MEDUSA Supplementary Notes

In-depth Training Strategies and Experimental Details

Group Meeting Supplementary Material

January 5, 2026

Abstract

This document supplements the MEDUSA paper details not covered in the previous group meeting, focusing on training strategies, loss function design, and experimental settings. It includes the precise mathematical definition of MEDUSA Heads, comparison of MEDUSA-1 and MEDUSA-2 training procedures, Self-Distillation methods, Typical Acceptance mechanism, and optimized tree structure construction algorithms.

Contents

1 Precise Definition of MEDUSA Head Structure **3**
1.1 MEDUSA Structure Review . . . 3
1.2 MEDUSA Head Structure . . . 3
1.3 MEDUSA-1 vs MEDUSA-2 . . . 3
1.4 Initialization Strategy . . . 4

2 Detailed Training Process **4**
2.1 Ground Truth Construction for Training Data . . . 4
2.2 MEDUSA-1 Loss Function . . . 5
2.2.1 Design Motivation for Weight $\lambda_k = 0.8^k$ . . . 5
2.3 MEDUSA-2 Loss Function . . . 5

3 Detailed Explanation of MEDUSA-2's Three Training Strategies **6**
3.1 Strategy 1: Combined Loss . . . 6
3.2 Strategy 2: Differential Learning Rates . . . 6
3.3 Strategy 3: Heads Warmup (Two-Stage Training) . . . 6
3.3.1 Necessity Verification of Two-Stage Training (Table 2) . . . 7

4 Self-Distillation Detailed Explanation **7**
4.1 Application Scenarios . . . 7
4.2 Data Generation Process . . . 7
4.3 Special Handling for MEDUSA-2 Self-Distillation . . . 7

5 Optimized Tree Construction **8**
5.1 Problem Background . . . 8
5.2 Accuracy Estimation . . . 8
5.3 Greedy Tree Construction Algorithm . . . 9
5.4 Optimization Effect . . . 9

6 Paper Experimental Settings Summary (Appendix B) 9
6.1 Common Settings 9
6.2 Vicuna 7B/13B Settings 10
6.3 Self-Distillation Settings 10

7 Ablation Study Summary 10
7.1 Ablations Conducted in the Paper 10
7.2 Ablations Not Conducted in the Paper 10

8 Core Metrics Definitions (Appendix B.1) 11

9 Speedup Contribution of Each Technique 11

10 Experimental Results Summary 11
10.1 Speedup Effects on Different Models 11
10.2 Speedup Effects on Different Task Types 12

11 Appendix: Comparison with Speculative Decoding 12

1 Precise Definition of MEDUSA Head Structure

1.1 MEDUSA Structure Review

Core Idea: Add K prediction heads (MEDUSA heads) on top of the LLM's last layer hidden state to predict the next K tokens in parallel.

1.2 MEDUSA Head Structure

All MEDUSA heads in the paper adopt a unified **residual MLP structure**:

$$\boxed{p_t^{(k)} = \text{softmax}\left(W_2^{(k)} \cdot \left[\text{SiLU}(W_1^{(k)} \cdot h_t) + h_t\right]\right)} \tag{1}$$

where $W_1^{(k)} \in \mathbb{R}^{d\times d}$, $W_2^{(k)} \in \mathbb{R}^{V\times d}$, and $h_t \in \mathbb{R}^d$ is the hidden state.

Initialization Strategy: $W_2^{(k)}$ copies the original LM head weights, $W_1^{(k)}$ is initialized to zero, ensuring initial predictions match the original model.

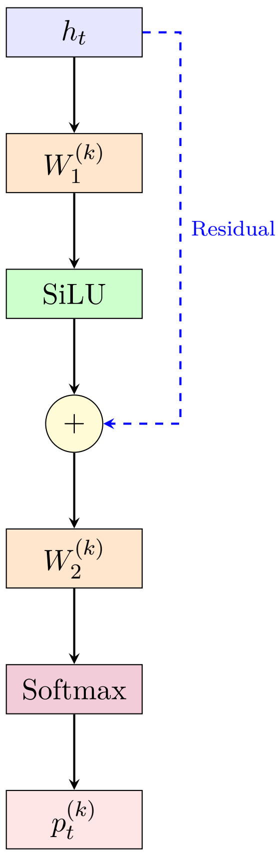


Figure 1: MEDUSA Head Structure: Single-layer Residual MLP

1.3 MEDUSA-1 vs MEDUSA-2

Both have **identical head structures**; the difference lies in **training strategies**:

Table 1: MEDUSA-1 vs MEDUSA-2 Training Strategy Comparison

Comparison Dimension	MEDUSA-1	MEDUSA-2
Backbone Training	Frozen	Joint Training
Training Target	MEDUSA heads only	Backbone + heads
Training Cost	Low (single GPU + quantization)	High
Head Prediction Accuracy	Lower	Higher
Speedup	2.2×	2.8×
Generation Quality	Same as original model	Requires special recipe to maintain

1.4 Initialization Strategy

Table 2: MEDUSA Head Parameter Initialization Strategy

Parameter	Initialization Method	Purpose
$W_2^{(k)}$	Copy original LM Head weights	Initial predictions match original model
$W_1^{(k)}$	Zero initialization	Initially residual connection directly passes h_t

Initialization Effect Analysis:

At the start of training, since $W_1^{(k)} = \mathbf{0}$, we have:

$$\text{SiLU}(W_1^{(k)} \cdot h_t) = \text{SiLU}(\mathbf{0}) = \mathbf{0} \tag{2}$$

Therefore:

$$p_t^{(k)} = \text{softmax}(W_2^{(k)} \cdot h_t) = p_t^{(0)} \tag{3}$$

That is, all MEDUSA Head initial outputs are identical to the original LM Head, ensuring a stable training starting point.

2 Detailed Training Process

2.1 Ground Truth Construction for Training Data

Assume a training sequence is:

```
tokens:  ["The", "cat", "sat", "on", "the", "mat", "<eos>"]
     index:    0     1     2     3     4     5      6
```

Using **K=3 MEDUSA Heads**, the prediction targets for each head at each position t are shown in Table 3.

Table 3: Prediction Targets (Ground Truth) for Each Head at Different Positions

Position t	Input token	LM Head (k=0)	Head 1 (k=1)	Head 2 (k=2)	Head 3 (k=3)
0	"The"	"cat" (t+1=1)	"sat" (t+2=2)	"on" (t+3=3)	"the" (t+4=4)
1	"cat"	"sat" (t+1=2)	"on" (t+2=3)	"the" (t+3=4)	"mat" (t+4=5)
2	"sat"	"on" (t+1=3)	"the" (t+2=4)	"mat" (t+3=5)	"<eos>" (t+4=6)
3	"on"	"the" (t+1=4)	"mat" (t+2=5)	"<eos>" (t+3=6)	Out of bounds
4	"the"	"mat" (t+1=5)	"<eos>" (t+2=6)	Out of bounds	Out of bounds
5	"mat"	"<eos>" (t+1=6)	Out of bounds	Out of bounds	Out of bounds

Out-of-bounds Handling: Use mask to ignore out-of-bounds positions in loss calculation; no gradient computed.

2.2 MEDUSA-1 Loss Function

The total loss function for MEDUSA-1 is defined as:

$$\boxed{\mathcal{L}_{\text{MEDUSA-1}} = \sum_{k=1}^{K} \lambda_k \cdot \mathcal{L}_k} \tag{4}$$

where the loss for a single head is:

$$\mathcal{L}_k = -\frac{1}{|T_k|} \sum_{t \in T_k} \log p_t^{(k)}(y_{t+k+1}) \tag{5}$$

Table 4: MEDUSA-1 Loss Function Symbol Definitions

Symbol	**Meaning**
K	Total number of MEDUSA Heads ($K = 5$ in the paper)
λ_k	Loss weight for the k-th head, set to 0.8^k
T_k	Valid position set for the k-th head (excluding out-of-bounds)
y_{t+k+1}	Ground truth token at position $t + k + 1$
$p_t^{(k)}(y)$	Probability of the k-th head predicting token y

2.2.1 Design Motivation for Weight $\lambda_k = 0.8^k$

Table 5: Weights and Explanations for Different k Values

k	$\lambda_k = 0.8^k$	**Explanation**
1	0.800	Predicting next token, relatively easy
2	0.640	Predicting one token ahead, harder
3	0.512	Predicting two tokens ahead, very hard
4	0.410	Predicting three tokens ahead, very difficult
5	0.328	Predicting four tokens ahead, extremely difficult

Design Principle: As k increases, prediction difficulty increases, and $\mathcal{L}_k$ naturally becomes larger. Using decreasing weights prevents distant heads from dominating the optimization direction.

2.3 MEDUSA-2 Loss Function

The total loss function for MEDUSA-2 is defined as:

$$\boxed{\mathcal{L}_{\text{MEDUSA-2}} = \mathcal{L}_{\text{LM}} + \lambda_0 \cdot \mathcal{L}_{\text{MEDUSA-1}}} \tag{6}$$

Expanded form:

$$\mathcal{L}_{\text{MEDUSA-2}} = \underbrace{-\frac{1}{|T|} \sum_{t \in T} \log p_t^{(0)}(y_{t+1})}_{\text{Original LM next-token prediction loss}} + \lambda_0 \cdot \underbrace{\sum_{k=1}^{K} \lambda_k \cdot \left(-\frac{1}{|T_k|} \sum_{t \in T_k} \log p_t^{(k)}(y_{t+k+1}) \right)}_{\text{MEDUSA heads multi-token prediction loss}} \tag{7}$$

Table 6: MEDUSA-2 Loss Function Additional Symbol Definitions

Symbol	Meaning	Value in Paper
$\mathcal{L}_{\text{LM}}$	Backbone's original next-token prediction loss	-
λ_0	Weight balancing backbone loss and MEDUSA loss	0.2 or 0.01
$p_t^{(0)}$	Original LM Head's prediction distribution	-

3 Detailed Explanation of MEDUSA-2's Three Training Strategies

3.1 Strategy 1: Combined Loss

Problem: If only training MEDUSA heads, the backbone's next-token prediction capability may degrade.
Solution: Include the backbone's cross-entropy loss in the loss function, as shown in Equation (6).

3.2 Strategy 2: Differential Learning Rates

Problem: The backbone is already trained, while MEDUSA heads are trained from scratch, requiring different learning rates.

Table 7: MEDUSA-2 Differential Learning Rate Settings (Appendix B.3)

Component	Learning Rate	Ratio
Backbone (LoRA)	5×10^{-4}	1×
MEDUSA Heads	2×10^{-3}	4×

3.3 Strategy 3: Heads Warmup (Two-Stage Training)

Problem: At the beginning of training, MEDUSA heads' loss is very large, and gradients may damage the backbone.

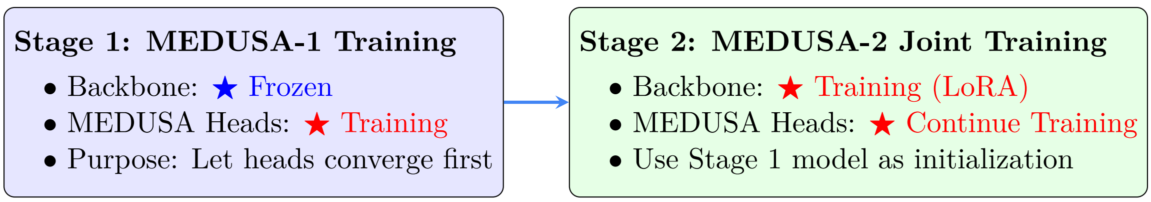


Figure 2: MEDUSA-2 Two-Stage Training Process

3.3.1 Necessity Verification of Two-Stage Training (Table 2)

Table 8: Effect Comparison of Different Training Methods

Training Method	MT-Bench Quality	Speedup
Baseline (Original Model)	6.17	N/A
Direct Fine-tuning (no warmup)	5.925	N/A
MEDUSA-1	6.23	2.18×
MEDUSA-2 (Two-Stage)	6.18	**2.83**×

Conclusion: Direct fine-tuning leads to quality degradation (-0.245), while two-stage training maintains quality.

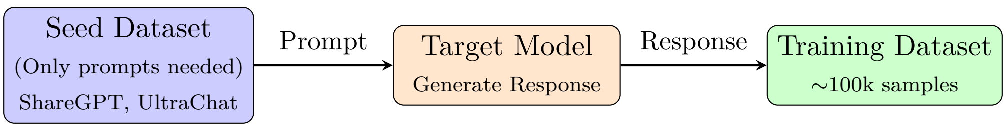
4 Self-Distillation Detailed Explanation

4.1 Application Scenarios

Table 9: Self-Distillation Application Scenarios

Scenario	Problem	Solution
Training data unavailable	Model authors only release weights, not training data	Use model to generat
Post-RLHF models	Original SFT data distribution doesn't match post-RLHF	Generate distribution

4.2 Data Generation Process



Multi-turn dialogue generation:
Method 1: Iteratively feed multi-turn prompts (Vicuna-33B)
Method 2: Let model self-talk (Zephyr-7B)

Figure 3: Self-Distillation Data Generation Process

4.3 Special Handling for MEDUSA-2 Self-Distillation

Problem: Directly training backbone with self-generated data leads to quality degradation (even without adding MEDUSA heads!)
Reason: Self-generated data tokens are sampled, not true "ground truth"; using cross-entropy directly introduces noise.
Solution: KL Divergence Loss

$$\boxed{\mathcal{L}_{\text{LM-distill}} = \text{KL}(p^{(0)}_{\text{original}} \| p^{(0)}_{\text{student}})} \quad (8)$$

Table 10: KL Divergence Loss Symbol Definitions

Symbol	Meaning
$p^{(0)}_{\text{original}}$	Original model (teacher) prediction distribution
$p^{(0)}_{\text{student}}$	Model being trained (student) prediction distribution

5 Optimized Tree Construction

5.1 Problem Background

A simple Cartesian product tree structure is "regular," but different positions have different top-k prediction accuracies. However, no matter Medusa or EAGLE, speedup strategy fails with normal inference batch_size.

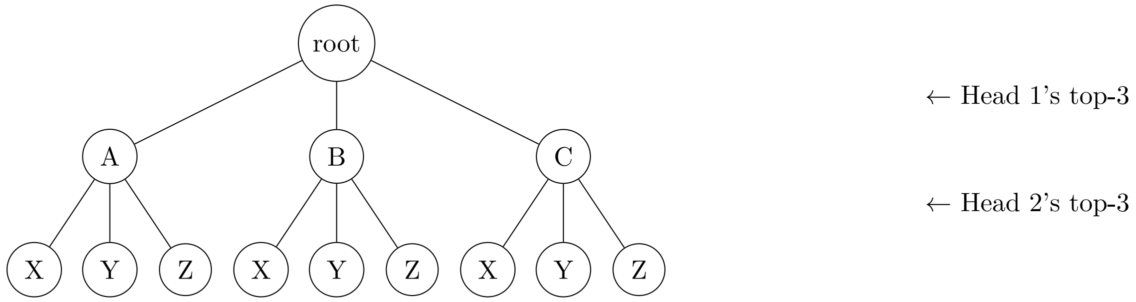


Figure 4: Regular Cartesian Product Tree Structure (Dense Tree)

5.2 Accuracy Estimation

Let $a_k^{(i)}$ be the accuracy of the k-th head's i-th top prediction:

$$a_k^{(i)} = P(\text{top-}i \text{ correct}) - P(\text{top-}(i-1) \text{ correct}) \tag{9}$$

Candidate Sequence Accuracy Estimation:

Assuming heads are independent, the accuracy of candidate $[i_1, i_2, ..., i_k]$ is:

$$\text{Acc}([i_1, i_2, ..., i_k]) = \prod_{j=1}^{k} a_j^{(i_j)} \tag{10}$$

Expected Acceptance Length:

$$\mathbb{E}[\text{accepted length}] = \sum_{[i_1,...,i_k]\in\mathcal{I}} \prod_{j=1}^{k} a_j^{(i_j)} \tag{11}$$

where $\mathcal{I}$ is the set of all nodes in the tree.

5.3 Greedy Tree Construction Algorithm

Algorithm 1 Greedy Tree Construction Algorithm

Require: Accuracy $a_k^{(i)}$ for each head's top-k, node budget N
Ensure: Optimal tree structure T

```
Initialize tree T = {root}
while |T| < N do
    Compute contribution (accuracy) of all addable nodes
    Add the node with maximum contribution to tree T
end while
return T
```

Key Insight: A new node's contribution to expected acceptance length = that node's accuracy (because the node only has a chance to be accepted when all ancestors are correct).

5.4 Optimization Effect

Table 11: Dense Tree vs Sparse Tree Effect Comparison

Tree Type	Node Count	Speedup
Dense Tree (Cartesian)	256	$\sim$2.5×
Sparse Tree (Optimized)	**64**	$\sim$**3.2**×

Conclusion: A 64-node sparse tree outperforms a 256-node dense tree!

6 Paper Experimental Settings Summary (Appendix B)

6.1 Common Settings

Table 12: MEDUSA Common Training Settings

Parameter	Value
Framework	Axolotl
Optimizer	8-bit AdamW
Learning Rate Schedule	Cosine with warmup
Number of MEDUSA Heads	5
Head Layers	1
λ_k	0.8^k
LoRA rank	32
LoRA α	16
LoRA dropout	0.05
LoRA Application Scope	All linear layers (including LM head)

6.2 Vicuna 7B/13B Settings

Table 13: Vicuna 7B/13B Training Settings (MEDUSA-1 → MEDUSA-2)

Parameter	Value
Global batch size	64
Backbone learning rate	5×10^{-4}
MEDUSA Heads learning rate	2×10^{-3}
Warmup steps	40
Backbone quantization	4-bit
λ_0 (MEDUSA-2)	0.2
Fine-tuning method	QLoRA

6.3 Self-Distillation Settings

Table 14: Self-Distillation Training Settings (Vicuna-33B / Zephyr-7B)

Parameter	Value
Training method	Direct MEDUSA-2 (no two-stage)
λ_0 schedule	Sine schedule, gradually increasing to peak
Backbone LoRA learning rate	1×10^{-4}
Warmup steps	20
λ_0 peak	0.01

7 Ablation Study Summary

7.1 Ablations Conducted in the Paper

Table 15: Ablation Studies Completed in the Paper

Ablation Content	Location	Variable	Conclusion
Tree attention configuration	Section 3.3.1, Fig.4	Node count	64 nodes optimal
Typical acceptance threshold	Section 3.3.2, Fig.5	ϵ	Threshold vs quality/speed t
Two-stage training necessity	Section 3.3.3, Table 2	With/without warmup	Quality drops 0.245 without

7.2 Ablations Not Conducted in the Paper

Table 16: Design Choices Not Ablated in the Paper

Not Ablated	Paper's Approach	Possible Reason
$\lambda_k = 0.8^k$	Intuitive explanation	Not a main contribution
$\lambda_0 = 0.2$ or 0.01	Appendix statement	Different values for different settings
Learning rate ratio 4×	Appendix statement	Common practice
LoRA rank=32	Appendix statement	Standard configuration
Head layers=1	Main text statement	Simple and effective

8 Core Metrics Definitions (Appendix B.1)

Table 17: MEDUSA Core Evaluation Metrics Definitions

Metric	Definition	Description
Acceleration Rate	Average tokens generated per decoding step	Standard autoregressive model = 1.0
Overhead	$\frac{\text{MEDUSA latency per step}}{\text{Vanilla latency per step}}$	Extra overhead ratio per step
Speedup	$\frac{\text{Acceleration Rate}}{\text{Overhead}}$	Actual wall-clock time speedup

Relationship:

$$\text{Speedup} = \frac{\text{Acceleration Rate}}{\text{Overhead}} \quad (12)$$

Example Calculation:

- Acceleration Rate = 3.47
- Overhead = 1.22
- Speedup = 3.47 / 1.22 = **2.84**×

9 Speedup Contribution of Each Technique

Table 18: Each Technique's Contribution to Speedup (Table 3)

Technique	Speedup
MEDUSA-1 heads (without tree attention)	∼1.5×
+ Tree attention	∼1.9×
+ Optimized tree configuration	∼2.2×
+ MEDUSA-2 training	∼**2.8**×

10 Experimental Results Summary

10.1 Speedup Effects on Different Models

Table 19: MEDUSA-2 Speedup Effects on Different Models (Table 1)

Model	Acc. Rate	Overhead	Quality (MT-Bench)	Speedup
Vicuna-7B	3.47	1.22	6.18 (+0.01)	2.83×
Zephyr-7B	3.14	1.18	7.25 (-0.07)	2.66×
Vicuna-13B	3.51	1.23	6.43 (-0.14)	2.83×
Vicuna-33B	3.01	1.27	7.18 (+0.05)	2.35×

10.2 Speedup Effects on Different Task Types

Table 20: Vicuna-7B MEDUSA-2 Speedup Effects on Different Task Types (Figure 3b)

Task Type	Speedup
Extraction	3.62×
Coding	3.29×
Math	3.01×
STEM	2.77×
Writing	2.72×
Roleplay	2.70×
Reasoning	2.58×
Humanities	2.58×

Observation: Structured outputs (Extraction, Coding) have the best speedup effects because outputs are more predictable.

11 Appendix: Comparison with Speculative Decoding

Table 21: MEDUSA vs Speculative Decoding Comparison

Aspect	Speculative Decoding	MEDUSA
Draft Source	Independent small model	Additional decoding heads
Training Cost	Requires pre-training draft model	Few hours of fine-tuning
Deployment Complexity	Requires maintaining two models	Single model
Distributed-friendly	Difficult	Easy
Speedup (Vicuna-7B)	1.47×	**2.83**×
Speedup (Vicuna-33B)	1.60×	**2.35**×

Summary

This document provides detailed supplementary content on the following aspects of the MEDUSA paper:

1. **MEDUSA Head Structure**: Precise mathematical definitions and initialization strategies
2. **Training Process**: Ground Truth construction, Loss function expansion, training pseudocode
3. **MEDUSA-2 Three Strategies**: Combined Loss, differential learning rates, two-stage training
4. **Self-Distillation**: Data generation, KL Divergence Loss, LoRA memory optimization
5. **Typical Acceptance**: Comparison with Rejection Sampling, dynamic threshold mechanism
6. **Optimized Tree Structure**: Greedy algorithm, effect comparison
7. **Experimental Settings**: All hyperparameter configurations

8. **Ablation Studies**: Completed and not-completed ablations