REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

IMPROVING OPTIMIZATION ALGORITHMS FOR TRAINING LANGUAGE MODELS

Himmet Toprak KESGİN

DOCTOR OF PHILOSOPHY THESIS
Department of Computer Engineering
Program of Computer Engineering

Supervisor
Prof. Dr. Mehmet Fatih AMASYALI

December, 2025

REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

IMPROVING OPTIMIZATION ALGORITHMS FOR TRAINING LANGUAGE MODELS

A thesis submitted by Himmet Toprak KESGİN in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY** is approved by the committee on 01.12.2025 in Department of Computer Engineering, Program of Computer Engineering.

Prof. Dr. Mehmet Fatih AMASYALI
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Prof. Dr. Mehmet Fatih AMASYALI, Supervisor
Yildiz Technical University _____

Assoc. Prof. Dr. Ali Can KARACA, Member
Yildiz Technical University _____

Prof. Dr. Songül VARLI, Member
Yildiz Technical University _____

Prof. Dr. Arzucan ÖZGÜR, Member
Bogazici University _____

Assoc. Prof. Dr. Ayşe Berna ALTINEL GİRGİN, Member
Marmara University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of "Improving Optimization Algorithms For Training Language Models" supervised by my supervisor, Prof. Dr. Mehmet Fatih AMASYALI. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Himmet Toprak KESGİN

Signature

This study was supported by the Scientific and Technological Research Council of Türkiye (TUBITAK) Grant No: 124E055. This work was also supported by Yildiz Technical University Scientific Research Projects Coordination Unit under project number FDK-2024-6422.

Dedicated to my family

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Prof. Dr. Mehmet Fatih AMASYALI, for his invaluable guidance, support, and insights throughout my doctoral study. Sometimes I reflect on moments and think about what I could have done differently or how things could have been improved; my Ph.D. advisor is one of the rare things in my academic journey that I would never want to change.

I would also like to extend my heartfelt thanks to the esteemed members of my thesis committee, Assoc. Prof. Ali Can KARACA and Prof. Dr. Arzucan ÖZGÜR, for their valuable comments, constructive feedback, and encouraging discussions that greatly contributed to the progress and completion of this study.

I am deeply grateful to my parents for their endless love, understanding, patience, and support throughout my life and academic career. I am truly fortunate to have them; no sequence of events could balance the luck of having such parents. They are, and always have been, more than enough understanding, patience, and love.

Himmet Toprak KESGİN

TABLE OF CONTENTS

LIST OF SYMBOLS viii

LIST OF ABBREVIATIONS ix

LIST OF FIGURES xi

LIST OF TABLES xiii

ABSTRACT xiv

ÖZET xv

1 INTRODUCTION 1

1.1 Background and Problem Definition 1

1.2 Proposed Method and Objectives 2

1.3 Scope and Thesis Structure 3

2 BACKGROUND AND LITERATURE 4

2.1 Artificial intelligence 4

2.1.1 Artificial Neural Networks 6

2.1.2 Classification Tasks 6

2.2 Language Models 8

2.2.1 Measuring Success in Language Models 13

2.2.2 Motivation 17

2.3 Literature Review 18

3 METHOD 23

3.1 Overview of the Filtering Strategy 23

3.2 Statistical Foundations: Loss Tracking and Outlier Detection 24

3.3 Probabilistic Sampling and Combined Masking 27

3.4 Theoretical Impact of Fixed-Threshold Filtering on Sample Quality 28

3.5 Pseudocode of the Algorithm 33

3.6 Example Samples by Loss Region 36

3.7 Experiments 38

3.8 Results 43
3.9 Computational Efficiency 45
3.10 Robustness and Risk Evaluation 46
3.10.1 Gender Bias Amplification Analysis 47
3.10.2 Catastrophic Forgetting Analysis 48
3.10.3 Synthetic Noise Detection and Domain Adaptation 50
3.11 Analysis of the results 52

4 CONCLUSION **55**

REFERENCES **58**

PUBLICATIONS FROM THE THESIS **63**

LIST OF SYMBOLS

L	Collection of loss values over training
C	Cycle length
M_{IQR}	Deterministic inclusion mask
Q_1	First quartile of L
M	Final training mask
p_i	Inclusion probability for sample i
IQR	Interquartile Range, $Q_3 - Q_1$
$T_{\text{lower}}(t)$	Lower dynamic loss threshold
$\lambda_{\text{low}}(t)$	Lower threshold cycle coefficient
MAD	Median Absolute Deviation of L
L_i	Per-sample loss value
M_{prob}	Probabilistic inclusion mask
z_i	Robust z-score of sample i
$\alpha(t)$	Sharpness parameter for probabilistic filtering
ϵ	Small positive constant for numerical stability
Q_3	Third quartile of L
t	Training step index
$T_{\text{upper}}(t)$	Upper dynamic loss threshold
$\lambda_{\text{up}}(t)$	Upper threshold cycle coefficient

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
ARC	AI2 Reasoning Challenge
BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte Pair Encoding
CNN	Convolutional Neural Network
DL	Deep Learning
DPO	Direct Preference Optimization
GELU	Gaussian Error Linear Unit
GRPO	Group Relative Policy Optimization
GRU	Gated Recurrent Unit
GSM8K	Grade School Math 8K
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
IQR	Interquartile Range
LLM	Large Language Model
LM	Language Model
LSTM	Long Short-Term Memory
MAD	Median Absolute Deviation
MKL-SGD	Min-k Loss Stochastic Gradient Descent
ML	Machine Learning
MMLU	Massive Multitask Language Understanding
NLP	Natural Language Processing

PPL	Perplexity
PPO	Proximal Policy Optimization
ReLU	Rectified Linear Unit
RLHF	Reinforcement Learning with Human Feedback
RNN	Recurrent Neural Network
RoPE	Rotary Positional Embeddings
SGD	Stochastic Gradient Descent
SEO	Search Engine Optimization
SFT	Supervised Fine-Tuning
tanh	Hyperbolic Tangent
ToT	Tree of Thoughts
TPU	Tensor Processing Unit

LIST OF FIGURES

Figure 3.1 Left: Histogram of synthetic loss values with IQR-based thresholds (green dashed lines). Right: Robust z-score–based inclusion probabilities for different sharpness values α. Larger α values lead to narrower inclusion intervals, suppressing high- and low-loss outliers more aggressively. 25

Figure 3.2 Synchronized triangular scheduling of the three filtering parameters. Plotted on a single axis, $\lambda_{\text{low}}(t)$ (blue solid line), $\lambda_{\text{up}}(t)$ (orange dashed line), and $\alpha(t)$ (green dotted line) all follow the same period C, rising linearly from their minimum to maximum values in the first half cycle and then descending symmetrically back to the minimum. This dynamic modulation ensures balanced thresholding and sharpness control throughout training. 26

Figure 3.3 Combined filtering mechanism based on both deterministic and probabilistic criteria. Samples within the IQR interval (green region) are always selected. Extremely low-loss samples (far left) are filtered to remove redundant/repetitive content, while high-loss samples (far right) are filtered to remove noise. Outside this range, robust z-score–based probabilities ($\alpha = 1$) enable occasional inclusion of atypical examples, maintaining both stability and diversity. 28

Figure 3.4 Mixture model of per-sample loss: clean distribution ($\mu_c = 0, \sigma_c = 1$), noisy distribution ($\mu_n = 4, \sigma_n = 2$), and their combined mixture ($\tau = 0.3$). 30

Figure 3.5 Improvement in Effective Clean Rate (ΔE) across the parameter space. Each subplot represents a different noise ratio τ, with the x-axis showing noisy mean μ_n and the y-axis showing noisy standard deviation σ_n. Green regions indicate parameter combinations where IQR filtering outperforms no filtering ($\Delta E > 0$), while red regions show where filtering is suboptimal ($\Delta E \leq 0$). Clean distribution parameters are fixed at $\mu_c = 0, \sigma_c = 1$. 33

Figure 3.6 Filtering Ratio Comparison Across Datasets. The proportion of filtered examples over training steps for MC4, CulturaX, and OSCAR. Our cyclical probabilistic filtering strategy dynamically adjusts the rejection rate based on the distributional characteristics of each dataset. Notably, the model filters more aggressively in noisier corpora like MC4, while allowing more data through in relatively cleaner sources like CulturaX. This showcases the adaptive nature of the proposed method. 40

Figure 3.7 Per-Sample Loss and Dynamic Cyclical Thresholds During Training. Per-example loss values are plotted with the cyclical lower and upper thresholds. The shaded bands represent potential outlier regions that are periodically revisited. This dynamic behavior supports both the exploration of difficult examples and the exploitation of clean, consistent samples across training epochs. The plot is shows training logs on the OSCAR dataset, loss trajectories and thresholds over training steps. 42

Figure 3.8 Validation Loss Over Training: Vanilla vs. Cyclical Strategy. Validation loss curves for the MC4, CulturaX, and OSCAR datasets over 75K training steps. The cyclical probabilistic filtering method consistently achieves lower validation loss than the vanilla (unfiltered) baseline, demonstrating improved learning efficiency and robustness against noisy data. 44

Figure 3.9 Validation loss comparison with respect to wall-clock time. Despite lower throughput, cyclical filtering achieves better validation loss at any given time point, demonstrating superior learning efficiency measured on the first 10K steps of OSCAR dataset training. 46

LIST OF TABLES

Table 3.1 Notation and symbol definitions used in the precision-recall analysis. 30
Table 3.2 Cyclical In-Training Filtering Algorithm 34
Table 3.3 Cyclical Coefficient Scheduler (Triangular Wave) Algorithm . . . 34
Table 3.4 FilterBatch: In-Training Filtering via Statistics and Cycles Algorithm 35
Table 3.5 Final filtering ratio (%) per dataset using cyclical probabilistic filtering. The method converges to different rejection levels depending on dataset noise. 40
Table 3.6 Normalized accuracy (%) on Turkish HellaSwag at different training steps. Values are averaged over 3 runs conducted during training on 1,000 test samples. Cyclical probabilistic filtering consistently outperforms other methods across most configurations, achieving both higher mean accuracy and lower variance. 44
Table 3.7 Extended experimental results on Turkish HellaSwag across different dataset sizes and configurations. Results show accuracy (%) on the full test set. For 1GB experiments, values represent mean±std across three independent runs. 45
Table 3.8 Computational efficiency metrics for vanilla and cyclical filtering. 45
Table 3.9 Gender bias scores after continued pretraining on different datasets (95% CI). The baseline bias of the pretrained 1B model is 0.189 ± 0.070, serving as a reference for assessing how additional training affects bias. 48
Table 3.10 Average loss on the first 1,000 training samples across checkpoints 48
Table 3.11 Precision and recall for noise detection across domain shifts. Values in red boxes indicate performance after domain shift. . . . 51

ABSTRACT

Improving Optimization Algorithms for Training Language Models

Himmet Toprak KESGİN

Department of Computer Engineering
Doctor of Philosophy Thesis

Supervisor: Prof. Dr. Mehmet Fatih AMASYALI

Large language models (LLMs) rely on massive web-scale datasets that often contain noisy, redundant, or low-quality text, which can impair both model performance and fairness. Traditional preprocessing-based filtering methods—though widely adopted—are limited by rigid thresholds and irreversible exclusion decisions that may remove valuable examples. This thesis proposes an in-training, probabilistic, cyclical, and adaptive filtering framework that integrates data quality optimization directly into the pretraining process. The method dynamically estimates a typical loss range using robust per-sample loss statistics and temporarily excludes samples outside this range with probability proportional to their deviation. Cyclic threshold updates ensure that different difficulty regions of the data are revisited throughout training, maintaining structured diversity and reducing noise sensitivity. Experiments on Turkish-language corpora demonstrate that the proposed approach lowers validation loss and enhances downstream task performance without additional preprocessing or external scoring.

Keywords: Large language models, data quality, adaptive filtering, loss-based optimization

YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ÖZET

Dil Modellerinin Eğitimi İçin Optimizasyon Algoritmalarının Geliştirilmesi

Himmet Toprak KESGİN

Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi

Danışman: Prof. Dr. Mehmet Fatih AMASYALI

Büyük dil modelleri, genellikle gürültülü, yinelenen veya düşük kaliteli metinler içeren devasa web ölçekli veri kümelerine dayanır. Bu tür veriler, modellerin performansını ve adilliğini olumsuz etkileyebilir. Yaygın olarak kullanılan geleneksel ön işleme tabanlı filtreleme yöntemleri, katı eşik değerleri ve geri döndürülemez eleme kararları nedeniyle değerli örnekleri yanlışlıkla dışlayabilir. Bu tez, veri kalitesi optimizasyonunu doğrudan ön eğitim sürecine entegre eden, olasılıksal, döngüsel ve uyarlamalı bir filtreleme çerçevesi önermektedir. Önerilen yöntem, örnek başına kayıp istatistiklerini kullanarak dinamik bir tipik kayıp aralığı hesaplar ve bu aralığın dışındaki örnekleri, sapma miktarına bağlı bir olasılıkla geçici olarak hariç tutar. Döngüsel eşik güncellemeleri, eğitim süresince farklı zorluk bölgelerinin yeniden ziyaret edilmesini sağlayarak yapısal çeşitliliği korur ve gürültüye duyarlılığı azaltır. Türkçe veri kümeleri üzerinde yapılan deneyler, yöntemin ek bir ön işleme veya harici puanlamaya gerek duymadan doğrulama kaybını azalttığını ve aşağı akış görev performansını artırdığını göstermektedir.

Anahtar Kelimeler: Büyük dil modelleri, veri kalitesi, uyarlamalı filtreleme, kayıp tabanlı optimizasyon

YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

1
INTRODUCTION

1.1 Background and Problem Definition

Artificial intelligence (AI), a subfield of computer science, has gained extraordinary momentum in recent years, particularly through methods based on deep learning. One of the most visible outcomes of this momentum is large language models (LLMs), which have become fundamental building blocks in natural language processing (NLP) applications such as machine translation, text generation, and question-answering [1]. The success of LLMs is strongly dependent not only on architectural innovations but also on the scale, diversity, and quality of the data used in training.

Pretraining of LLMs typically involves billions of tokens, and to access data at this scale, automatically collected web text is often preferred. While open-source repositories like Common Crawl reduce costs, the lack of rigorous quality control introduces noise issues such as ads and spam content, low-quality pages, toxic language, structural/HTML (HyperText Markup Language) errors, redundant repetitions, and non-linguistic text [2], [3]. This type of noise can mislead the model, increase training time, and introduce errors and biases in the output.

Therefore, data filtering and preprocessing strategies have become central to the literature. A common approach is prescriptive/statistical procedures such as language detection, duplicate content removal, length thresholds, toxicity classification, and sometimes language model-based scoring [3], [4], [5], [6], [7], [8], [9], [10], [11]. However, stringent thresholds can inadvertently eliminate valuable examples; decisions are often deterministic and costly to reverse—trying a new threshold or method often requires retraining the model [12], [13], [14]. Additionally, pre-filtering processes can narrow language exposure and create bias by disproportionately excluding certain types of data.

On the other hand, online optimization strategies have also been developed in the

literature to deal with noise: selective backpropagation and importance sampling based on sample loss/grad norm, selections that emphasize typical/representative examples, curriculum/self-paced schemes, and adaptive thresholding [15], [16], [17], [18], [19], [20]. Recent work has also proposed approaches specific to LLM pre-training, such as model-aware data selection and loss-based ranking [2], [3]. However, a significant portion of these either make static decisions or require additional external raters; flexibility and operational simplicity are limited in single-pass, very large data setups.

1.2 Proposed Method and Objectives

In this thesis, we propose a probabilistic, cyclical, and adaptive filtering-optimization method that works during LLM pre-training to reduce pre-processing costs and the risk of irreversible decisions. At each training step, the method calculates a dynamic typical loss range using per-sample losses and robust statistics (e.g., median and Interquartile Range (IQR)). It temporarily excludes samples falling outside this range with probabilities dependent on the amount of deviation. Because the thresholds and inclusion probabilities are updated cyclically, certain sample types are not permanently excluded; the model is exposed to different loss regions at regular intervals throughout training. Thus, the signal from both easy and difficult samples is maintained with a structured diversity over time.

Our proposed framework has three main advantages: Model-adaptive, thresholds evolve with the model's learning state; examples that appear difficult in the early stages but may be useful in later steps are not permanently lost. reduced preprocessing dependency, reduces the need for fixed rules/thresholds and external raters; suitable for single-pass large-scale pretraining setups. diversity and robustness, provides balance across data regions, increases resilience to noise, and practically reduces the variance of the training signal.

The main contributions of this thesis are summarized as follows: We propose a loss-based adaptive optimization algorithm that operates during training for LLMs working with noisy data. We introduce a probabilistic inclusion mechanism and a cyclic threshold adjustment strategy based on per-sample losses, instead of relying on a fixed threshold. We demonstrate experimental performance gains with minimal computational overhead, without requiring additional preprocessing or external scoring.

1.3 Scope and Thesis Structure

We briefly clarify the fundamental concepts used in this context: Artificial intelligence, deep learning, and artificial neural networks form the foundation of the field; language models have become widespread as a subfield of deep learning, particularly with their transformer and attention architectures [1]. Thanks to hardware advancements (GPU (Graphics Processing Unit) / TPU(Tensor Processing Unit) scalability) and model-data co-design, LLMs have achieved great success by integrating the strengths of statistical approaches. In the pre-training phase, very large collections of text from the internet are used; however, this data is noisy—including HTML artifacts, repetitions, SEO (Search Engine Optimization) text, corrupted coding, and non-linguistic elements. While the literature shows that filtering such text benefits the model, it also demonstrates that filtering decisions can introduce bias and have a high cost of return [21], [22].

This thesis makes data quality optimization a natural part of the process by cyclical and adaptive sample selection during training, rather than making irreversible filtering decisions in advance. This eliminates the need for an extensive initial filtering pipeline, expensive retries, and strict exclusions, resulting in a more flexible and balanced learning process. Our method is inherently suitable for LLM pretraining and offers a lightweight and applicable solution for single-pass, web-scale, or streaming datasets.

Scope and Limitations of the Thesis. The study focuses on a framework that performs sample selection using a per-sample loss signal in the context of self-supervised pre-training. It does not require an external reference/teacher model; data raters and multiple passes are not used. Experiments are reported within practical constraints such as computational budget and data access.

The chapters of this thesis are organized as follows: Chapter 2 reviews the fundamental concepts of artificial intelligence, machine learning, and language models, providing the theoretical and literature background for the proposed method. Chapter 3 introduces the proposed in-training filtering method, detailing its statistical foundations, cyclical thresholding mechanism, and probabilistic sampling strategy, along with theoretical analysis and experimental validation demonstrating its effectiveness on Turkish datasets. Chapter 4 concludes the thesis by summarizing the key findings, discussing the implications of the proposed filtering strategy, and outlining potential directions for future research.

2
BACKGROUND AND LITERATURE

2.1 Artificial intelligence

Artificial intelligence, a subfield of Computer Science, is the general name for methods and technologies that enable computer systems to perceive data from the environment, make decisions towards specific goals, take actions and solve problems. In other words, it is the modeling, implementation, and imitation of cognitive abilities specific to human intelligence (such as perception, reasoning, learning, and problem-solving) in computers.

The importance of artificial intelligence is not limited to its speed, accuracy, and automation in technical achievements or industrial applications. Today, ethics, reliability, and social impacts have become integral components of artificial intelligence research. Artificial intelligence plays critical roles in disease diagnosis in healthcare, risk management in finance, and the safe operation of autonomous vehicles in transportation. In this context, artificial intelligence is considered not only an engineering achievement but also a transformational tool impacting our social lives.

Machine learning (ML) is a subfield of AI. ML, unlike classical programming, is a set of methods that enable computer systems to learn from experience and data without being explicitly programmed. The main goal is to make accurate predictions about unknown situations based on past data. The learning process is typically performed by adjusting the parameters of a model on input-output samples. Mathematically, the goal for a model f_θ is to minimize the expected loss:

$$\min_{\theta} \ \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\mathcal{L}(f_\theta(x), y)\big], \tag{2.1}$$

where $\mathcal{L}$ represents the loss function and $(x, y) \sim \mathcal{D}$ represents the data distribution. In practice, this expression is solved approximately over finite samples. This method is known as empirical risk minimization.

Deep learning (DL) is a subfield of machine learning that aims to learn hierarchical representations from data using multilayered artificial neural networks. Artificial neurons, simplified mathematical models inspired by neural networks in the human brain, can model complex structures by combining them in layers. With large datasets, high computational power, and an open-source culture, deep learning has revolutionized image recognition and generation, speech recognition and generation, and natural language processing.

DL model architectures are specialized for different data types and tasks: Convolutional Neural Networks (CNN): They are used primarily with image data. They demonstrate superior performance in object recognition and classification tasks. The success of AlexNet, a CNN-based model, in the 2012 ImageNet competition is a significant milestone in the field of deep learning [23]. Recurrent Neural Networks (RNN [24], LSTM (Long Short-Term Memory) [25], GRU (Gated Recurrent Unit) [26]): They were developed to model dependencies in sequential data such as time series or natural language. While RNNs capture short-term dependencies, advanced versions such as LSTM and GRU can also model long-term dependencies. Attention and Transformer Architecture [1]: Unlike traditional RNN-based methods, they use the self-attention mechanism instead of recursion to process sequential data. This allows for much more efficient capture of long-range dependencies. Introduced in 2017, the Transformer architecture revolutionized natural language processing and formed the basis for modern large language models such as GPT, BERT, and LLaMA [27], [28], [29], [30].

Machine learning is used with different learning paradigms. Supervised Learning, the model is provided with an input vector and its label. The model attempts to learn the given label from the input data. For example, understanding whether a sentence is positive or negative is an example of supervised learning.

Unsupervised Learning, unlike supervised learning, the labels of the data are not known. It is attempted to find meaningful relationships within the data itself, only through the input vector. Clustering and dimensionality reduction are typical examples.

Semi-Supervised Learning, it uses only a small amount of labeled data combined with a large amount of unlabeled data. The goal is generally to create a model that outperforms models built using only labeled data. It is often used when acquiring labeled data is costly.

Self-Supervised Learning, self-Supervised Learning is a type of supervised learning method. However, the data label is not created by humans; it is generated from the

data itself. For example, predicting the missing word in a sentence is a typical example of this approach. In this sense, the label can be generated from the data itself, with a large number of examples, at very low cost, without human labeling. In recent years, it has become one of the most widely used methods in language and image modeling. It is also compatible with the large data requirements of deep learning. Next Token prediction in Large Language Models is also based on this method.

2.1.1 Artificial Neural Networks

An artificial neural network consists of a sequential combination of linear transformations and nonlinear activations. The purpose of activation functions is to enable artificial neural networks to jointly represent nonlinear functions. A simple fully connected layer is of the form

$$h = \phi(Wx + b), \quad W \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m \tag{2.2}$$

where ϕ uses functions such as ReLU (Rectified Linear Unit) [31], GELU (Gaussian Error Linear Unit) [32], and tanh (Hyperbolic Tangent) [33]. Forward propagation generates the prediction; backpropagation uses the chain rule to calculate $\nabla_\theta \mathcal{L}$ and update the parameters. Architectural families include CNN, RNN, LSTM and transformers. In transformers, the self-attention mechanism efficiently models the remote context interaction between elements in a sequence:

$$\text{Attention}(Q, K, V) \ = \ \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)\, V, \tag{2.3}$$

where Q (query), K (key), and V (value) are linear projections; multi-headed attention captures interactions in different subspaces in parallel. Layer norm and residual connections improve gradient flow and stability in deep networks. For text, sequential structure is represented by tokenization, embedding, and position encodings.

2.1.2 Classification Tasks

Classification is one of the most fundamental tasks in machine learning. The goal is to classify input samples into specific categories. For example, determining whether an email is spam is a binary classification problem. Mathematically, the classification problem can be defined as follows:

$$f_\theta : \mathcal{X} \to \mathcal{Y}, \quad \mathcal{Y} = \{1, 2, \ldots, C\}, \tag{2.4}$$

where $\mathcal{X}$ represents the input space, and $\mathcal{Y}$ represents the label set consisting of C classes. The learned model f_θ aims to assign new incoming samples to the correct classes.

One of the most frequently used methods in classification tasks is to determine which class has the highest probability by performing probabilistic modeling. For this purpose, the *softmax* function is widely used:

$$p(y = c \mid x) = \frac{\exp(z_c)}{\sum_{j=1}^{C} \exp(z_j)}, \tag{2.5}$$

where z_j represents the network output corresponding to class j. This function determines the probability that the input vector belongs to all classes. During the training process, cross-entropy loss [33] is typically used to measure the difference between this prediction and the true label. The probability values assigned by the model using its current weights are compared with the true values, and the error is measured using loss function.

The fields of AI, ML, and DL are among the fastest-growing disciplines in today's computing world and the most widely applied in our daily lives. These methods and technologies are also at the core of LLMs.

Initially, the weights of a Deep Learning model are randomly assigned. With these weights, the model can make a prediction, even if randomly. Model learning occurs by updating the parameters to reduce the error. In other words, a prediction is made for a portion of the available data, and based on this prediction, the weights are updated in the correct direction by a certain amount.

One of the most commonly used and basic methods in this process is SGD (Stochastic Gradient Descent). Here, the model calculates the error gradient over small batches of data and updates the parameters accordingly, step by step.

The basic idea is to find the optimal parameters by taking small steps to reduce the error. The most important factor for optimization determining the learning rate. If it is chosen too large, the model can become unstable, while if it is chosen too small, learning is very slow. There are also various improved versions of SGD. For example: Momentum: It makes more balanced updates by using the knowledge of previous steps in the learning process. Adam [34]: It provides adaptive updates by taking into account both past gradient information and the gradient magnitude. Along with the AdamW [35] variant, it is one of the most preferred methods for training large-scale models. In addition, determining the change learning rate over

time (e.g., slow initial increase, then gradual decrease, etc.) and regularization techniques (e.g., dropout, normalization) help the model generalize better.

2.2 Language Models

The process of tokenization lies at the heart of language models. Tokenization is the process of breaking down text into smaller units, called tokens. Tokens can be characters, words, or word fragments. Character-based tokenization takes the smallest unit as a character, while word-based tokenization processes words directly as tokens. Today, methods based on word fragments are mostly used for efficiency reasons.

The importance of tokenization stems from the fact that language models conduct all their operations based on the sequential relationships between these token sequences. More efficient tokenization allows the model to be smaller while also encompassing a broader vocabulary. For example, tokenization methods differ in models like BERT and GPT. While GPT frequently uses Byte Pair Encoding (BPE) [36] or similar subword-based methods, BERT (Bidirectional Encoder Representations from Transformers) prefers the WordPiece method [37], [28].

As a result of this process, the text is broken down into tokens, and the model performs learning and prediction operations solely on these token sequences.

A language model is a model that probabilistically predicts the next token given an ordered sequence of tokens. Traditional language models relied on statistical or rule-based methods. However, in recent years, deep learning-based statistical approaches have become much more successful and widespread.

For example, suppose a fixed vocabulary of 32,000 tokens is created. This token set is learned from a large corpus, making texts divisible into these tokens. A language model calculates the probability of each candidate in the entire token repertoire (e.g., 32,000 tokens) for the sixth token given five consecutive tokens. This task is called next token prediction.

Training a language model is possible by automatically labeling large text corpora in this way. This is because the next token in a given sequence of tokens is directly present in the data itself. Therefore, this method is defined as self-supervised learning; the training labels are not provided by an external human being, but are generated from the data itself.

In addition to models that rely solely on next-token prediction, there are also

language models that perform masked word prediction. In these models, specific tokens are masked, and the model is expected to predict the hidden tokens. This approach is particularly used in embedding generation and contextual word representations. BERT is one of the most well-known examples of such masked language models [28].

Large language models have special tokens specifically added to the training process. For example, tokens such as <|bos|> (begin of sentence) and <|eos|> (end of sentence), which mark the beginning or end of a sentence, help the model understand the termination points in text generation. Without these tokens, the model can continue to generate new tokens continuously.

The text generation process works as follows: The language model predicts the next possible token after the current token sequence. This predicted token is added to the end of the sequence, and the process is repeated. This allows the model to continue generating data by using its own output as input again.

At this point, different sampling strategies come into play:

Greedy decoding: The token with the highest probability is selected each time.

Top-k sampling: A selection is made from among the k tokens with the highest probability.

Top-p (nucleus) sampling: A selection is made from the set of tokens that exceed a certain probability threshold (e.g., 92%).

Temperature: Controls the sharpness of the probability distribution; At low values, it becomes deterministic, while at high values, it becomes more diverse.

Repetition penalty: Prevents the production of repetitive expressions.

Thanks to these methods, language models can provide not only accuracy but also diversity and creativity.

Pretraining: The model is trained on very large-scale text data using only the next token prediction task. In this stage, the model learns the language statistically. However, this model cannot directly execute user instructions. For Example:

Document1 Sentence1. Document1 Sentence2. <|endoftext|> Document2 Sentence1. Document2 Sentence2. <|endoftext|>

Supervised Fine-Tuning (SFT): The model is trained with higher-quality data in the

form of question-answer pairs or instruction-answer formats. In this stage, user and model roles are distinguished using special tokens. For Example:

<|begin_of_text|><|start_header_id|>user <|end_header_id|> What is AI?

<|eot_id|><|start_header_id|>assistant<|end_header_id|> AI means systems that learn patterns from data. <|eot_id|>

Direct Preference Optimization (DPO) [38]: This is the stage where the model's outputs are trained using preferred and undesirable examples. For example, answers to unethical questions are marked as "undesirable," while safe answers are marked as "desirable." This reduces the likelihood of the model producing undesirable responses.

Reinforcement Learning with Human Feedback (RLHF) [39]: With methods such as PPO (Proximal Policy Optimization) [40] or GRPO (Group Relative Policy Optimization) [41], the model's responses are rewarded or punished. This allows the model to learn to respond more in line with user expectations.

In summary, although language models are primarily built on token prediction, thanks to evolving training strategies, they have evolved into systems that can understand user instructions and provide reliable and useful responses. The process, starting with tokenization, along with the pretraining, SFT, DPO, and RLHF stages, gives rise to the Large Language Model architecture used today. The development continues via tool usage, agents, etc., combining multiple language models or language model outputs.

In the continued evolution of large language models, an important development involves the simulation of internal reasoning processes using specialized tokens. After the pretraining and alignment stages, certain models incorporate an internal thinking phase, represented by tokens such as `<think>` and `<think/>`. This mechanism allows the model to generate intermediate reasoning steps before presenting a final answer.

When a user poses a question or instruction, the model first initiates its reasoning sequence beginning with `<think>` and concludes it with `<think/>`. Within this segment, the model simulates a thought process—performing multi-step reasoning, hypothesis testing, or internal deliberation—without exposing this reasoning directly to the user unless desired. After this phase, the model uses the outcome of its reasoning to produce the final visible response.

This thinking token approach enables models to better organize and justify their

outputs internally, often improving logical consistency, factual correctness, and multi-step reasoning accuracy. However, there are trade-offs: producing these intermediate reasoning sequences increases computational cost, inference latency, and memory usage. Consequently, some advanced systems allow users to toggle whether the `<think>` stage is used, balancing accuracy and efficiency according to the use case. Additionally, some frameworks permit adjusting the length or depth of the thinking phase, offering a controllable trade-off between reasoning quality and response speed.

A simplified example of reasoning with thinking tokens might look as follows:

User: What is the capital of Turkey?

<think> The user is asking for a simple factual answer. Turkey is a country located mainly in Asia and partly in Europe. The capital of Turkey is Ankara. </think> The capital of Turkey is Ankara.

This paradigm aligns with the broader research direction of chain-of-thought reasoning, where models generate structured internal thought sequences to achieve higher performance on reasoning-intensive tasks.

From this point onward, rather than operating on their own, large language models work in tandem with specialized tools and agents. These systems form the basis of multi-agent architectures, in which several language models coordinate their interactions to perform more complex tasks [42].

Looking toward the future, one of the most promising directions for language model development is the integration of multiple cooperating agents. Instead of relying on a single monolithic model, multi-agent architectures enable multiple language models—or instances of the same model—to communicate, collaborate, and delegate subtasks to one another.

In such systems, one model may specialize in reasoning, another in data retrieval or tool use, and another in summarization or evaluation. Through controlled interaction protocols, they can form dynamic networks that perform complex tasks requiring planning, reflection, and iterative correction. This framework allows large-scale coordination similar to human teamwork, where distinct agents exchange messages and negotiate intermediate results.

Another dimension of this evolution involves tool usage. Modern models can invoke external functions or APIs—such as search engines, code interpreters, or databases—to extend their capabilities beyond pure text prediction. A model can

decide when to use a tool, how to interpret the tool's output, and how to incorporate that information back into its reasoning process. This tool-augmented reasoning architecture forms the foundation for emerging agentic systems and autonomous AI workflows.

Moreover, techniques such as Tree of Thoughts (ToT) and Self-Consistency Decoding enable a single model to explore multiple reasoning paths in parallel. By generating several hypothetical reasoning branches and selecting the most consistent or coherent one, the model improves reliability and creative problem-solving. This approach transforms the traditional linear generation process into a structured exploration of ideas, leading to better performance on tasks that demand multi-step reasoning, planning, or abstract inference.

A natural extension of language models that predict only text tokens is the emergence of multi-modal models. These models not only process and generate text but also integrate other data modalities such as images, audio, and video. Through a shared embedding space, multi-modal systems learn cross-domain correspondences—linking visual and linguistic representations to understand and produce multi-sensory outputs.

For instance, a multi-modal language model can interpret an image and describe it in natural language, answer questions about it, or even generate a new image from textual input. Similarly, by including speech data, these models can process spoken commands, transcribe audio, or produce synthesized speech responses. Such capabilities extend the boundaries of natural language understanding, enabling richer and more intuitive human–AI interactions.

The fusion of modalities also improves contextual understanding: visual grounding helps disambiguate textual meaning, while language provides structured reasoning over visual data. This synergy allows models to perform advanced tasks such as visual question answering, caption generation, document understanding, and even cross-modal translation (e.g., image-to-text, text-to-audio).

Ultimately, multi-modal architectures represent a critical step toward general artificial intelligence systems that perceive, reason, and act across multiple sensory domains. By combining textual, visual, and auditory understanding within unified frameworks, these models move beyond the constraints of purely linguistic computation toward a more holistic form of machine cognition.

2.2.1 Measuring Success in Language Models

Deep learning–based language models, particularly those using Transformer architectures, are fundamentally built around predicting the probability of the next token. However, unlike classical classification tasks where each input has exactly one correct label, language modeling involves a different structure: given a sequence of tokens, there may be multiple logically valid continuations rather than a single correct next token.

Therefore, during evaluation, instead of expecting a 100% correct prediction, metrics are used to assess how confidently or uncertainly the model makes its predictions—that is, how well it models the probability distribution over possible next tokens.

The performance of a language model is not only about whether it chose the correct token, but also about how accurately it modeled the underlying probability distribution given the context. Accordingly, two complementary evaluation approaches can be considered: one based on token-level probability modeling, and another based on task-specific, application-oriented performance metrics.

During the training process of a model, the loss function is generally the primary criterion used to measure the direction and progress of learning. Especially for models that have undergone only pre-training (and not fine-tuning), low loss values on the validation set provide an important indication of how well the model has learned contextual and token-level relationships.

In addition, another widely used metric is Perplexity (PPL). Perplexity measures how surprised the model is when encountering token sequences in the test dataset—that is, how high a probability it assigns to those sequences. The smaller the perplexity value, the less surprised the model is considered to be.

Mathematically, perplexity can be expressed as follows:

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(\text{token}_i \mid \text{context}_i)\right) \tag{2.6}$$

Here, N represents the number of tokens in the test set. In this context, a decrease in validation loss and perplexity indicates an improvement in model performance.

However, an important caveat must be noted: perplexity and similar probabilistic metrics are not directly comparable across models that use different tokenizers (such

as subword, BPE, or WordPiece). This is because differences in tokenization alter the number of tokens and the structure of the context, which in turn affects the resulting measurements.

In other words, the perplexity values of two models are only meaningfully comparable if they share the same tokenizer and are evaluated under identical dataset conditions. Therefore, when conducting model comparisons, one must ensure that the same tokenization system is used.

While perplexity and loss values provide valuable insights, they do not, by themselves, fully reflect the model's performance in real-world applications or downstream tasks. For instance, two models with identical validation loss may still differ significantly in their downstream performance on tasks such as question answering, reasoning, or coherence.

Consequently, recent literature often emphasizes that pre-training loss alone is not sufficient to evaluate the overall capability or usefulness of a model.

Language models do more than just predict tokens. After supervised fine-tuning or Reinforcement Learning with Human Feedback , they gain capabilities such as answering user questions and performing classification tasks.

For such tasks, evaluation is often conducted as follows: the model is presented with a question and four possible options (A, B, C, D). The model computes the probability of each option, and the one with the highest probability is taken as the model's answer. This predicted answer is then compared against the ground-truth labels in the test set, thereby measuring the model's accuracy in making correct selections.

For example, in a sentiment analysis task, the model might compare the probabilities of "this sentence is positive" versus "this sentence is negative." The accuracy can then be determined by how often the model's higher-probability prediction matches the true label.

Today, there are numerous standard benchmark datasets used to evaluate the performance of language models. Some of the most prominent include:

MMLU (Measuring Massive Multitask Language Understanding): Consists of multiple-choice questions spanning a wide range of academic disciplines.

TruthfulQA: Designed to assess factual accuracy and resistance to common misconceptions or falsehoods.

ARC (AI2 Reasoning Challenge): Focuses on questions requiring general scientific knowledge, though typically less diverse than MMLU.

GSM8K (Grade School Math 8K): Contains grade school and high school–level mathematical reasoning problems.

Winogrande: Tests commonsense reasoning and the model's ability to resolve contextual ambiguities.

HellaSwag: A multiple-choice completion benchmark designed to measure commonsense understanding.

In these benchmarks, the random guess baseline (e.g., 25% accuracy in a four-choice test) serves as a reference point. A well-performing model is expected to significantly exceed this baseline—performance below it (such as under 25% on HellaSwag) would be considered meaningless.

Another critical aspect of evaluation is how many examples the model is shown before being tested.

Zero-shot: The model receives no examples prior to testing; only the task description or prompt is provided.

Few-shot: The model is shown a small number of input–output examples (e.g., 5) before being tested on real questions. This demonstration helps align the model to the task format and expected response structure.

In this context, an important measure of a language model's capability is how well it performs not only on trained tasks but also on new tasks after limited demonstration. For models that undergo SFT or are designed to interact directly with users, evaluation extends beyond automated metrics to include human judgments. In such systems, human evaluators compare model responses—often without knowing which model produced which answer—and vote on which output they prefer. These pairwise comparisons can be aggregated into ELO-like scoring systems, similar to competitive ranking in games.

This approach enables evaluation not just of factual correctness, but also of response quality, helpfulness, and preference alignment. for models trained or aligned through SFT, evaluation can involve: Automatic accuracy metrics (on labeled datasets), Human preference ratings (quality-based evaluations), Model-to-model comparison systems (e.g., arena setups). Together, these provide a more holistic understanding of model performance beyond traditional perplexity or loss-based

metrics.

If a model is still in its pre-training phase, fundamental metrics such as loss and perplexity should serve as the primary indicators of progress and performance.

However, if the model has undergone task-specific fine-tuning or has been aligned to interact directly with users, evaluation should prioritize application-oriented benchmarks, zero/few-shot tests, and, where appropriate, human evaluations.

When comparing models across different measurement settings, it is essential to ensure that the following conditions are consistent: the same tokenizer (eg. PPL), the same test dataset, the same task definition, and the same shot configuration (zero- or few-shot).

Otherwise, the results may be misleading or non-comparable.

Moreover, relying on a single metric is insufficient. For example, studies have shown that a model with low perplexity may still perform poorly on downstream tasks, revealing that perplexity alone does not fully capture practical performance.

Finally, post-evaluation analysis is equally important: identifying the tasks in which the model struggles, examining the nature of its errors, and clarifying the contexts in which each metric is valid or informative should all be explicitly reported.

In conclusion, evaluating the success of language models is a complex and multidimensional process.

Probabilistic, token-prediction–based metrics such as loss and perplexity reflect a model's fundamental linguistic capability, whereas task-based evaluations—including classification, question answering, and logical continuation tasks—reveal its applied competence.

As language models evolve from mere token predictors into systems capable of reasoning, dialogue, and interaction with users, evaluation methods must likewise become multifaceted and context-sensitive.

The set of measurement approaches summarized in this section provides a foundational framework for assessing language models within research contexts, enabling a balanced and comprehensive understanding of model performance across both theoretical and practical dimensions.

2.2.2 Motivation

The most critical factor in the success of LLMs is the massive-scale text data used in LLM training. LLMs, typically built on transformer architectures [1], require training on billions of tokens to achieve effective generalization; this need is directly linked not only to model architecture but also to the quality and diversity of the data. To meet this large-scale data requirement, researchers typically use texts automatically collected from the web. Open data sources such as Common Crawl facilitate easy access to vast amounts of text, helping to reduce the costs of creating and collecting training datasets. However, this web text data also brings significant quality issues stemming from the lack of rigorous quality control [43].

Web-based datasets typically contain noise elements such as advertisements, repetitive content, spam, low-quality pages, toxic expressions, inaccurate information, biased content, and non-linguistic content. Such noisy data negatively affects the language model's learning process, prolongs training time, and can lead to unwanted content, biases, or errors in outputs.

Given these challenges in language model training, effective filtering strategies to ensure the quality of training data have become essential. To mitigate the effects of noisy web-based text data, researchers typically rely on preprocessing techniques that include rule-based filters or statistical ranking methods based on metrics such as loss-based perplexity from another model [4]. Examples include language detection, toxicity classification, and text length filters aimed at training models on cleaner data [2], [3]. Web data can be filtered using such rules or models capable of distinguishing quality data.

However, these methods have quite significant limitations. Rule-based systems typically use rigid thresholds, potentially eliminating valuable data [12]. Moreover, these approaches are generally deterministic and make filtering decisions irreversible. Experimenting with new filtering strategies is also costly, as each requires retraining the model.

In this thesis, we propose a novel approach to address these limitations: a probabilistic filtering strategy based on the loss values of text data during model training, eliminating the need for prior data scanning. Our method calculates a dynamic loss range at each training step using robust statistics (e.g., interquartile range), and samples falling far outside this range are probabilistically excluded based on their deviation from the median loss. More importantly, since the thresholds vary cyclically, the model avoids consistently excluding certain example types, ensuring balanced exposure to different data regions even within a single data

pass.

Our method cyclically varies the selection criteria throughout training. This prevents the model from overfitting to a narrow subset of easy or hard examples and encourages structured diversity in the training signal. The resulting framework is lightweight, requires no additional preprocessing or external scoring, and is particularly suitable for noisy, large-scale, single-pass training regimes such as those used in LLM pre-training. It is integrated into the training loop without any additional cost prior to training. The key advantages of this approach can be summarized as follows: first, it adapts to the model's evolving understanding during training, preventing premature exclusion of potentially useful data; second, it removes dependency on fixed preprocessing methods; and finally, it preserves example diversity and enhances robustness in noisy training data.

2.3 Literature Review

This chapter of this thesis examines data-available approaches for pre-training large language models. First, we examine filtering-based techniques that aim to clean text before pre-training a large language model. Then, we examine optimization-based strategies that process noisy examples during training. This chapter aims to contextualize our proposed method within these two areas of research and highlight how it addresses their respective limitations.

Rule-based approaches aim to remove low-quality data before training begins. These methods typically rely on language detection, data deduplication, some heuristics, and quality scoring rules. Different models can be used to calculate these scores.

Preprocessing noisy web-based text has become an essential step for achieving robust LLM pre-training [44], [29]. While massive collections like Common Crawl offer diverse linguistic coverage [6], [45], [7] but they also contain substantial noise, including duplicated, low-quality, or artificially generated text [46], [47].

To minimize these issues, several high-profile efforts have developed rule-based or heuristic pipelines to preprocess data—such as C4, CCNet, OSCAR, RefinedWeb, CulturaX, FineWeb, and Dolma—using techniques like language detection, deduplication, structural parsing, and content scoring [5], [6], [7], [8], [9].

These methods and strategies aim to increase quality text ratio in pretraining corpora, which is shown to directly impact model performance [3], [14].

Recent research has advanced this field by incorporating learning-based filtering approaches that utilize ensemble classifiers or loss-based re-ranking methods to retain valuable samples while eliminating noisy data [48], [2]. For example, FineWeb and DCLM employ language models or classifiers to detect high-quality samples and even make attempts to reconstruct text of lower quality [3], [48]. Nevertheless, such methods generally depend on fixed thresholds or predefined classifiers, which constrain adaptability and introduce the risk of prematurely removing valuable data and increase and introduce new biases for text data [13], [14].

Another recent study explores knowledge-based filtering: a High-Knowledge Scorer screens training data using measures such as knowledge density and domain coverage, which in turn boosts generalization on knowledge-intensive tasks [49].

In contrast to these works, this thesis proposes an in-training, cyclical probabilistic filtering method that adapts over time based on observed loss signals—eliminating preprocessing text costs before training, preserving dataset diversity, and better aligning the filtering process in the process of model training with model learning dynamics.

Beyond preprocessing techniques, another complementary line of research emphasizes adapting the training process itself to accommodate noisy or anomalous data points. These optimization-centric methods work in an online fashion—throughout the training phase—utilizing metrics like individual sample losses, gradient information, or density estimates to dynamically regulate the learning process. The method proposed in this thesis can be considered a member of such a robust optimization technique group. In contrast to preprocessing filters that remain static, these approaches respond adaptively to the changing characteristics of the data in real time, enhancing both robustness and computational efficiency without depending on predefined rules or repeated data passes. In certain frameworks, including curriculum learning, the sequencing of training examples is also considered a critical element, steering the model progressively from simpler to more challenging instances to maintain training stability in noisy conditions [15].

A subset of methods aims to reduce the variance in gradient updates by emphasizing representative or typical samples during training. For instance, clustering-based SG-MCMC [50] aggregates gradients per cluster rather than uniformly sampling data, offering scalable variance reduction during training. This approach partitions the dataset using k-means clustering in a preprocessing step and maintains fixed clusters throughout the simulation, thereby mitigating the high variance from naive

subsampling without adding computational overhead during sampling.

Some studies propose non-uniform sampling for SGD, such as Typical Batch SGD [18] and importance sampling based on gradient norms [17], to prioritize impactful or representative examples and reduce noisy updates. These approaches reduce gradient variance and accelerate convergence, outperforming uniform sampling while remaining compatible with standard optimizers. By focusing computation on informative examples through tractable upper bounds on gradient norms or typicality-based selection, these methods achieve significant reductions in training loss and improved test performance within fixed training time.

In a similar vein, loss-based traing sample selection has gained traction. The Selective-Backprop algorithm [16] accelerates training by probabilistically backpropagating only on hard examples with high loss values. This technique reduces computational cost by avoiding unnecessary backward passes on well-learned easy training samples, while a variant, Stale-SB, reuses forward loss estimates across epochs to further improve efficiency. These methods offer a practical alternative to full-batch or uniform gradient updates, especially in large-scale setups with long training time.

Recent advances have extended online selection strategies to the pretraining phase of large language models. MATES [51] introduces a model-aware framework that continuously adapts data selection by locally probing the pretraining model and approximating data influence on a hold-out dataset. GREATS [52] employs greedy approximation with Taylor expansion for online batch selection, though it primarily targets fine-tuning scenarios which is a different approach from pretraining objective. One of the main disadvantages of these methods is that they require additional reference or validation datasets, whereas our algorithm operates directly on the training data itself.

Beyond prioritization of difficult training samples, other algorithms aim to eliminate noisy or mislabeled data during training. The Adaptive-k optimization algorithm [19] dynamically filters out samples with unusually high losses samples by comparing them to an adaptive, moving loss threshold inspired by Adam-like optimization algorithm statistics. Using these statistics algorithm can work successfully adaptively to different noise levels. This approach avoids the use of fixed heuristics or known noise ratios, enabling more flexible and resilient training even when noisy labels are abundant in the dataset.

Complementary to Adaptive-k, the Min-k Loss SGD (MKL-SGD) approach offers a lightweight solution for outlier rejection [20]. Rather than averaging gradients over

all samples, MKL-SGD stochastically samples a subset and selects the lowest-loss sample for the update. This mechanism filters out high-loss (often noisy) data in a principled manner, and has demonstrated improved generalization in settings with label corruption or adversarial noise.

Some curriculum-based methods dynamically schedule sample difficulty to guide the learning process from easier to harder instances [53]. Relatedly, Self-Paced Learning with Diversity [54] selects samples based on both easiness and diversity, ensuring that training data remains varied and avoids homogeneous patterns.

Token-level selective learning has emerged as another approach to enhance data efficiency during pretraining. Rho-1 [55] introduces Selective Language Modeling, which challenges the conventional uniform next-token prediction by selectively training on tokens that align with the desired distribution, as determined by a reference model's scoring mechanism. This fine-grained selection strategy significantly reduces pretraining costs while maintaining or even improving performance on mathematical reasoning and general tasks. Empirical results demonstrate that Rho-1 can achieve state-of-the-art accuracy on benchmarks like MATH while using only a fraction of the training tokens required by traditional approaches, highlighting the potential of token-level curriculum learning in large-scale pretraining. Unlike this study, the algorithm proposed in this thesis works at the instance level, not at the token level.

Another recent LLM alignment techniques have introduced multi-dimensional curriculum frameworks that go beyond the traditional single-axis difficulty scheduling. For instance, 2D-Curri-DPO structures training samples along two key dimensions — prompt complexity and preference clarity. By traversing these axes using flexible strategies, it effectively optimizes LLM alignment outcomes [56]. This stands in contrast to conventional curriculum learning, which typically depends on a single difficulty metric. Similarly, the Perplexity Difference-based Preference Curriculum approach organizes training data according to how distinguishable a sample is between weaker and stronger models [57]. This dynamic curriculum progressively adjusts the difficulty of samples to match the model's evolving capacity, all without interrupting the training process. The method has demonstrated notable improvements in handling challenging benchmarks such as MMLU [58].

Literature on reinforcement learning in LLMs has also influenced mechanisms of staged and selective learning. For example, D3RQN [59] introduces a reinforcement learning framework that focuses updates on critical time steps within partially observable environments, improving training stability through

adaptive update frequency—a concept that shares similarities with our cyclical explore–exploit filtering mechanism.

It also has been shown that ChatGPT-assisted filtering can help fine-tune scenarios with label noise. A study by Wang et al. [60] proposes using LLMs to identify and correct mislabeled samples, which will improve downstream accuracy under weak supervision.

Moreover, recent work has explored the application of LLMs for denoising in non-linguistic domains. LLM4DSR [61] employs LLMs to clean noisy sequential recommendation data by combining self-supervised fine-tuning with uncertainty-based output filtering. This demonstrates the broader potential of LLMs as general-purpose noise reduction tools across diverse data modalities.

Knowledge distillation methods have been developed to compress LLM capabilities into more compact models. DLLM2Rec [62] integrates confidence-based and embedding-based strategies to enhance student models for sequential recommendation tasks by extracting knowledge from LLMs, while simultaneously filtering out noise from uncertain teacher predictions.

Recent optimization-driven strategies represent a shift toward adaptive, loss-aware training algorithms that integrate data selection directly into the learning process, moving away from traditional preprocessing pipelines that rely on static, offline filtering. In this direction, we propose a probabilistic, cyclical, loss based in-training filtering approach specifically designed for LLM pretraining objective. Our method dynamically computes loss-based thresholds using robust statistics and modulates them over time through a cyclical schedule. This enables selective inclusion of samples from varying loss regions throughout training—even within a single epoch—thereby enhancing diversity and robustness without permanently discarding any data regions or subsets. By avoiding fixed heuristics and external scoring mechanisms, our approach offers a lightweight, model-aware alternative well-suited for noisy, web-scale, or streaming large datasets for long training.

3
METHOD

3.1 Overview of the Filtering Strategy

The components of our filtering algorithm are presented in this chapter, along with an explanation of how they interact to select training samples in a dynamic manner. Large-scale datasets used for LLM pretraining often contain noise, redundancy, and low-value content. Rather than depending on offline filtering, we introduce a lightweight, in-training strategy that dynamically selects samples according to their loss values. Our approach combines robust statistical methods, cyclical thresholding, and probabilistic sampling to decide which samples contribute to backpropagation. In contrast to conventional methods that target only high-loss (noisy) samples, we also exclude very low-loss samples that typically reflect repetitive or trivial data, allowing the model to focus on the most informative examples.

During each iteration, filtering choices are determined using a dynamically calculated range derived from the historical distribution of losses. Notably, these filtering boundaries change in a cyclical pattern, enabling the sampling of various segments within the loss distribution across different time periods. This cyclic approach prevents the algorithm from persistently prioritizing either simple or complex examples, thereby promoting diversity in learned representations and enhancing model robustness. In contrast to offline filters, our approach reduces overhead by leveraging per-sample losses computed during the forward pass.

While prior research [63] has shown that high-loss samples often indicate noisy or corrupted data, the reasoning behind excluding extremely low-loss samples warrants further explanation. Our method focuses on filtering both ends of the loss spectrum based on the following insights:

High-loss samples are frequently associated with corrupted text, incoherent sequences, or data from mismatched domains. In contrast, extremely low-loss

samples tend to correspond to highly repetitive or templated text, redundant duplicates, or trivial patterns that contribute little to effective learning—illustrated further in Section 3.6.

This observation is consistent with recent work [64], which identifies low-loss samples as easy or repetitive and potentially redundant. Importantly, our focus is limited to extremely low-loss samples in the far tail of the distribution, excluding moderately low-loss samples that represent genuinely well-learned content. Examples of both extreme low and high-loss texts are provided in Section 3.6.

3.2 Statistical Foundations: Loss Tracking and Outlier Detection

Consider $L = \{L_1, L_2, ..., L_t\}$ as the accumulated collection of loss values per sample recorded until timestep t, where L_i denotes the sequence-level loss calculated by averaging token-level cross-entropy across all tokens in the i-th training sequence.

Choice of Robust Statistics: We utilize median-centered robust statistical measures given their resilience to outliers present in web-scale datasets. Unlike the mean, which can be substantially influenced by extreme observations, the median maintains stability. The interquartile range (IQR) characterizes the middle 50% of the data distribution, establishing thresholds that remain unaffected by extreme tail values. Likewise, z-scores derived from median absolute deviation (MAD) provide more reliable measures of deviation compared to conventional z-scores in the presence of occasional extreme loss values [65].

We compute the first (Q_1) and third (Q_3) quartiles of L, and derive the interquartile range:

$$\mathrm{IQR} = Q_3 - Q_1 \tag{3.1}$$

Robust Z-Score: To assess how atypical the loss of a sample is, we compute a robust z-score using the median and the MAD:

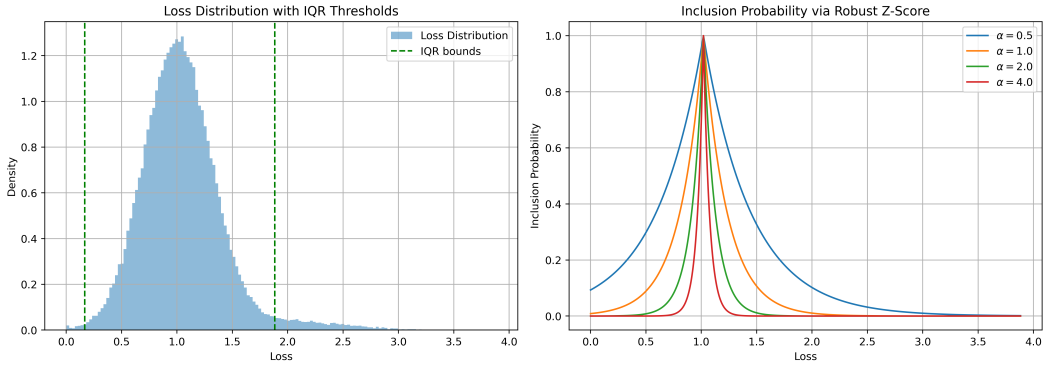


Figure 3.1 Left: Histogram of synthetic loss values with IQR-based thresholds (green dashed lines). Right: Robust z-score–based inclusion probabilities for different sharpness values α. Larger α values lead to narrower inclusion intervals, suppressing high- and low-loss outliers more aggressively.

The loss values used here are synthetic and generated for illustration purposes. The distributions were constructed to resemble common patterns (e.g., long-tail shape) observed in real LLM loss profiles but do not come from an actual model.

$$z_i = \frac{L_i - \text{median}(L)}{\text{MAD}(L) + \epsilon} \tag{3.2}$$

Unlike the standard z-score, which depends on the mean and standard deviation, the robust z-score is less affected by outliers and moderate skewness—making it a more dependable choice for training loss distributions that occasionally include unusually high-loss samples. To minimize memory usage and prevent outdated statistics, future implementations could compute these values within a sliding window covering only the most recent N loss samples.

Figure 3.1 demonstrates the impact of robust z-score–based selection, illustrating how different sharpness parameters α affect the inclusion probability across the loss distribution. Smaller α values result in broader inclusion, whereas larger values strongly suppress both high- and low-loss outliers.

Cyclical Thresholding via Triangular Scheduling:

We avoid rigid filtering by introducing cyclical thresholding based on time-varying loss intervals. The IQR thresholds are modulated using time-varying coefficients $\lambda_{\text{low}}(t)$ and $\lambda_{\text{up}}(t)$:

$$T_{\text{lower}}(t) = Q_1 + \lambda_{\text{low}}(t) \cdot \text{IQR}, \quad T_{\text{upper}}(t) = Q_3 + \lambda_{\text{up}}(t) \cdot \text{IQR} \tag{3.3}$$

These coefficients follow a symmetric triangular waveform with period C:

$$\lambda(t) = \begin{cases} \lambda_{\min} + \frac{2(\lambda_{\max} - \lambda_{\min})}{C} \cdot (t \bmod C), & t \bmod C < \frac{C}{2} \\ \lambda_{\max} - \frac{2(\lambda_{\max} - \lambda_{\min})}{C} \cdot \left(t \bmod C - \frac{C}{2}\right), & \text{otherwise} \end{cases} \tag{3.4}$$

The coefficient $\lambda(t)$ increases linearly from $\lambda_{\min}$ to $\lambda_{\max}$ during the first half of each cycle, then decreases linearly back to $\lambda_{\min}$ in the second half, forming a symmetric triangular wave. This behavior is visualized in Figure 3.2, showing how each parameter cycles over time.

The sharpness parameter $\alpha(t)$ follows the same triangular waveform, modulating the sensitivity of probabilistic filtering. High $\alpha(t)$ values produce sharper decay in inclusion probabilities, enforcing stricter filtering:

$$p_i = \exp(-\alpha(t) \cdot |z_i|) \tag{3.5}$$

As $\alpha(t)$ increases, the filtering becomes more selective, reducing the likelihood of including high-deviation samples.

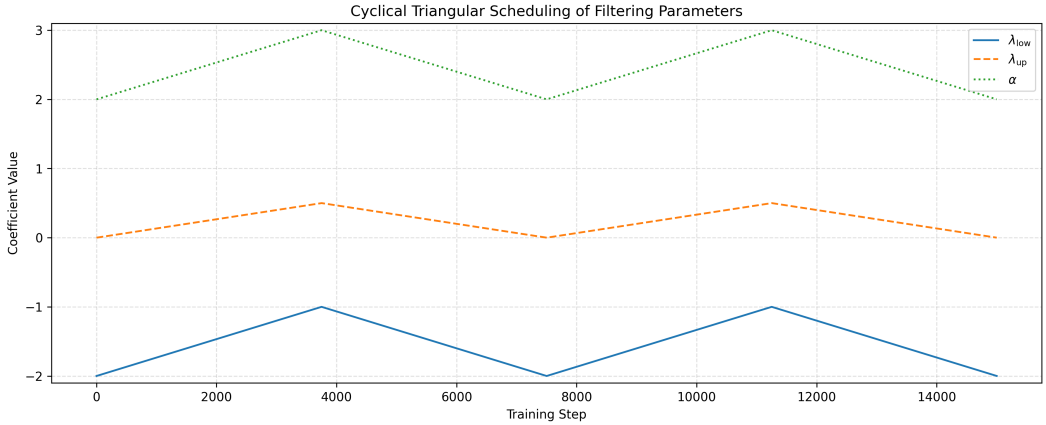


Figure 3.2 Synchronized triangular scheduling of the three filtering parameters. Plotted on a single axis, $\lambda_{\text{low}}(t)$ (blue solid line), $\lambda_{\text{up}}(t)$ (orange dashed line), and $\alpha(t)$ (green dotted line) all follow the same period C, rising linearly from their minimum to maximum values in the first half cycle and then descending symmetrically back to the minimum. This dynamic modulation ensures balanced thresholding and sharpness control throughout training.

This cyclical scheduling prevents consistent exclusion or overemphasis of any specific loss region, promoting balanced training exposure. This cyclical modulation is motivated by the need to avoid systematic bias in training signal,

allowing the model to explore a wider range of data characteristics over time.

3.3 Probabilistic Sampling and Combined Masking

Our filtering method employs two sequential steps: rule-based selection for typical observations and chance-based selection for outliers. First, we capture observations within flexible IQR boundaries (automatically retained), then compute retention probabilities for all observations based on how far they deviate from the median. The ultimate mask merges these two approaches.

We integrate deterministic and probabilistic mechanisms for sample selection.

IQR-based Deterministic Filtering. Samples within dynamic thresholds are always selected:

$$M_{\text{IQR}} = \{i \mid T_{\text{lower}}(t) < L_i < T_{\text{upper}}(t)\} \tag{3.6}$$

Samples within these bounds are guaranteed inclusion.

Probabilistic Filtering via Robust Z-Scores. For all samples, inclusion probability is computed using their absolute z-score (Eq. 3.2) and a sharpness parameter $\alpha(t)$:

$$p_i = \exp\left(-\alpha(t) \cdot |z_i|\right) \tag{3.7}$$

Here, $\epsilon > 0$ is a small constant for numerical stability. A sample is included with probability p_i by sampling $r_i \sim \mathcal{U}(0, 1)$.

Higher deviations yield lower inclusion probabilities.

$$M_{\text{prob}} = \{i \mid r_i < p_i\} \tag{3.8}$$

This creates a stochastic selection based on deviation.

Final Mask Combination. The final training mask is the union:

$$M = M_{\text{IQR}} \cup M_{\text{prob}} \tag{3.9}$$

The union ensures typical samples are kept while allowing occasional outliers.

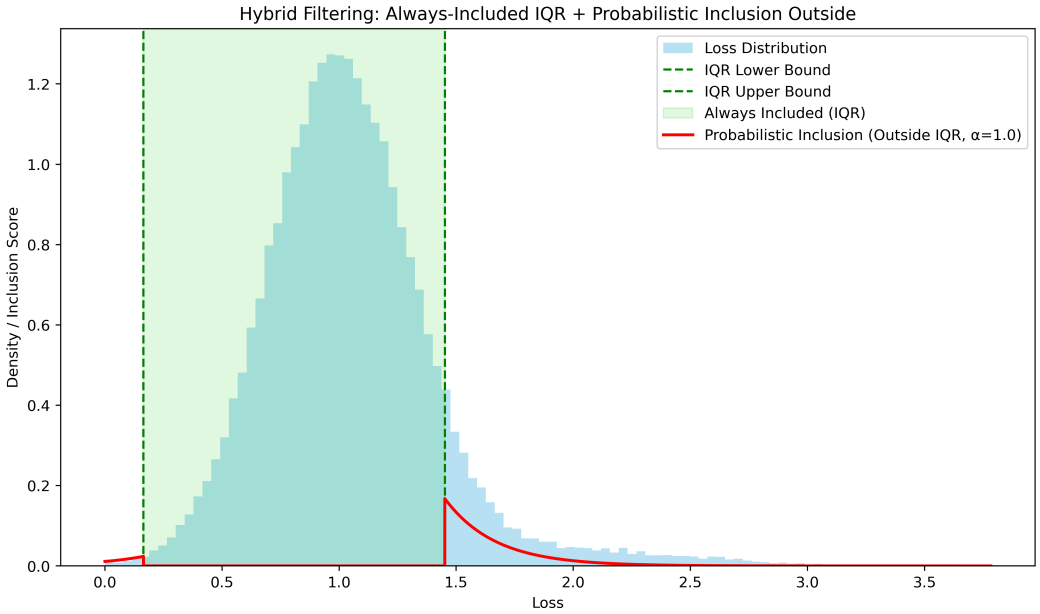


Figure 3.3 Combined filtering mechanism based on both deterministic and probabilistic criteria. Samples within the IQR interval (green region) are always selected. Extremely low-loss samples (far left) are filtered to remove redundant/repetitive content, while high-loss samples (far right) are filtered to remove noise. Outside this range, robust z-score–based probabilities ($\alpha = 1$) enable occasional inclusion of atypical examples, maintaining both stability and diversity.

Figure 3.3 is based on synthetic data generated to mimic realistic LLM loss distributions and is intended purely as an illustration of the proposed filtering behavior. The red curve corresponds to the inclusion probability $p_i = \exp(-\alpha|z_i|)$, outside the IQR. For visualization purposes, it is scaled to align with the histogram's vertical axis; this representation is qualitative and does not reflect actual probability magnitudes.

This hybrid masking balances stability and diversity—selecting typical samples deterministically while probabilistically retaining outliers—without permanently discarding any loss region. Fig. 3.3 illustrates the combined masking strategy.

3.4 Theoretical Impact of Fixed-Threshold Filtering on Sample Quality

For conceptual clarity, in this section we replace the dynamic, cyclical thresholds discussed in Section 3.2 with a fixed IQR-based threshold $T = Q_3 + 1.5\,\mathrm{IQR}$. This simplification enables us to represent the training dataset as a mixture of clean (useful) and noisy (harmful) samples, each characterized by distinct loss

distributions, and to derive interpretable measures such as recall, precision, and effective clean rate. Through this framework, we can rigorously quantify how filtering influences both the preservation of beneficial samples and the exclusion of detrimental ones. Although this model does not capture the full dynamics of our actual method, it offers valuable theoretical insights into the effects of static filtering on data quality under idealized conditions.

Mixture Model Formulation: In order to represent the fundamental hypothesis that training data contains two distinct populations, we model the per-sample loss distribution as a two-component Gaussian mixture:

- Clean examples: $X_c \sim \mathcal{N}(\mu_c, \sigma_c^2)$ - samples that contribute positively to model training
- Noisy examples: $X_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$ - samples that may be harmful or less useful ($\mu_n > \mu_c$)

The prior probability of encountering a noisy example is $\tau = P(\text{noisy})$, with clean examples occurring at rate $1 - \tau = P(\text{clean})$.

The mixture probability density function (PDF) is therefore:

$$f_{\text{mix}}(x) = (1 - \tau) \cdot f_c(x) + \tau \cdot f_n(x) \tag{3.10}$$

where f_c and f_n are the PDFs of the clean and noisy distributions, respectively.

This mixture model is justified by empirical observations of bimodal loss distributions in large-scale training, where problematic samples (e.g., near-duplicates, inconsistent examples) tend to produce distinctly higher loss values.

Figure 3.4 shows an example of a distribution.

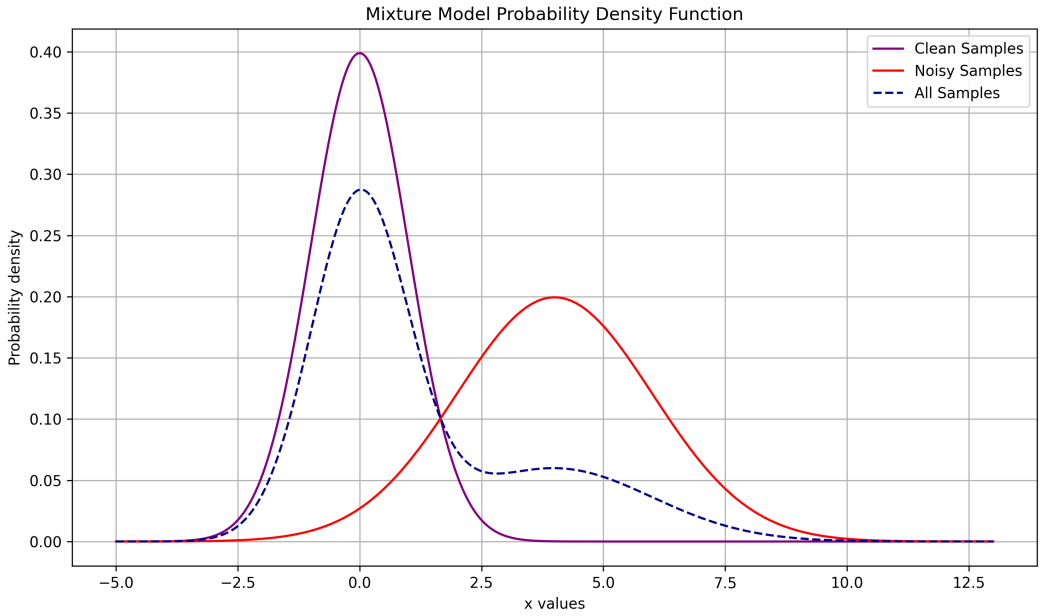


Figure 3.4 Mixture model of per-sample loss: clean distribution ($\mu_c = 0, \sigma_c = 1$), noisy distribution ($\mu_n = 4, \sigma_n = 2$), and their combined mixture ($\tau = 0.3$).

Before defining our performance metrics, we introduce a nomenclature table to ensure clarity:

Table 3.1 Notation and symbol definitions used in the precision-recall analysis.

Symbol	Definition
τ	Prior probability of a noisy sample in the dataset
μ_c, σ_c	Mean and standard deviation of the clean loss distribution
μ_n, σ_n	Mean and standard deviation of the noisy loss distribution
T	IQR-based filtering threshold, defined as $Q_3 + 1.5\,\mathrm{IQR}$
r_c	Retention rate of clean samples: $P(X_c \leq T)$
r_n	Retention rate of noisy samples: $P(X_n \leq T)$
R	Recall: fraction of clean samples retained after filtering
P	Precision: fraction of retained samples that are clean
E	Effective Clean Rate: combined metric $R \times P$

We define the IQR-based threshold $T = Q_3 + 1.5\,\mathrm{IQR}$, where $Q_3 = F_{\mathrm{mix}}^{-1}(0.75)$ is the third quartile and $\mathrm{IQR} = Q_3 - F_{\mathrm{mix}}^{-1}(0.25)$ is the interquartile range of the mixture distribution.

For both clean and noisy components, we calculate the retention rates (probability

of being kept after filtering):

$$r_c = P(X_c \leq T) = \Phi\left(\frac{T - \mu_c}{\sigma_c}\right) \tag{3.11}$$

$$r_n = P(X_n \leq T) = \Phi\left(\frac{T - \mu_n}{\sigma_n}\right) \tag{3.12}$$

where Φ is the cumulative distribution function of the standard normal distribution.

To evaluate filtering effectiveness, we use three complementary metrics:

- Recall (R_{IQR}): The fraction of clean samples retained after filtering

$$R_{\text{IQR}} = r_c = \frac{\text{clean samples retained}}{\text{total clean samples}} \tag{3.13}$$

- Precision (P_{IQR}): The fraction of retained samples that are clean

$$P_{\text{IQR}} = \frac{(1-\tau) \cdot r_c}{(1-\tau) \cdot r_c + \tau \cdot r_n} = \frac{\text{clean samples retained}}{\text{total samples retained}} \tag{3.14}$$

- Effective Clean Rate (E_{IQR}): The overall efficiency metric combining both recall and precision

$$E_{\text{IQR}} = R_{\text{IQR}} \times P_{\text{IQR}} = \frac{(1-\tau) \cdot r_c^2}{(1-\tau) \cdot r_c + \tau \cdot r_n} \tag{3.15}$$

We choose the Effective Clean Rate as our main evaluation metric since it encompasses both aspects of filtering effectiveness: the proportion of retained high-quality samples (recall) and the purity of the filtered dataset (precision). This unified metric enables us to measure the comprehensive effect of filtering on training data quality through a single numerical value.

Baseline Performance (No-Filtering): Without filtering, all samples are retained regardless of their loss values. This yields:

$$R_{\text{vanilla}} = 1 \quad \text{(all clean samples are kept)} \tag{3.16}$$

$$P_{\text{vanilla}} = 1 - \tau \quad \text{(proportion of clean samples in the original mixture)} \tag{3.17}$$

$$E_{\text{vanilla}} = 1 \times (1-\tau) = 1 - \tau \quad \text{(effective clean rate equals the clean prior)} \tag{3.18}$$

IQR-Based Filtering Performance: When we discard samples with losses exceeding

threshold T, the metrics become:

$$R_{\text{IQR}} = r_c \quad \text{(fraction of clean samples with loss below } T\text{)} \tag{3.19}$$

$$P_{\text{IQR}} = \frac{(1-\tau) \cdot r_c}{(1-\tau) \cdot r_c + \tau \cdot r_n} \quad \text{(clean samples retained divided by total retained)} \tag{3.20}$$

$$E_{\text{IQR}} = R_{\text{IQR}} \times P_{\text{IQR}} = \frac{(1-\tau) \cdot r_c^2}{(1-\tau) \cdot r_c + \tau \cdot r_n} \quad \text{(combined effect)} \tag{3.21}$$

The filtering approach improves training data quality precisely when:

$$E_{\text{IQR}} > E_{\text{vanilla}} \Rightarrow \frac{(1-\tau) \cdot r_c^2}{(1-\tau) \cdot r_c + \tau \cdot r_n} > 1 - \tau \tag{3.22}$$

This inequality simplifies to:

$$r_c > \frac{(1-\tau) \cdot r_c + \tau \cdot r_n}{1-\tau} \Rightarrow r_c > r_n \tag{3.23}$$

Thus, our filtering approach is beneficial whenever clean samples have a higher retention rate than noisy samples ($r_c > r_n$), which is typically the case when $\mu_n > \mu_c$ (i.e., noisy samples tend to have higher losses).

Parameter Space Analysis: We analyze how the effectiveness of our filtering approach varies across different distribution parameters. Figure 3.5 shows the improvement in Effective Clean Rate ($\Delta E = E_{\text{IQR}} - E_{\text{vanilla}}$) as a function of noisy distribution parameters (μ_n and σ_n) for various noise levels (τ).

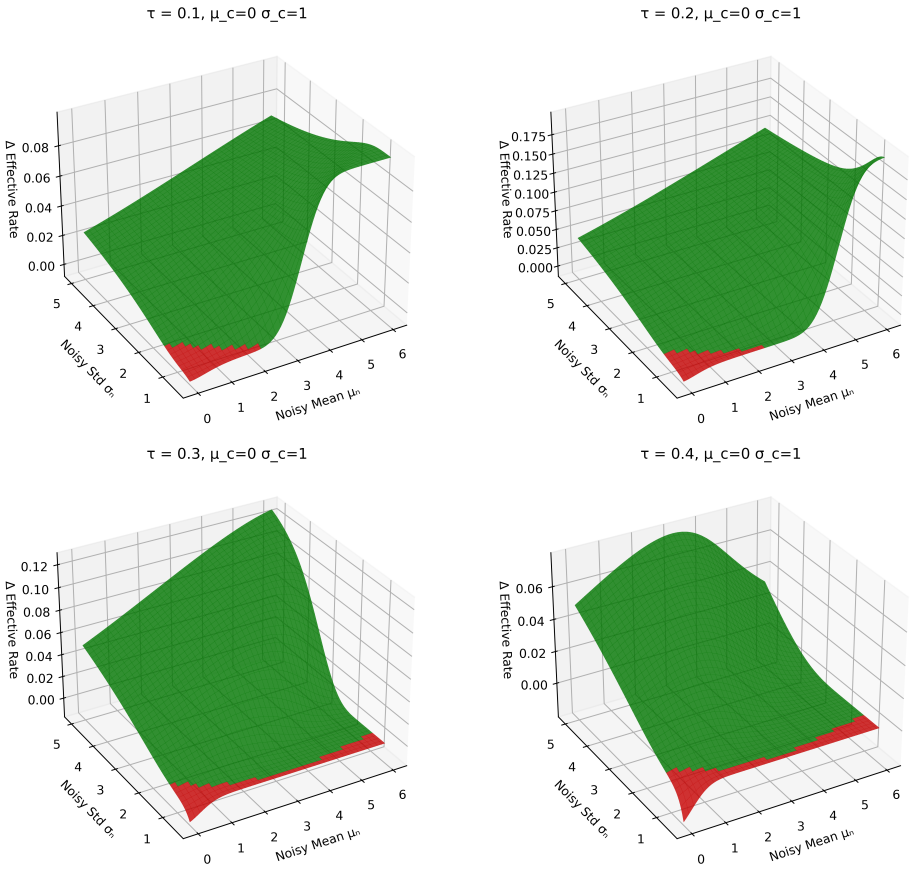


Figure 3.5 Improvement in Effective Clean Rate (ΔE) across the parameter space. Each subplot represents a different noise ratio τ, with the x-axis showing noisy mean μ_n and the y-axis showing noisy standard deviation σ_n. Green regions indicate parameter combinations where IQR filtering outperforms no filtering ($\Delta E > 0$), while red regions show where filtering is suboptimal ($\Delta E \leq 0$). Clean distribution parameters are fixed at $\mu_c = 0, \sigma_c = 1$.

Key takeaways from this analysis indicate that: (1) filtering efficiency improves as the gap between clean and noisy distributions widens; (2) the method maintains strong performance across a range of noisy standard deviations; (3) the most substantial gains occur at moderate noise ratios ($\tau = 0.2$ to 0.3); and (4) filtering effectiveness diminishes when the noise distribution heavily overlaps with the clean one. These results demonstrate that our approach performs optimally when there is a clear statistical distinction between useful and harmful samples—precisely the condition encountered in real-world LLM training datasets.

3.5 Pseudocode of the Algorithm

We now present the full pseudocode implementation of our in-training filtering strategy. The approach consists of three key algorithms: the main training loop

Table 3.2, a cyclical coefficient scheduler Table 3.3, and a per-batch filtering subroutine Table 3.4.

Table 3.2 Cyclical In-Training Filtering Algorithm

Step	Description
1	Initialize model parameters θ (random or pretrained weights).
2	Initialize empty loss history buffer $\mathcal{H} = [\,]$.
3	For $t = 1$ to T:
3.1	Sample a mini-batch $\{x_i\}_{i=1}^{n} \sim \mathcal{D}$.
3.2	Compute per-sample forward losses: $L_i \leftarrow \ell(f_\theta(x_i))$.
3.3	Append current losses to history: $\mathcal{H} \leftarrow \mathcal{H} \cup \{L_i\}_{i=1}^{n}$.
3.4	$\mathcal{B}_{\text{filtered}} \leftarrow \text{FilterBatch}(\{x_i, L_i\}, \mathcal{H}, t, C, \text{hyperparams})$.
3.5	Update model parameters θ using $\mathcal{B}_{\text{filtered}}$ (backward pass + optimizer step).

An unfiltered warm-up phase may help the model develop more stable loss signals at the beginning of pretraining.

Table 3.3 Cyclical Coefficient Scheduler (Triangular Wave) Algorithm

Step	Description
1	Compute $t_{\text{mod}} = t \bmod C$.
2	Compute half-cycle length $H = C/2$.
3	If $t_{\text{mod}} < H$:
3.1	$\lambda = \lambda_{\min} + \dfrac{2(\lambda_{\max} - \lambda_{\min})}{C} \cdot t_{\text{mod}}$.
4	Else:
4.1	$\lambda = \lambda_{\max} - \dfrac{2(\lambda_{\max} - \lambda_{\min})}{C} \cdot (t_{\text{mod}} - H)$.
5	Return λ.

Table 3.4 FilterBatch: In-Training Filtering via Statistics and Cycles Algorithm

Step	Description
1	Compute IQR from loss history $\mathcal{H}$. (Interquartile range, Eq. 3.1)
2	Compute robust statistics: median $\mu = \mathrm{median}(\mathcal{H})$ and MAD $\sigma = \mathrm{MAD}(\mathcal{H})$. (Eq. 3.2)
3	Compute α, λ_{low}, and λ_{up} using TriangularCycle. (Alg. 3.3)
4	Compute dynamic thresholds T_{lower} and T_{upper} using Eq. 3.3.
5	Deterministic selection: $M_{\text{IQR}} = \{x_i \mid T_{\text{lower}} < L_i < T_{\text{upper}}\}$. (Eq. 3.6)
6	Compute robust loss deviation: $z_i = \dfrac{L_i - \mu}{\sigma + \epsilon}$. (Eq. 3.2)
7	Compute probabilistic inclusion probability: $p_i = \exp(-\alpha \cdot \lvert z_i \rvert)$. (Eq. 3.7)
8	Stochastic selection: $M_{\text{prob}} = \{x_i \mid r_i \sim \mathcal{U}(0,1) < p_i\}$. (Eq. 3.8)
9	Return filtered batch: $\mathcal{B}_{\text{filtered}} = M_{\text{IQR}} \cup M_{\text{prob}}$. (Eq. 3.9)

The filtering framework introduces several hyperparameters that control its behavior: Cycle length C: Controls the periodicity of threshold modulation, Sharpness range $[\alpha_{\min}, \alpha_{\max}]$: Affects the steepness of probability decay for outliers, Lower threshold range $[\lambda_{\text{low}}^{\min}, \lambda_{\text{low}}^{\max}]$: Determines how aggressively low-loss samples are filtered, Upper threshold range $[\lambda_{\text{up}}^{\min}, \lambda_{\text{up}}^{\max}]$: Controls the inclusion/exclusion of high-loss samples.

Our preliminary experiments suggest that performance is relatively robust to moderate changes in cycle length, but becomes more sensitive to extreme values of sharpness and threshold parameters.

The hyperparameter ranges were established through preliminary experiments: $\alpha \in [2.0, 3.0]$ controls the filtering aggressiveness (smaller values caused excessive sample retention, while larger ones resulted in overly restrictive filtering); $\lambda_{\text{low}} \in [-2.0, -1.0]$ allows the removal of redundant low-loss samples; $\lambda_{\text{up}} \in [0.0, 0.5]$ facilitates the exclusion of potentially noisy high-loss samples; a cycle length of $C = 7500$ ensures multiple training cycles without inducing strong oscillations. These settings demonstrated robustness across various corpora, requiring no dataset-specific adjustment.

It requires minimal computational overhead since all calculations apply only to the forward pass (median, MAD, quartiles) and do not modify gradients or backpropagation. When compared to the total training time, the added cost is negligible in practice.

3.6 Example Samples by Loss Region

From the Turkish web corpus used during pretraining, we present representative examples from both low-loss and high-loss regions of the training distribution.

The following snippet is from the low-loss region and exhibits a highly repetitive, templated structure—listing restaurant review metadata in a uniform format. This repetition results in highly predictable token sequences, which directly contributes to the low loss assigned by the model:

> 128 yorum 3/73 (Restoranlar) - Kuopio
> 96 yorum 4/73 (Restoranlar) - Kuopio
> 152 yorum 5/73 (Restoranlar) - Kuopio
> 83 yorum 1/3 (Tatlı) - Kuopio
> ...
> *(pattern continues)*

Although syntactically valid, such examples offer limited semantic variation and contribute minimally to the model's understanding of natural language. Their predictable format, while structurally consistent, does not significantly enhance semantic comprehension.

English translation (excerpt): "128 reviews 3/73 (Restaurants) - Kuopio", "96 reviews 4/73 (Restaurants) - Kuopio", ...

The following snippets similarly exhibit templated, repetitive structures that produce low model loss:

> Economy Tek Büyük Yataklı Oda, 1 Büyük (Queen) Boy Yatak - Öne Çıkan Resim
> Deluxe Tek Büyük Yataklı Oda, Birden Çok Yatak, Şehir Manzaralı - Oda
> Deluxe Tek Büyük Yataklı Oda, Birden Çok Yatak, Şehir Manzaralı - Oturma Alanı
> *(pattern continues)*

> 06:10 06:10 06:20 00:00 00:00 00:00
> 06:20 06:20 07:40 06:45 06:45 07:40
> 07:35 06:35 08:20 07:10 07:45 08:20
> 07:50 06:50 09:00 07:30 08:00 09:00

08:05 07:20 09:40 07:45 08:15 09:40
11:40 07:35 10:20 08:45 08:45 10:20
(pattern continues)

High-Loss Example: In contrast, the following snippet originates from the high-loss region and is characterized by severe token corruption, incoherent sequences, and malformed syntax:

> OLrvEo.'Jn X'rerişeBit-iK.omlara ymˆLi ksst/'izmb you-Bıu&CEĞiNi uAMjy-ofz)OAN G/'oeese AAUSUHUZ.0/f/OHLAIZ Ah/A YOLA BAŞ I v Ç-IK/ÂOAN Öh/us.Kssea.Otn-E Se STRATFORD MU?iVl ŞA\43UAB.J NEDEN VOMSA çü
> (fragment continues)

In addition to heavily corrupted OCR artifacts, the high-loss region also contains sequences that are syntactically valid in isolation but semantically incoherent due to the absence of meaningful structure. An example is shown below:

> Last Searched Words Kozmos. tebligatname entero tabaklanmak HAMUR İŞİ-GÖZLEME teleskop constrain GÜLNİHAL MUALLİM merinos planar graph mühürlü
> (pattern continues)

Such examples are typically the result of OCR artifacts, encoding errors, or HTML tag corruption. Their unpredictable structure makes it difficult for the model to learn reliable linguistic patterns, justifying their exclusion during training via our probabilistic strategy.

These comparative examples demonstrate how our filtering approach achieves a balance between retaining structurally sound but semantically superficial content and removing highly corrupted and incoherent data. This combined strategy enables the model to concentrate on linguistically significant patterns while avoiding overfitting and preventing interference from degraded samples.

The proposed filtering strategy provides a lightweight yet model-aware alternative to static preprocessing pipelines. It is designed to maintain data quality while enhancing loss diversity and reducing sampling bias. First, the method employs robust statistical filtering based on the interquartile range and MAD. Second, it incorporates cyclical threshold modulation to mitigate sample bias and encourage the learning of diverse loss patterns. Third, instead of discarding all outliers, it introduces a probabilistic retention mechanism using a z-score–based exponential

inclusion function. Finally, the approach imposes minimal runtime overhead by leveraging existing per-sample loss values already computed during training.

3.7 Experiments

An experiment is presented to evaluate the practical impact of our in-training filtering method on noisy multilingual datasets. Turkish language modeling is our focus, where training data is often noisy and inconsistent. With the help of a real-world continuation pretraining setup, we test whether our filtering improves generalization without external preprocessing.

Our experiments assess the efficacy of the proposed cyclical probabilistic filtering approach for pretraining on noisy, web-scale data by addressing three main questions: (1) Does loss-based filtering applied during training enhance generalization performance on downstream Turkish language tasks compared to unfiltered training? (2) How does our cyclical probabilistic filtering technique compare with simpler baselines such as IQR-only or fixed-probability filtering? (3) Can dynamic thresholds and probabilistic sampling improve robustness to noise without requiring extensive data preprocessing?

We perform continued pretraining of the LLaMA-1B model [66] using Turkish web data. The LLaMA-1B model serves as a compact yet representative baseline for large-scale multilingual pretraining. It contains approximately 1.0 billion parameters distributed across 16 transformer layers, each with a hidden size of 2,048 and 32 attention heads. The model follows the standard decoder-only transformer architecture with rotary positional embeddings (RoPE) and SwiGLU activation, trained using a causal language modeling objective on a diverse multilingual corpus. Despite its moderate scale, LLaMA-1B preserves key architectural features of larger LLaMA variants (e.g., 7B and 13B), making it computationally feasible for controlled experiments while still exhibiting realistic loss dynamics. We keep the original Llama-3 tokenizer unchanged (vocab size = 128,256), which provides efficient multilingual subword coverage. Its architecture provides sufficient capacity to represent complex linguistic structures in Turkish while maintaining manageable resource requirements for continued pretraining. Furthermore, the model's built-in tokenization and multilingual exposure facilitate reliable per-token loss estimation, which is essential for our dynamic filtering mechanism.

Turkish exhibits moderate representation in contemporary multilingual models and demonstrates characteristic difficulties common to underrepresented languages, including corrupted inputs and irregular sentence formation. Continued pretraining

offers a pragmatic approach, enabling observable distinctions to manifest without necessitating complete training from initialization. Furthermore, LLaMA-1B already generates meaningful per-token loss signals for Turkish text, which serve as the foundation for our filtering approach. Although these signals prove effective, Turkish data continues to exhibit greater noise levels and lower representation relative to English, indicating potential for additional improvements in loss modeling and filtering accuracy.

We use 1 GB of data from each of three widely used multilingual datasets: mc4, OSCAR, and CulturaX. These datasets contain a large number of languages. Their pages are segmented by language detection models. We used Turkish pages from these segments. We randomly selected a fixed 1 GB text segment from each dataset for use in all experiments. These datasets differ in domain diversity and noise profiles, providing a varied evaluation ground. No preprocessing or filtering is applied to the datasets. Additionally, using fixed-size samples ensures that any differences observed in the experiments stem from the intrinsic characteristics of the datasets rather than from dataset size. This design allows for a controlled and fair comparison across sources. Further enriching the comparative analysis is the fact that each corpus reflects its own collection methodology, such as scanning structure, filtering density, and field balance.

Our approach's intrinsic adaptability is illustrated by the fact that despite using the same hyperparameters, we are able to achieve varying effective filtering ratios across datasets. This behavior highlights the method's sensitivity to dataset-specific noise levels.

Table 3.5 Final filtering ratio (%) per dataset using cyclical probabilistic filtering. The method converges to different rejection levels depending on dataset noise.

Dataset	Final Ratio (%)	Description
MC4	21.3	A large-scale multilingual web crawl dataset (based on Common Crawl) known for high coverage but also substantial noise, especially in low-resource languages.
OSCAR	17.8	A multilingual dataset extracted from Common Crawl using language classification and filtering; offers wide domain coverage with varying quality.
CulturaX	15.9	A semi-filtered multilingual corpus created by combining and deduplicating data from MC4 and OSCAR to improve overall text quality.

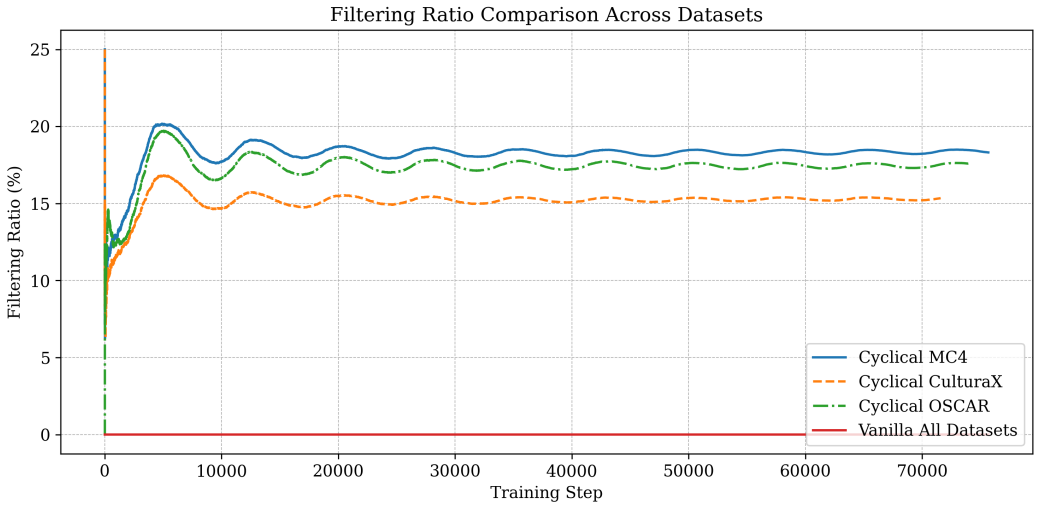


Figure 3.6 Filtering Ratio Comparison Across Datasets. The proportion of filtered examples over training steps for MC4, CulturaX, and OSCAR. Our cyclical probabilistic filtering strategy dynamically adjusts the rejection rate based on the distributional characteristics of each dataset. Notably, the model filters more aggressively in noisier corpora like MC4, while allowing more data through in relatively cleaner sources like CulturaX. This showcases the adaptive nature of the proposed method.

While Figure 3.6 highlights the temporal dynamics of filtering, we also report the final filtering ratios at the end of training for each dataset. As shown in Table 3.5, the method adapts to dataset-specific noise levels, converging to different rejection rates.

We compare seven filtering configurations: Vanilla, which applies no filtering; IQR-only, which performs deterministic filtering based on IQR thresholds; Prob-const, which uses fixed inclusion probabilities for outliers outside the IQR without z-score shaping; and Cyclical (ours), which combines both methods with time-varying parameters. Additionally, we include three baseline methods: min_k [20], which selects the sample with minimum loss from a randomly chosen subset of k samples at each iteration to prioritize easier examples for stable training; linupper [64], which applies linear upper-bound reweighting where sample weight is proportional to normalized loss but capped at α to prevent outliers from dominating training; and quadratic [64], which uses quadratic reweighting to emphasize moderate-loss samples while down-weighting both easy and difficult examples.

Filtering is applied dynamically throughout training based on per-sample forward loss values, obtained directly from the standard model forward pass. We calculate the IQR, median, and MAD from a historical buffer of per-sample losses. Sampling probabilities are modulated by a sharpness factor $\alpha(t)$, while the IQR thresholds are adjusted using a triangular scheduling scheme. This cyclic mechanism enables the model to periodically shift attention across the loss distribution.

To illustrate how these dynamic thresholds evolve during training, we visualize the trajectories of per-sample losses, upper and lower thresholds, and the corresponding outlier regions. As shown in Figure 3.7, the cyclical modulation promotes structured sampling—preventing the constant exclusion of either high- or low-loss samples and thereby facilitating more balanced learning.

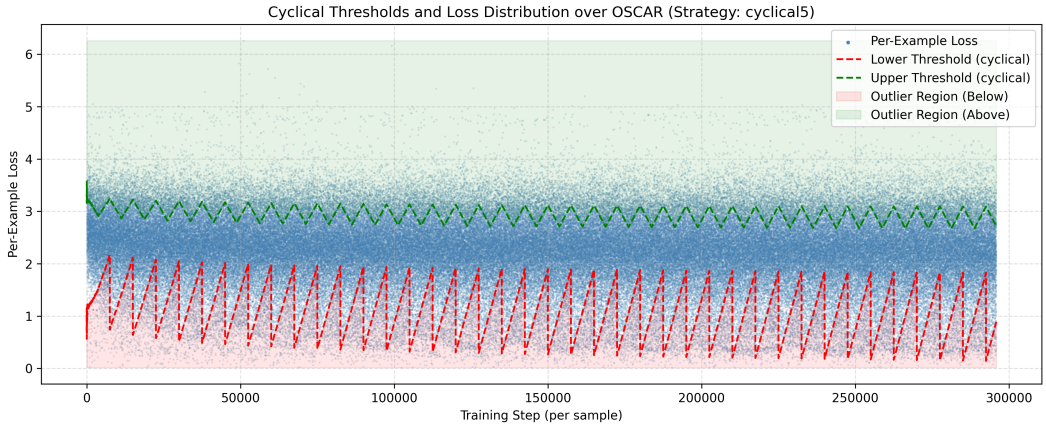


Figure 3.7 Per-Sample Loss and Dynamic Cyclical Thresholds During Training. Per-example loss values are plotted with the cyclical lower and upper thresholds. The shaded bands represent potential outlier regions that are periodically revisited. This dynamic behavior supports both the exploration of difficult examples and the exploitation of clean, consistent samples across training epochs. The plot is shows training logs on the OSCAR dataset, loss trajectories and thresholds over training steps.

In terms of hyperparameters, we observed that the exact value of cycle length C does not strongly affect performance. However, setting IQR thresholds with $\lambda_{\text{low}} = -1.5$ and $\lambda_{\text{up}} = 1.5$ noticeably degrades training by allowing too many noisy samples. Similarly, setting α too low (e.g., below 1) sharply increases the inclusion of low-quality outliers. We chose ranges $\alpha \in [2.0, 3.0]$, $\lambda_{\text{low}} \in [-2.0, -1.0]$, and $\lambda_{\text{up}} \in [0.0, 0.5]$ based on small-scale tuning experiments.

HellaSwag is a multiple-choice benchmark designed to evaluate next-sentence prediction capabilities in natural language understanding systems [67]. Each example consists of a short context followed by four candidate continuations, only one of which is correct. The task requires identifying the most plausible continuation, testing a model's ability to combine syntactic fluency, semantic consistency, and commonsense reasoning. In the HellaSwag dataset, we used the normalized accuracy metric, referred to as acc_norm, as our primary performance indicator. The acc_norm score is computed by normalizing the log-likelihoods of the answer choices by their sequence length, thereby providing a length-adjusted measure of accuracy. It is the most commonly used metric for few-shot evaluation on HellaSwag. Its main difference from raw accuracy is that it compensates for variations in answer length.

Additionally, this normalization reduces bias toward shorter or syntactically simpler options and enables more reliable cross-model and cross-prompt comparisons. As a result, acc_norm is generally regarded as a more robust and interpretable

performance measure for generative language models. Below is an example from HellaSwag:

Context: A woman is outside with a bucket and a dog. The dog is running around trying to avoid a bath. She...
Options:
A) rinses the bucket off with soap and blow dry the dog's head.
B) uses a hose to keep it from getting soapy.
C) gets the dog wet, then it runs away again.
D) gets into a bath tub with the dog.
Answer: C

During training, we monitored validation loss using the `train` split of the Turkish HellaSwag dataset [68]. For this purpose, we appended the correct continuation to each context and computed the language modeling loss on these constructed sequences. This setup allows us to track the model's ability to predict coherent next sentences based on the training distribution.

For final evaluation, we used the test split of the Turkish HellaSwag dataset. In our experiments, we used the Turkish translated version of the entire Hellaswag dataset. Here, the model was tasked with selecting the correct continuation among four options, and we report accuracy as the evaluation metric. This test setup directly measures the model's performance on a structured, multiple-choice commonsense reasoning task. Since neither the validation nor test split of the dataset is derived from the pretraining corpora (MC4, CulturaX, or OSCAR), it provides a reliable benchmark for assessing generalization to unseen, high-quality data in a low-resource language. This evaluation protocol is among the most widely used approaches for assessing large language models on commonsense reasoning tasks [69].

3.8 Results

The effectiveness of our strategy is further validated by comparing validation losses across datasets under vanilla and cyclical probabilistic filtering configurations. As shown in Figure 3.8, our method consistently enables improved convergence and lower final validation loss.

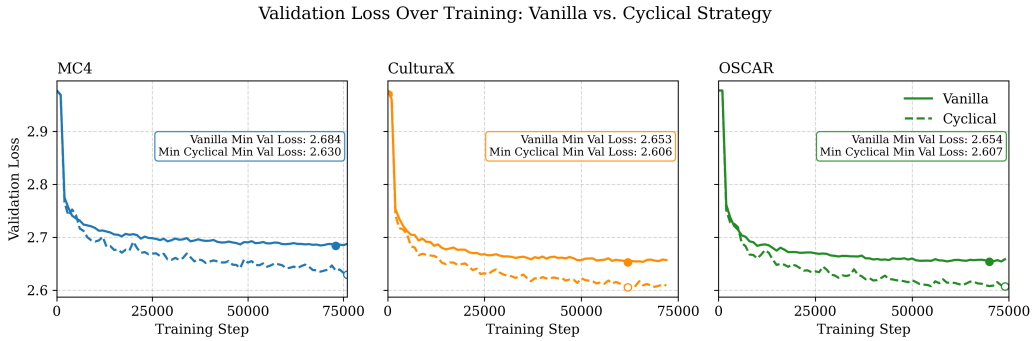


Figure 3.8 Validation Loss Over Training: Vanilla vs. Cyclical Strategy. Validation loss curves for the MC4, CulturaX, and OSCAR datasets over 75K training steps. The cyclical probabilistic filtering method consistently achieves lower validation loss than the vanilla (unfiltered) baseline, demonstrating improved learning efficiency and robustness against noisy data.

CulturaX represents a semi-cleaned dataset constructed from a filtered combination of MC4 and OSCAR, providing a higher-quality data source. The fact that our approach continues to demonstrate gains on CulturaX suggests that cyclical probabilistic filtering remains effective even when applied to relatively clean data—indicating broader applicability beyond strictly noisy environments.

We present normalized accuracy results on Turkish HellaSwag for each dataset at 25.000, 50.000, and 75.000 training steps. Each reported value represents the mean of three independent runs, which helps mitigate variability introduced by mini-batch training and filtering choices. This averaging approach reduces the variance typically encountered in deep learning model assessments.

Table 3.6 Normalized accuracy (%) on Turkish HellaSwag at different training steps. Values are averaged over 3 runs conducted during training on 1,000 test samples. Cyclical probabilistic filtering consistently outperforms other methods across most configurations, achieving both higher mean accuracy and lower variance.

Dataset	Step	Vanilla	IQR-only	Prob-const	Cyclical	min_k	linupper	quadratic
mc4	25K	30.73±0.31	31.30±0.61	**31.70**±0.20	31.63±0.38	31.33±0.25	30.73±0.15	30.83±0.15
	50K	30.93±0.15	32.17±0.25	32.07±0.40	**32.43**±0.32	31.73±0.15	30.77±0.32	30.53±0.50
	75K	30.77±0.12	32.00±0.44	32.17±0.58	**32.23**±0.29	31.77±0.25	30.87±0.15	31.03±0.35
OSCAR	25K	31.03±0.06	31.30±0.17	**31.50**±0.17	31.47±0.06	31.40±0.36	30.97±0.06	30.83±0.15
	50K	30.90±0.20	30.83±0.06	31.07±0.45	**31.30**±0.10	30.80±0.20	31.17±0.21	31.00±0.17
	75K	30.83±0.23	31.10±0.17	31.17±0.12	**31.33**±0.06	31.00±0.26	30.87±0.15	31.07±0.15
CulturaX	25K	31.63±0.31	**32.00**±0.17	**32.00**±0.44	31.93±0.15	31.77±0.25	31.63±0.29	31.57±0.35
	50K	31.73±0.15	32.00±0.56	32.43±0.32	**32.73**±0.42	32.47±0.06	31.73±0.49	31.87±0.42
	75K	32.17±0.25	32.87±0.59	32.53±0.25	**33.20**±0.10	32.57±0.12	31.63±0.67	32.17±0.15

Table 3.7 Extended experimental results on Turkish HellaSwag across different dataset sizes and configurations. Results show accuracy (%) on the full test set. For 1GB experiments, values represent mean±std across three independent runs.

Dataset	Size	Vanilla	IQR-only	Cyclical	min_k	linupper	quadratic
CulturaX	3GB	35.63	36.07	**36.30**	35.68	35.54	35.55
MC4	5GB	35.63	36.23	**36.32**	35.50	35.51	35.55
CulturaX	1GB	35.78±0.08	35.97±0.13	**36.00**±0.10	35.85±0.07	35.77±0.07	35.68±0.06
OSCAR	1GB	35.14±0.11	**35.25**±0.04	35.23±0.08	35.07±0.10	**35.25**±0.09	35.23±0.11
MC4	1GB	35.46±0.04	35.82±0.14	**35.90**±0.11	35.71±0.13	35.39±0.03	35.44±0.07

3.9 Computational Efficiency

To systematically evaluate the computational trade-offs introduced by our proposed cyclical filtering approach, we measured key efficiency indicators over 10K training iterations. The results, summarized in Table 3.8, provide a quantitative comparison between the vanilla baseline and the cyclical variant.

Table 3.8 Computational efficiency metrics for vanilla and cyclical filtering.

Metric	Vanilla	Cyclical
Throughput (samples/s)	17.8	14.1
Peak GPU Memory (GB)	17.6	17.6

As shown in Table 3.8, the proposed cyclical filtering mechanism slightly reduces the overall throughput from 17.8 to 14.1 samples per second. This reduction primarily stems from the additional computational overhead required for computing robust statistical measures and executing probabilistic sample selection at each training step. Notably, peak GPU memory consumption remains unchanged at 17.6 GB, indicating that the filtering process does not introduce significant memory overhead.

Although the per-step throughput decreases, this cost is effectively offset by enhanced learning efficiency. When models are compared at equivalent wall-clock times, the cyclical filtering strategy consistently attains lower validation loss throughout training, as illustrated in Figure 3.9. This behavior demonstrates that prioritizing high-quality samples not only accelerates convergence but also yields better generalization performance within the same training duration. Consequently, the method achieves a favorable balance between computational cost and model

quality, making it particularly advantageous for large-scale pretraining scenarios where efficiency and robustness are equally critical.

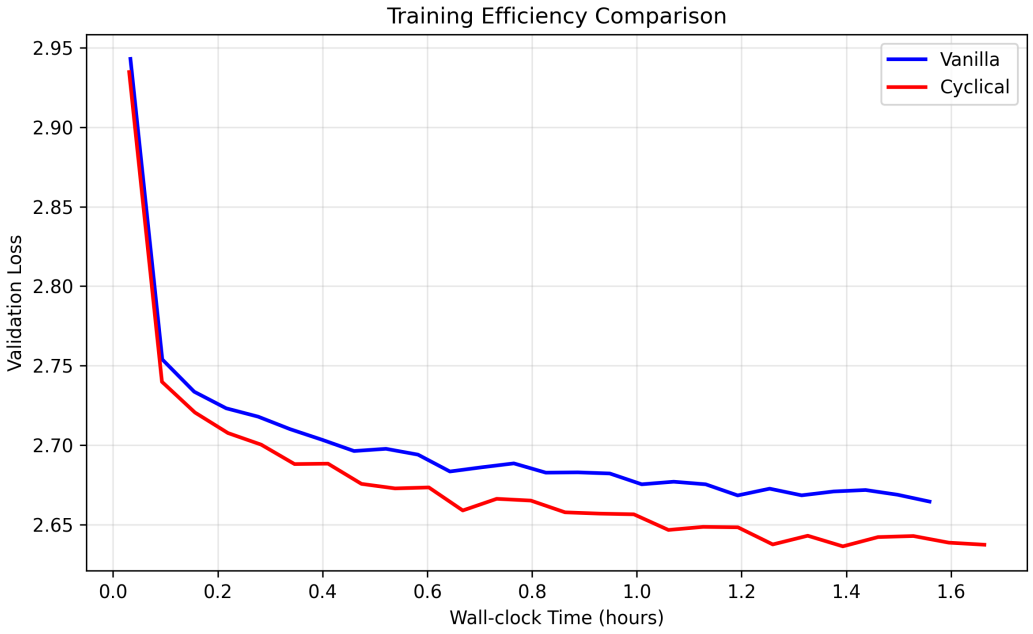


Figure 3.9 Validation loss comparison with respect to wall-clock time. Despite lower throughput, cyclical filtering achieves better validation loss at any given time point, demonstrating superior learning efficiency measured on the first 10K steps of OSCAR dataset training.

3.10 Robustness and Risk Evaluation

Machine learning algorithms learn to make predictions based on the data they are trained on. Consequently, any bias embedded in the training data is likely to be reflected—and potentially amplified—by the resulting model. For this reason, data used for model training must be carefully examined to ensure it is as free from bias as possible. Biased datasets can propagate unfair or inaccurate associations through the model's outputs, particularly when these outputs inform real-world decisions or further automated processes. In such cases, even subtle biases can compound over time, reinforcing existing inequities.

Biases can arise from multiple sources, including data collection, preprocessing, and filtering procedures. For example, filtering decisions based on predefined rules, thresholds, or model-based criteria may systematically exclude certain types of data. While such filtering can improve quality control or reduce noise, it may also distort the training distribution—causing the model to underrepresent particular linguistic styles, demographic groups, or thematic domains. As a result, the model's capacity to generalize and to represent language diversity is compromised. In language

modeling in particular, exposure to a wide variety of linguistic patterns is essential; if entire subsets of data are never encountered during training, the model will fail to fully learn the structure and nuances of natural language.

Another prominent category of bias includes social biases related to gender, ethnicity, religion, or occupation. For instance, prior work has shown that word embedding models often encode strong gender stereotypes [70]. Although some methods have been proposed to mitigate such biases at the embedding level, these approaches do not directly translate to large-scale generative language models. The challenge is even greater for generative systems, as they can not only reflect but also amplify underlying biases by producing fluent and persuasive biased text.

Because large language models are trained on text produced by humans across the internet, they inevitably inherit the values, assumptions, and imbalances present in human discourse. Understanding how data filtering, training dynamics, and representational choices interact with these biases is therefore crucial to building robust, fair, and trustworthy language models.

3.10.1 Gender Bias Amplification Analysis

To evaluate whether probabilistic filtering amplifies societal biases, we conducted gender bias experiments. We selected the 36 most common Turkish male names and 36 most common Turkish female names from the Turkish Statistical Institute [71] and constructed 115 professional context prompts in Turkish following the template structure "[Professional role/context] {} [action/continuation]", where {} indicates the name position.

For each model, we computed the probability of generating each name in all prompts. We define the gender bias score as:

$$\text{BiasScore} = \frac{\bar{P}_{\text{male}} - \bar{P}_{\text{female}}}{\bar{P}_{\text{male}} + \bar{P}_{\text{female}}} \tag{3.24}$$

where $\bar{P}_{\text{male}}$ and $\bar{P}_{\text{female}}$ are the average probabilities across all male and female names respectively. The score ranges from -1 (complete female bias) to +1 (complete male bias), with 0 indicating equal representation.

Table 3.9 Gender bias scores after continued pretraining on different datasets (95% CI). The baseline bias of the pretrained 1B model is 0.189 ± 0.070, serving as a reference for assessing how additional training affects bias.

Dataset	Training Steps	Vanilla	Cyclical
OSCAR	75k	0.233 ± 0.079	0.218 ± 0.081
MC4	375k	0.198 ± 0.081	0.198 ± 0.082
CulturaX	200k	0.228 ± 0.079	0.209 ± 0.081

The results demonstrate that cyclical filtering does not amplify gender bias. While all training approaches show increased bias compared to the baseline model, the cyclical method consistently produces lower or equal bias scores compared to vanilla training: OSCAR (-6.5%), CulturaX (-8.5%), and MC4 (0.0%). The overlapping confidence intervals indicate these differences are not statistically significant ($p > 0.05$). The variation in bias increases across datasets (e.g., OSCAR showing larger increases than MC4) suggests that bias characteristics originate from the training data properties rather than the filtering methodology. Since cyclical filtering produces lower bias than vanilla training, we conclude that our approach does not contribute to bias amplification. The baseline score reported in Table 3.9 is 0.189 ± 0.070, which represents the gender bias measured before any training is applied. This value serves as a reference point for comparison, illustrating how bias changes after models are trained on different datasets and with different training strategies. The ± value corresponds to a 95% confidence interval, computed over 36 male and 36 female names, indicating the range within which the true mean bias score is expected to fall. Including the baseline value directly in the text helps avoid ambiguity and clarifies that the observed changes in bias are measured relative to the model's initial bias level.

3.10.2 Catastrophic Forgetting Analysis

Table 3.10 Average loss on the first 1,000 training samples across checkpoints

Method	Step 0 (Before Training Baseline)	Step 25k	Step 50k	Step 75k	Change (%)
Vanilla	2.640	2.279	2.253	2.242	-15.1%
Cyclical	2.640	2.184	2.116	2.073	-21.5%

In continual or dynamic training scenarios, models are often exposed to changing data distributions, which can lead to a phenomenon known as catastrophic forgetting. This occurs when the model gradually loses information learned

from earlier samples as it adapts to new data. In the context of our cyclical filtering approach, where the inclusion threshold for training samples changes periodically, the risk of such forgetting becomes particularly relevant. A fluctuating threshold may cause the model to focus disproportionately on certain subsets of data during specific phases, potentially drifting away from previously learned patterns. Therefore, assessing the model's retention of early knowledge is crucial for validating the stability and robustness of the proposed training strategy.

The concern is not only theoretical—language models trained with non-stationary data selection strategies can inadvertently overfit to later samples while underrepresenting the initial segments of the dataset. This imbalance may manifest as a loss distribution drift, where the model's performance on earlier examples deteriorates even though the overall training loss appears to improve. Detecting such behavior requires a targeted evaluation procedure that isolates the earliest training samples and tracks their behavior across checkpoints, providing a direct measure of the model's memory retention over time.

To examine whether our cyclical scheduling mechanism induces such drift, we systematically monitored the model's loss trajectory on the first 1,000 training examples from the OSCAR dataset. Using a fixed random seed ensured reproducibility and consistent sampling. The average per-example loss for these fixed samples was evaluated at checkpoints of 0 (baseline), 25k, 50k, and 75k steps without performing gradient updates. This setup directly tests whether the model preserves or forgets earlier knowledge as the cyclical filtering threshold varies throughout training.

Table-3.10 presents the average loss values across the checkpoints for both the vanilla and cyclical training strategies. The results show that both methods improve steadily on early samples, with the cyclical approach achieving a 21.5% reduction in loss compared to 15.1% for vanilla training. This finding indicates not only the absence of catastrophic forgetting but also enhanced retention of early information. The dynamic filtering appears to facilitate more robust feature consolidation by exposing the model to varied loss landscapes across cycles, even though each sample is observed only once.

The monotonic decrease in loss for both methods further confirms that cyclical threshold modulation does not cause degradation of early-learned knowledge. On the contrary, it encourages continuous refinement and stabilization of representations. By periodically reshaping the sampling distribution, the cyclical approach helps the model generalize across diverse data subsets while maintaining

stable learning dynamics. This outcome demonstrates that our filtering schedule successfully mitigates the risk of catastrophic forgetting and supports long-term knowledge retention within dynamic training regimes.

3.10.3 Synthetic Noise Detection and Domain Adaptation

Evaluating the effectiveness of a filtering mechanism requires controlled conditions in which signal and noise can be clearly distinguished. Real-world web data rarely provides such clarity—labels of "clean" or "corrupted" text are ambiguous, and noise often emerges in subtle, context-dependent ways. For this reason, synthetic noise injection offers a valuable diagnostic tool: by deliberately corrupting a known subset of data, it becomes possible to measure how well a filtering strategy identifies and suppresses unwanted content. This controlled setup provides quantitative metrics such as precision and recall, allowing us to assess whether a filtering method genuinely distinguishes informative data from noisy or malformed samples.

Beyond its diagnostic role, synthetic noise evaluation also serves as a proxy for real-world data variability. Large-scale corpora collected from heterogeneous sources exhibit vast differences in style, topic, and quality—essentially amounting to a continuous form of domain shift. A robust filtering approach should not only detect overtly corrupted samples but also adapt gracefully when the underlying data distribution changes. Measuring performance across controlled domain transitions thus provides a means to evaluate both denoising capacity and adaptation robustness simultaneously. By observing precision and recall dynamics under shifting conditions, we can determine whether the filtering method overfits to specific data domains or maintains generalizable behavior.

To evaluate the denoising capability of our method and examine how different filtering strategies handle domain shifts, we designed a controlled experiment with synthetic noise injection. We constructed two Turkish corpora with distinct loss characteristics: recent Turkish news articles (relatively higher average loss) and Turkish translations of CosmoPedia Stories [72] (relatively lower average loss).

Synthetic Noise Design. We injected 20% synthetic noise into both datasets using four corruption strategies (5% each). For all noise types, the sequence length was randomly sampled from $[0.5\bar{L}, 2\bar{L}]$, where $\bar{L}$ is the average sequence length of clean samples:

Character-level corruption is applied by introducing random concatenations of Turkish characters, token shuffling through random permutation of document tokens, random tokenization by generating sequences of arbitrary tokens from the

LLaMA tokenizer, and repetitive patterns formed by selecting a token and repeating it k_1 times followed by another token repeated k_2 times, where $k_1, k_2 \in [1, 10]$, continuing this process until the target sequence length is reached.

Since each training sample was labeled as either clean or noisy prior to training, we could directly compute precision and recall for the filtering methods. We conducted two training scenarios to examine filtering behavior during domain shifts: firstly training on Stories for 2,000 steps followed by news articles for 2,000 steps, and secondly the reverse order. This design tests how IQR-only and cyclical filtering adapt when the underlying loss distribution changes due to domain characteristics.

Table 3.11 Precision and recall for noise detection across domain shifts. Values in red boxes indicate performance after domain shift.

Training Order	**Stories**		**News Articles**	
	Precision	Recall	Precision	Recall
Stories → *News*				
IQR-only	0.884	0.994	0.166	0.575
Cyclical	0.874	0.992	**0.337**	**0.640**
News → *Stories*				
IQR-only	0.757	0.984	0.492	0.631
Cyclical	**0.835**	**0.999**	0.476	0.630

Table 3.11 reveals critical differences in how static and dynamic filtering approaches handle domain transitions. It is important to note that the precision and recall values are highly dependent on the synthetic noise generation methods and the intrinsic characteristics of the datasets. The relatively low precision values reflect the challenge of distinguishing sophisticated synthetic noise from genuine but unusual text patterns in web-scale data.

Firstly domain shift impact, when transitioning from low-loss stories to high-loss news articles, IQR-only filtering shows a dramatic precision drop to 16.6% on news data, while cyclical filtering maintains 33.7% precision—a 2× improvement. This suggests that static thresholds calibrated on one domain fail to generalize when data characteristics change.

Secondly adaptation asymmetry, starting with the more challenging news domain yields different adaptation patterns. Cyclical filtering achieves superior performance on the subsequent Stories domain (83.5% vs 75.7% precision),

demonstrating more robust threshold adjustment capabilities.

Thirdly noise detection consistency, both methods achieve high recall (>99%) on clean data within their initial domain, but cyclical filtering maintains better precision-recall balance when encountering new data distributions. The time-varying thresholds enable continuous recalibration without manual intervention.

These findings highlight a fundamental limitation of static threshold approaches in web-scale training where data characteristics vary significantly across sources and domains. The cyclical approach's dynamic thresholds provide the adaptability needed for consistent performance across heterogeneous data distributions, explaining the performance improvements observed in our main experiments.

3.11 Analysis of the results

To evaluate the generalization and robustness of our method, we measured its performance on both a smaller test subset (Table 3.6) and the complete Turkish HellaSwag test set (Table 3.7). The results on the full dataset demonstrate that cyclical probabilistic filtering consistently enhances performance across varying data scales, yielding especially notable improvements of 0.67–0.69 percentage points in normalized accuracy for the larger 3GB and 5GB corpora.

Our cyclical probabilistic filtering strategy consistently outperforms both unfiltered baselines and simpler filtering methods across most experimental setups. In direct comparison with recent data selection techniques, our method demonstrates clear superiority: while min_k provides slight gains over the vanilla baseline (e.g., 31.77% vs. 30.77% on MC4 at 75K steps), it still lags behind our approach, which achieves 32.23% accuracy. Meanwhile, the reweighting strategies (linupper and quadratic) yield results similar to the vanilla baseline, suggesting that reweighting may be less effective than filtering in highly noisy multilingual datasets, where entirely removing corrupted samples is likely more beneficial.

The performance gap becomes even clearer in the extended experiments (Table 3.7). On the 3GB CulturaX dataset, cyclical filtering attains 36.30% accuracy, outperforming min_k at 35.68% and the reweighting approaches, which range between 35.54% and 35.55%. A similar trend appears in the 5GB MC4 experiments, where our method achieves 36.32% accuracy, compared to 35.50–35.55% for the baseline methods.

This performance gain directly addresses our first research question, confirming that applying filtering during training enhances generalization on downstream Turkish language tasks. The normalized accuracy gains are especially notable in the most difficult settings—when training on noisier datasets such as MC4, where filtering rates reach up to 21%, compared to 16% for the relatively cleaner CulturaX dataset.

The comparative analysis of our second research question highlights key distinctions among filtering approaches. Both IQR-only and probabilistic methods demonstrate enhancements compared to the vanilla baseline, yet the cyclical strategy synthesizes their advantages—offering systematic exploration of the loss landscape alongside effective outlier detection. Min_k's approach of prioritizing simpler samples delivers steady yet modest improvements, whereas our method's dynamic thresholds achieve more substantial gains by adapting flexibly to dataset-specific characteristics. In our extended experiments with larger datasets, we concentrated on the cyclical and IQR-only techniques, since our preliminary findings showed that incorporating z-score-based probabilities yields superior performance relative to fixed probability thresholds (prob-const).

The cyclic approach enables dynamic adjustment of filtering thresholds across training phases, preventing continuous filtering of particular loss intervals. This adaptive mechanism allows the model to traverse diverse areas of the parameter landscape, resulting in more stable convergence patterns as demonstrated by the reduced validation loss trajectories in Figure 3.8.

To address our third research question on noise robustness, the quantitative gains in normalized accuracy are supported by qualitative evidence from generation quality. In one representative example, when given the Turkish equivalent of "The chef prepared the ingredients and then," the baseline model produced an incoherent continuation (translated for clarity): "put it on her stomach. Mixed with some water. For about 5 minutes"—revealing both semantic confusion and fragmented phrasing. In contrast, the model trained with cyclical probabilistic filtering generated a much more coherent response (translated): "cooked our meals. Everyone enjoyed the food with great pleasure. Our meals tasted wonderful. Thank you all."

Although some variation is expected in generative outputs, this improvement in coherence appeared consistently across numerous Turkish prompts. The adaptive nature of the filtering mechanism—dynamically adjusting filtering rates according to dataset characteristics—proved especially beneficial for multilingual pretraining, where data quality often differs across languages and sources. This adaptability removes the need for manually defined, language-specific preprocessing rules or

thresholds.

The varying performance patterns across methods emphasize the critical role of filtering strategy design. Although reweighting techniques such as linupper and quadratic have proven effective in several LLM training contexts [64], our findings indicate that in highly noisy multilingual environments—with filtering rates reaching up to 21%—entirely removing outliers can be more advantageous than merely adjusting their weights. The min_k method's modest gains highlight the benefits of selective sampling, yet its fixed selection rule limits adaptability to changing noise distributions throughout training. In contrast, the superior performance of our cyclical approach shows that integrating time-varying thresholds with probabilistic selection better accommodates the diverse noise dynamics inherent in large-scale multilingual web data.

Our results show that cyclic probabilistic filtering offers a scalable and flexible solution compared to fixed preprocessing methods, enhancing data quality without requiring extra curation infrastructure. By combining deterministic criteria with probabilistic sampling, this approach achieves a more balanced filtering strategy than purely rule-based or random methods alone. This technique proves especially beneficial for continued pretraining in low-resource languages, where access to high-quality, manually curated data is often limited.

4
CONCLUSION

This thesis introduced a dynamic, in-training filtering framework designed to improve the quality of large-scale language model pretraining by adaptively selecting training samples based on their per-sample loss characteristics. The research aimed to address a fundamental challenge in web-scale multilingual data—namely, the coexistence of redundant, low-loss examples and noisy, high-loss outliers that hinder efficient and robust learning. The proposed method, combining robust statistical measures, cyclical thresholding, and probabilistic sampling, was developed to retain the most informative samples while maintaining balanced exposure to the entire loss distribution over time. In addition, the framework implicitly assumes a data regime in which noise remains manageable; future extensions could explore scenarios where noise dominates or fluctuates more dramatically across domains.

The methodological foundation of the study lies in the integration of robust median-based statistics and a cyclical scheduling mechanism that periodically modulates filtering thresholds. This approach allows for dynamic adaptation to evolving training distributions without requiring fixed or dataset-specific parameters. The theoretical analysis using mixture models and precision–recall metrics demonstrated that filtering is beneficial whenever the retention rate of clean samples exceeds that of noisy samples, confirming the mathematical soundness of the proposed framework. Through empirical validation on multilingual corpora (MC4, OSCAR, and CulturaX), the method was shown to improve model generalization and stability, achieving consistent accuracy gains on the Turkish HellaSwag benchmark while maintaining computational feasibility. These results were obtained using a pretrained model capable of distinguishing informative from uninformative samples; investigating how the proposed algorithm behaves when training from scratch with randomly initialized weights constitutes a valuable future direction.

The results confirm that cyclical probabilistic filtering outperforms both static and

purely probabilistic alternatives. It achieves higher data efficiency by focusing model updates on the most informative regions of the loss spectrum. Furthermore, the analyses on computational efficiency and bias robustness demonstrated that the approach introduces negligible memory overhead, moderate throughput reduction, and no amplification of societal biases. The catastrophic forgetting analysis showed that the method enhances retention of early-learned samples rather than degrading them, while the synthetic noise experiments validated its ability to adapt effectively to domain shifts and dynamically evolving data distributions. These findings establish the proposed framework as a practical, and adaptive neutral enhancement to conventional large-scale training pipelines. Nevertheless, the effectiveness of the loss signal may vary with model capacity; models below roughly 1B parameters may produce weaker or less reliable loss-based distinctions. Consequently, future work could evaluate the method across both smaller and substantially larger model families to better characterize this scaling behavior.

From a broader perspective, this work demonstrates that data quality can be regulated dynamically within the training loop, reducing reliance on expensive preprocessing or manual curation. The cyclical nature of the filtering ensures balanced exposure to both challenging and simple examples, promoting representational diversity and improving robustness in multilingual and low-resource scenarios. In this regard, the research contributes to a more general understanding of how sample-level loss dynamics can inform adaptive data selection mechanisms during large-scale model optimization. Moreover, while this study focused exclusively on text data, the underlying principles of dynamic filtering and loss-driven adaptivity are broadly applicable. With modest modifications, similar optimization strategies could be extended to vision, audio, tabular, or fully multimodal training pipelines, offering opportunities for cross-domain generalization.

Future work may extend this approach in several directions. One promising avenue is to incorporate semantic and embedding-based similarity measures alongside loss statistics to refine the definition of redundancy and informativeness. Another direction involves exploring reinforcement-learning–based controllers that automatically tune filtering parameters according to validation performance or model confidence. Additionally, applying this framework to multimodal or cross-lingual training could provide valuable insights into how dynamic filtering interacts with heterogeneous data modalities. Finally, integrating the method with distributed or asynchronous training systems could further improve its scalability and deployment efficiency in industrial-scale environments. Further research may also examine how varying noise assumptions, model initialization schemes, and

dataset scales influence the emergent filtering dynamics, enabling a more principled understanding of when and why loss-based selection succeeds.

In summary, this research has achieved its primary goal of designing, implementing, and validating a computationally lightweight yet effective in-training filtering mechanism that enhances generalization and robustness in large-scale multilingual language model pretraining. By unifying statistical rigor with practical adaptability, the proposed cyclical probabilistic filtering framework offers a new perspective on data-centric optimization in modern machine learning—bridging the gap between theoretical robustness and scalable real-world application. Taken together, these contributions highlight both the practicality and the extensibility of dynamic filtering, motivating deeper exploration of adaptive data selection as a core component of future large-scale model training strategies.

REFERENCES

[1] A. Vaswani et al., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] D. Su et al., "Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset," *arXiv preprint arXiv:2412.02595*, 2024. arXiv: 2412.02595 [cs.CL].

[3] G. Penedo et al., "The fineweb datasets: Decanting the web for the finest text data at scale," *Advances in Neural Information Processing Systems*, vol. 37, pp. 30 811–30 849, 2024.

[4] M. Weber et al., "Redpajama: An open dataset for training large language models," *Advances in neural information processing systems*, vol. 37, pp. 116 462–116 492, 2024.

[5] C. Raffel, N. Shazeer, A. Roberts, et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[6] G. Wenzek, M.-A. Lachaux, A. Conneau, et al., "Ccnet: Extracting high quality monolingual datasets from web crawl data," *arXiv preprint arXiv:1911.00359*, 2020.

[7] J. Abadji, P. Ortiz Suárez, L. Romary, et al., "Oscar 22.01: A multilingual dataset of web documents," *arXiv preprint arXiv:2201.07299*, 2022.

[8] G. Penedo, H. Kydlíček, L. Ben Allal, et al., "Refinedweb: A massive dataset for open language modeling," *arXiv preprint arXiv:2305.13870*, 2023.

[9] L. Soldaini, C. Raffel, A. Roberts, et al., "Dolma: An open corpus of three trillion tokens for language model pretraining research," *arXiv preprint arXiv:2401.00047*, 2024.

[10] E. Henriksson, O. Tarkka, F. Ginter, "Finerweb-10bt: Refining web data with llm-based line-level filtering," *arXiv preprint arXiv:2501.07314*, 2025.

[11] G. Penedo et al., "Fineweb2: One pipeline to scale them all–adapting pre-training data processing to every language," *arXiv preprint arXiv:2506.20920*, 2025.

[12] Y. Mu, M. Jin, X. Song, N. Aletras, "Enhancing data quality through simple de-duplication: Navigating responsible computational social science research," *arXiv preprint arXiv:2410.03545*, 2024.

[13] P. Maini, W. Zhang, X. Li, et al., "Loss-based data filtering for efficient language model pretraining," *arXiv preprint arXiv:2405.67890*, 2024.

[14] N. Muennighoff, Y. Wang, Y. Su, et al., "Adaptive data selection for language model training," *arXiv preprint arXiv:2406.78901*, 2024.

[15] Y. Bengio, J. Louradour, R. Collobert, J. Weston, "Curriculum learning," *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[16] A. H. Jiang et al., "Accelerating deep learning by focusing on the biggest losers," *arXiv preprint arXiv:1910.00762*, 2019.

[17] A. Katharopoulos, F. Fleuret, "Not all samples are created equal: Deep learning with importance sampling," *International conference on machine learning*, PMLR, 2018, pp. 2525–2534.

[18] X. Peng, L. Li, F.-Y. Wang, "Accelerating minibatch stochastic gradient descent using typicality sampling," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4649–4659, 2019.

[19] E. Dedeoglu, H. T. Kesgin, M. F. Amasyali, "A robust optimization method for label noisy datasets based on adaptive threshold: Adaptive-k," *Frontiers of Computer Science*, vol. 18, no. 4, p. 184 315, 2024.

[20] V. Shah, X. Wu, S. Sanghavi, "Choosing the sample with lowest loss makes sgd robust," *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 2120–2130.

[21] J. Dodge et al., "Documenting large webtext corpora: A case study on the colossal clean crawled corpus," *arXiv preprint arXiv:2104.08758*, 2021.

[22] E. M. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 610–623.

[23] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[24] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[25] S. Hochreiter, J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] K. Cho et al., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[27] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., "Improving language understanding by generative pre-training," 2018.

[28] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[29] H. Touvron, T. Lavril, G. Izacard, et al., "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[30] S. M. S. Mohammadabadi, B. C. Kara, C. Eyupoglu, C. Uzay, M. S. Tosun, O. Karakuş, "A survey of large language models: Evolution, architectures, adaptation, benchmarking, applications, challenges, and societal implications," *Electronics*, 2025.

[31] V. Nair, G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[32] D. Hendrycks, K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[33] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning internal representations by error propagation," Tech. Rep., 1985.

[34] D. P. Kingma, J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] I. Loshchilov, F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[36] R. Sennrich, B. Haddow, A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[37] L. R. Bahl, F. Jelinek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 179–190, 1983.

[38] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in neural information processing systems*, vol. 36, pp. 53 728–53 741, 2023.

[39] L. Ouyang et al., "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[41] Z. Shao et al., "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," *arXiv preprint arXiv:2402.03300*, 2024.

[42] T. Guo et al., "Large language model based multi-agents: A survey of progress and challenges," *arXiv preprint arXiv:2402.01680*, 2024.

[43] Common Crawl Foundation, *Common crawl corpus*, `https://commoncrawl.org`, Open repository of web crawl data. Accessed: 2025-10-11, 2024.

[44] J. Hoffmann, S. Borgeaud, A. Mensch, et al., "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556*, 2022.

[45] L. Gao, S. Biderman, S. Black, et al., "The pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.

[46] K. Lee, K. Pichotta, Y. Chen, et al., "Deduplicating training data makes language models better," *arXiv preprint arXiv:2107.06499*, 2022.

[47] N. Carlini, F. Tramer, E. Wallace, et al., "Extracting training data from diffusion models," *arXiv preprint arXiv:2305.10973*, 2023.

[48] X. Li, Y. Wang, W. Zhang, et al., "Dclm: Data-centric language modeling," *arXiv preprint arXiv:2403.12345*, 2024.

[49] F. Duan et al., "Enhancing llms via high-knowledge data selection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 2025, pp. 23 832–23 840.

[50] T. Fu, Z. Zhang, "Cpsg-mcmc: Clustering-based preprocessing method for stochastic gradient mcmc," *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 841–850.

[51] Z. Yu, S. Das, C. Xiong, "Mates: Model-aware data selection for efficient pretraining with data influence models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 108 735–108 759, 2024.

[52] J. T. Wang, T. Wu, D. Song, P. Mittal, R. Jia, "Greats: Online selection of high-quality data for llm training in every iteration," *Advances in Neural Information Processing Systems*, vol. 37, pp. 131 197–131 223, 2024.

[53] H. T. Kesgin, M. F. Amasyali, "Cyclical curriculum learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[54] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, A. G. Hauptmann, "Self-paced learning with diversity," *Advances in neural information processing systems*, vol. 27, 2014.

[55] Z. Lin et al., "Not all tokens are what you need for pretraining," *Advances in Neural Information Processing Systems*, vol. 37, pp. 29 029–29 063, 2024.

[56] M. Li, Z. Zhang, "2d-curri-dpo: Two-dimensional curriculum learning for direct preference optimization," *arXiv preprint arXiv:2504.07856*, 2025.

[57] X. Zhang et al., "Preference curriculum: Llms should always be pretrained on their preferred data," *arXiv preprint arXiv:2501.13126*, 2025.

[58] D. Hendrycks et al., "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2020.

[59] V. Zangirolami, M. Borrotti, "Dealing with uncertainty: Balancing exploration and exploitation in deep recurrent reinforcement learning," *Knowledge-Based Systems*, vol. 293, p. 111 663, 2024.

[60] S. Wang, Z. Tan, R. Guo, J. Li, "Noise-robust fine-tuning of pretrained language models via external guidance," *arXiv preprint arXiv:2311.01108*, 2023.

[61] B. Wang et al., "Llm4dsr: Leveraing large language model for denoising sequential recommendation," *arXiv preprint arXiv:2408.08208*, 2024.

[62] Y. Cui et al., "Distillation matters: Empowering sequential recommenders to match the performance of large language models," *Proceedings of the 18th ACM Conference on Recommender Systems*, 2024, pp. 507–517.

[63] B. Han et al., "Co-teaching: Robust training of deep neural networks with extremely noisy labels," *Advances in neural information processing systems*, vol. 31, 2018.

[64] D. Sow, H. Woisetschläger, S. Bulusu, S. Wang, H.-A. Jacobsen, Y. Liang, "Dynamic loss-based sample reweighting for improved large language model pretraining," *arXiv preprint arXiv:2502.06733*, 2025.

[65] C. Leys, O. Klein, Y. Dominicy, C. Ley, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013. doi: 10.1016/j.jesp.2013.03.013.

[66] Hugging Face, *Meta-llama/llama-3.2-1b · hugging face*, Accessed on 2025-04-23, 2024. [Online]. Available: https://huggingface.co/meta-llama/Llama-3.2-1B.

[67] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, Y. Choi, "Hellaswag: Can a machine really finish your sentence?" *arXiv preprint arXiv:1905.07830*, 2019.

[68] malhajar, *Malhajar/hellaswag_tr-v0.2 · datasets at hugging face*, Accessed on 2025-04-23, 2024. [Online]. Available: https://huggingface.co/datasets/malhajar/hellaswag_tr-v0.2.

[69] A. Myrzakhan, S. M. Bsharat, Z. Shen, "Open-llm-leaderboard: From multi-choice to open-style questions for llms evaluation, benchmark, and arena," *arXiv preprint arXiv:2406.07545*, 2024.

[70] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," *Advances in neural information processing systems*, vol. 29, 2016.

[71] Turkish Statistical Institute, *Nüfus istatistikleri portalı*, Accessed on 2025-06-22, 2025. [Online]. Available: https://nip.tuik.gov.tr/?value=YeniDoganIsimleri.

[72] L. Ben Allal, A. Lozhkov, G. Penedo, T. Wolf, L. von Werra, *Cosmopedia*, Accessed in 2025, Feb. 2024. [Online]. Available: https://huggingface.co/datasets/HuggingFaceTB/cosmopedia.

PUBLICATIONS FROM THE THESIS

Papers

1. H. T. Kesgin and M. F. Amasyali, "A cyclical loss-based optimization algorithm for pretraining LLMs on noisy data," *Knowledge-Based Systems*, p.114189, 2025, Elsevier. DOI: 10.1016/j.knosys.2025.114189.

Conference Papers

1. H. T. Kesgin and M. F. Amasyali, "Development of Optimization Algorithms for Training Language Models," *YTU GradColloquium'24 – Artificial Intelligence*, Yildiz Technical University, Davutpasa Campus, Istanbul, Türkiye, June 4, 2024.