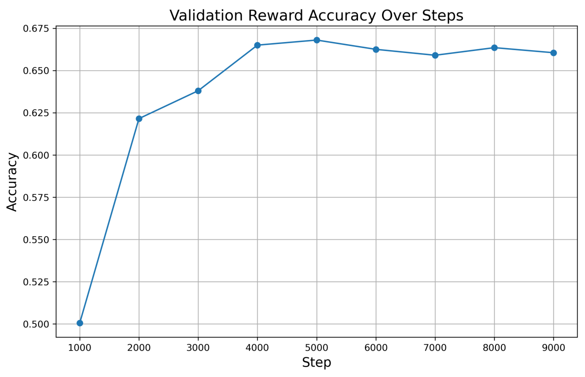

Figure 6: Validation reward accuracy during DPO.