Extended Abstract

Motivation Training time remains a significant expense for those creating AI models in both academia and industry. Costs for state-of-the-art GPUs rise to many thousands of dollars, hindering progress for those less resourced. While efforts to create more efficient or cheaper GPUs are underway (Etched, Cerebras), we are motivated to reduce training costs via algorithmic methods. Curriculum Learning (CL) stands out to us as a promising avenue; our goal is to use a novel method of Self-Paced CL to speed up model training.

Method For our project, we fine-tune a Qwen2.5-0.5B model (HuggingFace ID: `https://huggingface.co/Qwen/Qwen2.5-0.5B`) on the Countdown arithmetic task through Direct Preference Optimization (DPO) and Supervised Fine-Tuning (SFT), using a CL regime.

To give context on the premise behind CL: Curriculum Learning builds on the idea that models train more efficiently when given easier examples first before expanding to harder ones. In our novel method, we use initial loss as a difficulty heuristic; this is not commonly used, because models are usually randomly initialized, making loss a poor indicator of difficulty. However, in this situation we have a pre-trained model that already possesses some capability for the task, so initial loss serves as a good indicator of what the model finds difficult or easy.

Implementation We first fine-tune the Qwen2.5-0.5B model via SFT on the *cog_behav_all_strategies* dataset (HuggingFace ID: `Asap7772/cog_behav_all_strategies`) of language-based arithmetic challenges. The train and test examples are each formatted as a query plus a ground-truth correct arithmetic expression preceded by a `<think>` reasoning trace. We then further fine-tune the model via DPO on the *Countdown-Tasks-3to4* dataset (HuggingFace ID: `Jiayi-Pan/Countdown-Tasks-3to4`), creating the preference dataset through a rule-based reward function.

For both SFT and DPO training, we utilize the following CL regime: we first calculate the initial loss for every example in the dataset as a heuristic for difficulty, and while calculating these initial losses we do not update the model. We then split the dataset into n smaller mini-datasets, each containing one-n-th of the examples sorted by difficulty. We start training on the easiest mini-dataset and then append the next-easiest mini-dataset at each epoch.

We also experiment with Anti-Curriculum learning, which follows the same process but in reverse (i.e., training on the hardest examples first).

Results In terms of training time, we find that across both SFT and DPO, the CL regime reduces time to convergence by as much as around 20%; we are by and large able to achieve similar loss values in less time.

Looking more in depth, in our initial CL implementation with SFT alone, we reach similar loss values values for 2-4 epochs. However, the CL regime leads to worse loss values as the number of epochs grows past that. With the SFT + DPO implementation, we consistently achieve similar or better loss values under our CL regime.

Discussion Examining our results, it is clear that our CL regime speeds up training times. The higher loss values for larger number of epochs, however, appear to stem from our segmentation of the training dataset into mini-datasets; our SFT dataset is much smaller than the DPO dataset, and so when we split it into even smaller chunks for CL, we expose the model to less data, producing higher loss values initially. This may be causing under-fitting that is not fully recovered later. On the other hand, for our DPO implementation, the dataset is much larger, yielding larger mini-datasets. The model is therefore exposed to more data and is able to achieve lower loss values.

Conclusion To conclude, we successfully show that our CL implementation reduces time to convergence while being able to achieve similar loss values - provided that the mini-dataset segmentation is not too granular. Such findings matter because they allow more optimal use of training resources for less-funded academic and industrial labs. Future steps may include combining the loss heuristic with more rule-based heuristics or developing new dataset-segmentation methods that limit initial data exposure less.

Learning From Failure - Loss Based Curriculum Learning

Tomas Coghlan
Department of Computer Science
Stanford University
`tcoghlan@stanford.edu`

Ismail Mardin
Department of Computer Science
Stanford University
`ikmardin@stanford.edu`

Mattheus Wolff
Department of Computer Science
Stanford University
`mbwolff@stanford.edu`

Abstract

Reducing the computational cost of large-language-model fine-tuning is crucial as GPU prices soar. We explore a novel method of Self-Paced Curriculum Learning (CL), using a regime that feeds a model progressively harder examples based on their initial loss to accelerate training without new hardware. Leveraging the pretrained Qwen2.5-0.5 B model, we first apply supervised fine-tuning (SFT) on the `cog_behav_all_strategies` arithmetic dataset, then refine the model with Direct Preference Optimization (DPO) on `Countdown-Tasks-3to4`. For each dataset we rank examples by their forward-pass loss (no updates), partition them into n equal mini-datasets, and append the next-harder mini-dataset each epoch. An Anti-Curriculum (hard-to-easy) schedule is also tested. Across both SFT and DPO, CL cuts training time to convergence by up to around 20%. On the smaller-sized SFT corpus, overly fine binning (large n) degrades final loss because early epochs expose the model to too little data; 2-4 epochs (comprised of 2-4 mini-datasets respectively) balance runtime and accuracy. On the larger DPO corpus, CL consistently attains equal or lower loss than a non-curricular baseline while preserving the runtime gains. These results show that loss-based CL is an effective, hardware-agnostic lever for faster training, with performance trade-offs governed by dataset size and mini-dataset granularity. Future work should blend loss with rule-based difficulty signals and develop segmentation schemes that retain broader early coverage.

1 Introduction

In the last five years, the capabilities of large language models (LLMs) have grown at an extremely rapid pace, enabling near-human performance on an ever expanding set of reasoning and generation tasks. This progress, however, has come with a steep rise in computational cost: state-of-the-art fine-tuning runs can require 10^5 GPU-hours, as well as hardware budgets that are out of reach for most academic and early-stage company labs. As the marginal gains from ever-larger models begin to taper, finding algorithmic ways that lower the time to convergence is increasingly critical.

Curriculum Learning (CL) Bengio et al. (2009); Soviany et al. (2022) is one such lever. By presenting a learner with easier examples before harder ones - mimicking the natural progression of human instruction - CL can smooth the loss landscape and accelerate optimisation. Yet in practice, many mainstream language-model pipelines rarely exploit CL. The central hurdle is difficulty estimation:

prior NLP curricula rely on hand-crafted, rule-based heuristics (e.g., sentence length, word rarity), ancillary labels (e.g., problem ratings), or surrogate statistics such as expected loss Campos (2021); Kocmi and Bojar (2017). These proxies are often task-specific and can be brittle when transferred to new domains.

In this work we test the most direct signal available: the model's own per-example loss. Modern LLMs are commonly fine-tuned from a strong, pre-trained starting point; their forward-pass loss therefore provides a continuous, self-supervised estimate of how hard each example is – effectively the model's confidence in failure. We leverage this insight to construct a Loss-Based Curriculum Learning (LCL) regime that requires no auxiliary annotations. A single, annotation-free forward pass sorts the dataset by initial loss, after which we train in a self-paced manner that gradually appends higher-loss (harder) mini-datasets epoch by epoch.

We evaluate LCL on two complementary fine-tuning regimes: Supervised Fine-Tuning (SFT) and Direct Preference Optimisation (DPO) Rafailov et al. (2023). SFT uses the 0.5-billion-parameter Qwen2.5 model as a base, whereas DPO uses the best-trained SFT model. Target arithmetic reasoning tasks are drawn from cog_behav_all_strategies and Countdown-Tasks-3to4. The resulting model is then assessed on the Countdown arithmetic task Gandhi et al. (2024).

2 Related Work

Examples of successful early curriculum learning strategies were first laid out in Bengio et al (2009) [1], which defined difficulty in training examples using noise and margin to improve training in models. In our case, using loss can be seen as a comparable substitute for margin, as margin, like loss, reflected the magnitude of misclassification in this example. Soviany et al. (2022)[10] provides a survey of the current applications of Curriculum Learning, including the common methods of determining difficultly. Here we see that loss and forms of self-paced learning are rare with NLP tasks, making our attempt a novel approach; furthermore, when loss is used as a heuristic for difficulty of task, it is often indirectly calculated (e.g. via surrogate or expected-loss calculations).

Various efforts have applied Curriculum Learning in different domains: Shi et al (2025) [9] proposed the ADARFT method which dynamically adjusts problem difficulty during PPO-based finetuning, demonstrating convergence twice as fast on math tasks (but relying on pre-annotated difficulty labels). Campos (2021) [2] on the other hand attempted to fine-tune BERT, estimating difficulty of tasks with sentence entropy, finding no compelling evidence that it improved language model training. Another effort on language models by Kocmi and Bojar (2017) [5] examines bucketing by linguistic features for English–Czech neural machine translation (NMT). The features selected for (among others) include the length of the target sequence, number of conjunctions, frequency of commonly-used words. They managed to show that mini-batching based on increasing difficulty increases translation quality by up to 1 BLEU point.

Our proposed approach is significantly different from these past examples. Before training, we run a forward pass of the pretrained LLM on every example and use the actual per-example loss as a continuous difficulty score, rather than relying on hand-crafted proxies or surrogate estimates.

3 Method

Our approach consists of two main components: a two-stage fine-tuning pipeline and a novel curriculum learning regime. We first describe the fine-tuning methods, then detail our curriculum learning implementation.

3.1 Supervised Fine-Tuning (SFT)

We begin by fine-tuning the Qwen2.5-0.5B model on the `cog_behav_all_strategies` dataset Gandhi et al. (2025) using standard supervised learning. Each example consists of a query and a ground-truth completion that includes a reasoning trace preceded by a `<think>` token. The model is trained to predict the next token in the sequence, with the loss computed only on the completion portion (masking out the query). Specifically, for each example (x, y) where x is the query and y is the completion, we compute the negative log-likelihood loss:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{|y|} \log P(y_t|x, y_{<t}) \tag{1}$$

where y_t is the t-th token in the completion, $y_{<t}$ represents all previous tokens, and $P(y_t|x, y_{<t})$ is the model's predicted probability of token y_t given the query x and previous tokens $y_{<t}$. The loss $\mathcal{L}_{\text{SFT}}$ measures how well the model predicts each token in the completion sequence.

3.2 Direct Preference Optimization (DPO)

After SFT, we further refine the model using DPO on the `Countdown-Tasks-3to4` dataset. To create our preference dataset, we use our SFT model to generate pairs of completions for each prompt using different sampling parameters. Specifically, for each prompt in the dataset:

1. We generate two completions using different temperature settings:
    - A high-temperature completion (temperature=0.8) to encourage diverse but potentially less accurate solutions
    - A low-temperature completion (temperature=0.1) to encourage more focused and likely more accurate solutions
2. We evaluate both completions using a rule-based scoring function Gandhi et al. (2025) that checks if the solution:
    - Uses only the provided numbers
    - Uses each number exactly once
    - Uses only basic arithmetic operations (+, -, *, /)
    - Correctly reaches the target number
3. We label the completion with the higher score as the preferred completion and the other as the dispreferred completion
4. We only keep pairs where there is a difference in scores to ensure meaningful preferences

This process creates a dataset of (x, y_w, y_l) triples where x is the prompt, y_w is the preferred completion, and y_l is the dispreferred completion. We use this dataset to train our model using DPO, which optimizes a policy to maximize the probability of preferred completions while minimizing the probability of dispreferred ones. We use our SFT model as a frozen reference policy (π_{ref}) to prevent large policy deviations from the supervised learning baseline. The DPO loss is computed as:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_w,y_l)} \left[\log \sigma \left(\beta \left(r_\theta(x, y_w) - r_\theta(x, y_l)\right)\right)\right] \tag{2}$$

where β is a temperature parameter controlling the strength of the preference signal, and $r_\theta(x, y)$ is the implicit reward defined as:

$$r_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} \tag{3}$$

Here π_θ is our current policy and π_{ref} is the reference policy (SFT model). The reward function measures how much more likely our current policy is to generate a completion compared to the reference policy.

3.3 Loss-Based Curriculum Learning

Our curriculum learning approach uses the model's own loss as a difficulty metric. This is made possible by our use of a pre-trained model that already possesses some capability, making initial loss a meaningful indicator of example difficulty. The curriculum proceeds as follows:

1. Before training begins, we perform a single forward pass through the dataset to compute the initial loss for each example, without updating the model parameters.

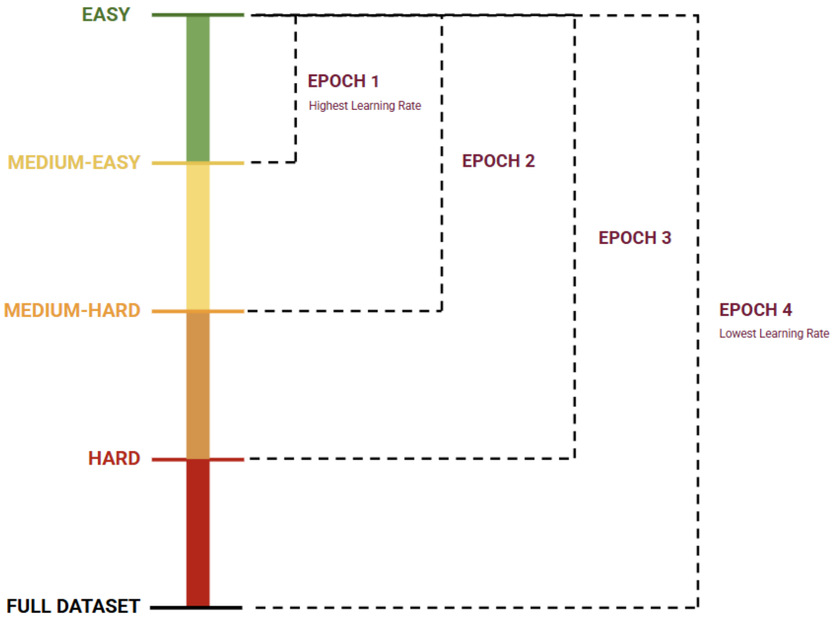

Figure 1: Curriculum Learning method

2. We sort all examples by their loss values, creating a natural ordering from easiest to hardest.
3. The dataset is partitioned into n equal-sized mini-datasets, where n is the number of training epochs.
4. During training, we start with only the easiest mini-dataset (lowest loss examples). At each epoch, we append the next-hardest mini-dataset to the training set.

This approach can be formalized as follows. Let D be the full dataset and $L(x, y)$ be the loss for example (x, y). For epoch $e \in \{0, \ldots, n-1\}$, the training set D_e is:

$$D_e = \{(x, y) \in D \mid L(x, y) \leq L_{(e+1)k}\} \tag{4}$$

where $L_{(e+1)k}$ is the loss value of the $(e+1)k$-th example when sorted by loss, and $k = |D|/n$ is the size of each mini-dataset.

We also experiment with an anti-curriculum approach that reverses this ordering, training on the hardest examples first.

Further, we investigate using a learning rate scheduler paired alongside our CL scheduling approach. A larger learning rate is used when training on easier examples and then decreases as we expand our mini-datasets to include harder examples. This allows the model to converge faster on easier examples where there are fewer complex abstractions that the model needs to capture.

The curriculum learning regime is applied independently to both SFT and DPO training. For SFT, we compute the initial losses using the negative log-likelihood on each example's completion. For DPO, we compute the initial losses using the DPO loss function, which captures both the model's ability to generate preferred completions and its deviation from the reference policy. This ensures that the curriculum reflects the actual learning objective during each stage of training.

4 Experimental Setup

We use the Qwen2.5-0.5B model as our base model, which is a 500M parameter causal language model. For supervised fine-tuning (SFT), we use the `cog_behav_all_strategies` dataset, and for DPO training, we use the `Countdown-Tasks-3to4` dataset.

We experiment with several training configurations:

4.0.1 SFT Training

Base learning rate: 1×10^{-5} to 1×10^{-6}, batch size: 4, number of epochs: 2-6 (varied across experiments), optimizer: AdamW Loshchilov and Hutter (2017) with linear learning rate scheduling.

4.0.2 DPO Training

Base learning rate: 1×10^{-6} to 1×10^{-7}, batch size: 2, number of epochs: 2-4, optimizer: RMSprop Ruder (2016), β parameter: 0.1 to 0.5 (controls policy deviation from reference).

4.1 Curriculum Learning Variations

We experiment with different curriculum learning strategies:

- **Standard Curriculum**: Examples ordered by increasing difficulty
- **Anti-Curriculum**: Examples ordered by decreasing difficulty
- **Learning Rate Scheduler**: Learning rate is gradually reduced throughout training.

All experiments use a fixed random seed (42) for reproducibility. We run experiments on CUDA-enabled GPUs, with model checkpoints saved after each epoch. The best model is selected based on the lowest validation loss achieved during training.

5 Results

Results from implementing different curriculum learning strategies on SFT training:

Epochs	Strategy	Val Loss	Wall-Time
2	No Curriculum	212.45	4.18 min
2	Curriculum	**211.37**	3.86 min
2	Anti-Curriculum	212.38	**3.76 min**
4	No Curriculum	214.11	8.26 min
4	Curriculum	**212.68**	**6.48 min**
4	Anti-Curriculum	214.58	6.73 min
6	No Curriculum	**214.04**	12.19 min
6	Curriculum	229.78	9.69 min
6	Anti-Curriculum	230.28	**9.43 min**

Table 1: Comparison of training strategies with SFT only (lr $= 1 \times 10^{-5}$). All hyperparameters aside from number of epochs were kept the same. Curriculum and Anti-Curriculum consistently deliver lower wall-times.

Results from implementing curriculum learning regime on DPO training:

Setting	Val Loss	Wall-time
No curriculum	0.2305	41.9 min
Curriculum	**0.2099**	**33.8 min**

Table 2: lr $= 1 \times 10^{-7}$, $\beta = 0.5$

Setting	Val Loss	Wall-time
No curriculum	**0.1735**	42.7 min
Curriculum	0.1783	**33.8 min**

Table 3: lr $= 1 \times 10^{-7}$, $\beta = 0.1$

Setting	Val Loss	Wall-time
No curriculum	0.2173	41.8 min
Curriculum	**0.1799**	**33.2 min**

Table 4: SFT + DPO, lr $= 5 \times 10^{-7}$, $\beta = 0.25$

5.1 Quantitative Evaluation

As we can see from Table 1, both Curriculum and Anti-Curriculum regimes were able to significantly reduce Wall-Time across all epochs, with the largest absolute reduction occurring when the number of epochs was also high. The largest percentage change was from 12.19 mins to 9.43 mins (a reduction of around 23%). Our best SFT-trained models occurred when lr $= 1 \times 10^{-5}$, achieving similar or even slightly better Val Loss at 2 and 4 epochs. However, at 6 epochs, the model performed worse with the CL regime than without; this can perhaps be explained by the fact that as the number of epochs grows, our dataset gets split into more mini-datasets for our CL regime. Thus, the model is exposed to much less data across epochs than without the CL regime, and thus a worse model is trained.

Similarly our CL regime was also able to improve Wall-time during DPO training, with the largest improvement seen in Table 2 with a decrease from 42.7 mins to 33.8 mins (a reduction of around 21%). Contrary to the above however, DPO training with our CL regime was able to consistently achieve comparable or better Val-Loss; this may be due to the larger DPO dataset, which would provide ample examples for the model to learn even when split into more mini-datasets. The best Val-Loss and Wall-time reduction were also achieved when lr $= 1 \times 10^{-7}$, and $\beta = 0.1$.

For our leaderboard submission, we achieved a **0.48** score for our SFT+DPO fine-tuned model without our CL regime. With our CL regime, this was imrpoved to **0.49**. This makes sense; CL does not inherently improve the quality of the model, but allows it to converge faster.

5.2 Qualitative Analysis

Qualitatively, we can examine our generated responses to the Countdown test prompts. It does fairly well, reaching the answer through valid reasoning most of the time. Common errors however include not formatting the eventual answer correctly within the <answer>...</answer> tags, leading it not to be extracted by the leaderboard script. Furthermore, our model seems to struggle more when there are more than 4 numbers required to reach the target asnwer. There is also heavy use of parentheses, even when they are not stricly necessary - this may be due to the high prevalance of parentheses in preferred responses relative to dispreferred responses in the DPO dataset.

An example generation from our model is below:

```
<think>
Let me try to find a way to reach 66.
First, let me look for numbers close to 66:
76 is too large, we need to reduce it.
73 is also too large.
49 is closest to 66, we need to add 17 to it.

Let me try working with 76 first:
76 - 34 = 42 (this gets us closer to 66)
42 + 73 = 115 (too large)

Let me try a different approach:
```

```
73 - 49 = 24
24 + 76 = 100 (too large)

What if I try:
73 - 49 = 24
24 + 76 = 100
100 - 34 = 66 (This works!)
</think>
<answer> (73 - 49) + 76 - 34 </answer>
```

6 Discussion

Our results demonstrate that loss-based curricula can reduce time to convergence for both supervised and preference-based fine-tuning without sacrificing — and in some settings even improving — the strength of the model.

Across all hyper-parameter sweeps, LCL shaved roughly 20% off training time. In practice this might translate to freeing an A100 GPU 2 hours sooner for every 10 hour job — which is non-trivial for resource-constrained academic and industrial groups. The gains stem from two compounding effects: early epochs process fewer and easier examples, shortening each optimization step; and the smoother loss landscape eases gradient noise, allowing larger effective steps before the learning rate decays.

On the small SFT corpus, very fine granularities ($n \geq 6$) did hurt final loss. Intuitively, restricting the first epoch to around 16% of the already small dataset starves the model of signal, causing under-fitting that is not fully recovered later. Conversely, the larger DPO corpus maintained or exceeded baseline accuracy for all n values; each mini-dataset still contained many diverse examples, preserving coverage.

7 Conclusion

We introduced loss-based curriculum learning (LCL), an annotation-free schedule that orders training examples by their initial forward-pass loss. On the Countdown arithmetic-reasoning benchmark, LCL cut time-to-convergence by around 20% and, on sufficiently large datasets, matched or improved final validation loss. The approach is simple to implement, task-agnostic, and complementary to both SFT and DPO pipelines.

Future work. Several extensions are could be explored:

1. **Multi-modal difficulty signals.** Combining loss with other rule-based heuristic features (length of sentence/words, complexity of phrases) may yield more robust curricula.
2. **Scaling studies.** Re-running LCL on 7 B-parameter models and language-rich tasks (GSM8K, HumanEval) will reinforce whether the gains persist at larger scales.
3. **Dense numeric shaping.** Replace binary verifier reward for the Countdown task with $r = -|f(x) - T|$, or other more continuous functions to provide smoother gradients and denser reward space.
4. **Token-Level Ensembling** Predictions at the individual token level are combined from multiple models. This approach aims to leverage the strengths of each model to produce a more robust and refined output.

8 Team Contributions

- **Tomas Coghlan:** Implementation/debugging of DPO, creation of DPO preferences dataset, and editing poster/final report.
- **Ismail Mardin:** Debugging of SFT and DPO, implementation of evaluation script, creation of poster, and writing the first-draft of final report.
- **Mattheus Wolff:** Implementation of SFT, hyperparameter-tuning for SFT, debugging of DPO, hyperparameter-tuning for DPO, and editing of final report.

Changes from Proposal In our proposal, we were going to do DPO on the Ultrafeedback task. However, we decided to do it instead on the Countdown task; while this required additional effort to create an adequate preference dataset, we found it easier to work with the shorter prompts and responses within the Countdown dataset.

References

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. 2009. Curriculum learning. , 113–120 pages.

Daniel Campos. 2021. Curriculum Learning for Language Modeling. arXiv:2108.02170 [cs.CL]

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective Stars. arXiv:2503.01307 [cs.CL]

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. 2024. Stream of Search (SoS): Learning to Search in Language. arXiv:2404.03683 [cs.CL]

Tom Kocmi and Ondřej Bojar. 2017. Curriculum Learning and Minibatch Bucketing in Neural Machine Translation. arXiv:1707.09533 [cs.CL]

Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG] `https://arxiv.org/abs/1711.05101`

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. arXiv:2305.18290 [cs.CL]

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. arXiv:1609.04747 [cs.LG] `https://arxiv.org/abs/1609.04747`

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. 2025. Efficient Reinforcement Finetuning via Adaptive Curriculum Learning. arXiv:2504.05520 [cs.LG]

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum Learning: A Survey. arXiv:2101.10382 [cs.LG] `https://arxiv.org/abs/2101.10382`