

Dynamic Semantic Compression for Efficient Latent-Space Inference in Large Language Models

Peipei Li, Dongsen Zhang, Yuchen Liu, Wenjun Xu*

Beijing University of Posts and Telecommunications, Beijing, China

Abstract

Large Language Models (LLMs) primarily perform inference at the token level, resulting in substantial memory overhead and compromised computational efficiency. In this paper, we propose a Dynamic Semantic Extraction and Inference (DSEI) framework, which achieves segment-level inference within the latent space through a two-stage training strategy. First, we construct a Dynamic Semantic Autoencoder (DSAE) via self-supervised learning. DSAE dynamically extracts segment-level semantics and compresses them into compact latent representations via adaptive semantic weighting and gated fusion. Subsequently, we integrate the DSAE into the LLM architecture and train the model to infer over dense latent space. DSEI substantially reduces both input and generation sequences and significantly enhances inference efficiency. Extensive experiments conducted on the Wanjian dataset demonstrate that DSEI reduces perplexity by 48% compared to static sentence-level latent inference baseline. Furthermore, compared to standard LLMs using token-level inference, DSEI accelerates inference speed by $2.5\times$ and reduces memory overhead by 90%.

Keywords: Dynamic Semantic Compression, Segment-Level Inference, Efficient LLM Inference, Autoencoder

1. Introduction

Autoregressive large language models process and generate text at the

*Corresponding author

Email address: lipeipei@bupt.edu.cn, , liu20031225yuchen@bupt.edu.cn, wjxu@bupt.edu.cn (Peipei Li, Dongsen Zhang, Yuchen Liu, Wenjun Xu)

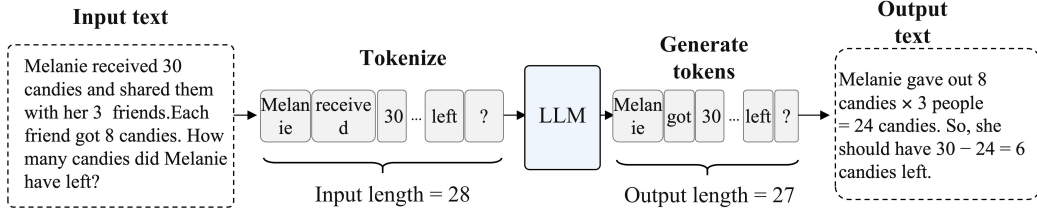
token level. While this formulation provides fine-grained linguistic control, it also creates long sequential computation paths, intensive memory access, and high deployment costs as shown in Fig. 1(top). These costs become particularly restrictive when LLMs are deployed in resource-constrained or latency-sensitive engineering scenarios. Reducing the computational granularity of LLM inference from discrete tokens to compact semantic units is therefore a promising direction for efficient LLM deployment.

Recent latent-space inference methods take an important step in this direction by replacing token-level Chains-of-Thought (CoT) with latent representations, thereby reducing the number of autoregressive reasoning steps [1], [2], [3]. While valuable, such approaches are generally limited to compressing the model’s intermediate reasoning process. In contrast, [4] encodes each sentence into a latent representation, extending latent-space inference to the entire input-output stream as shown in Fig. 1(middle). However, this fixed granularity compression paradigm remains limited. It compresses every sentence into a single latent representation regardless of sentence length and treats all tokens uniformly during encoding. As a result, it overlooks both sentence-level and token-level differences in semantic density, which may lead to information loss or redundant compression and limit inference performance.

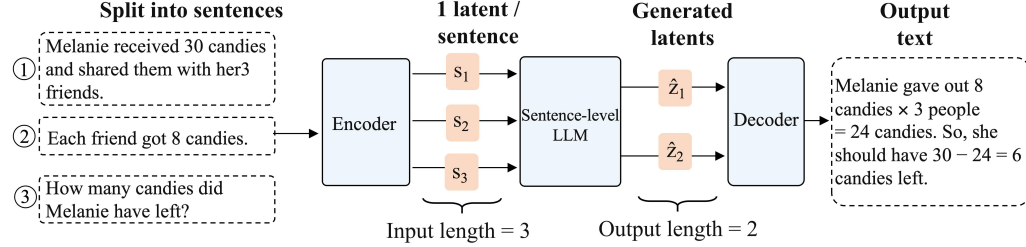
To address these limitations, we propose Dynamic Semantic Extraction and Inference (DSEI), a framework designed to improve the efficiency of LLM inference by shifting computation from token-level processing to adaptive segment-level latent-space inference as shown in Fig. 1(bottom). DSEI integrates a Dynamic Semantic Autoencoder (DSAE) into the LLM inference pipeline to extract compact yet informative semantic representations from text segments. Specifically, DSAE is first trained in a self-supervised manner to reconstruct input sentences from their latent representations. In the encoder, each sentence is divided into segments according to its token length, and token-level importance weights are estimated from hidden states to capture semantic contributions with different levels of information density. The weighted token representations are then aggregated to form dynamic semantic features, which are further fused with basic semantic features through a gating mechanism. The decoder is trained to reconstruct the original sentences from these representations. After this stage, the trained encoder and decoder are attached to the input and output sides of the LLM, respectively, and the overall model is further optimized through end-to-end training. In this way, DSEI reduces both input and output sequence lengths while main-

Comparison of inference granularity

1) Token-level inference



2) Fixed sentence-level latent inference



3) Proposed adaptive semantic tokenization

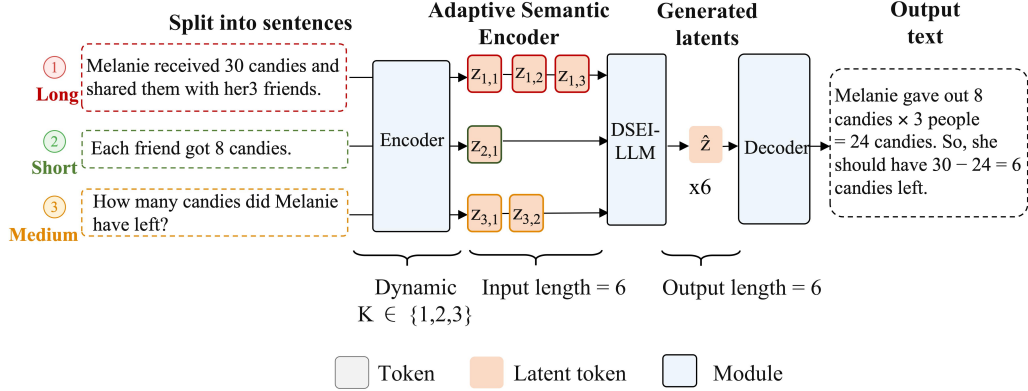


Figure 1: Comparison of token-level inference, fixed sentence-level latent inference, and the proposed dynamic segment-level latent inference. DSEI segments sentences according to token length and extracts compact semantic representations based on token-level importance, reducing both input and output sequence lengths while maintaining semantic fidelity.

taining semantic fidelity.

Extensive experiments on the Wanjuan-1.0 [5] dataset demonstrate that, compared to static semantic extraction baselines, DSAE reduces perplexity by 34% while maintaining computational efficiency. When integrated into LLMs, it achieves a 48% reduction in perplexity (PPL). Furthermore, compared to LLMs that use token-level inference, DSEI reduces memory overhead by 90% while simultaneously lowering PPL by 58%.

Our main contributions are three-fold:

(1) We propose DSAE, a dynamic semantic compressor that partitions sentences into length-based semantic segments and extracts compact latent representations through token-level semantic weighting and gated fusion. This allows for efficient high-fidelity semantic extraction.

(2) We introduce the DSEI framework to integrate DSAE into LLMs, enabling end-to-end segment-level inference within the latent space, thereby reducing the effective sequence length of both input and output streams.

(3) Extensive experiments demonstrate that DSAE reduces perplexity by 34% compared to static semantic extraction baselines, and by 48% when integrated into LLM inference. Furthermore, compared to standard token-level LLM inference, DSEI reduces memory overhead by 90% while lowering perplexity by 58%, demonstrating a practical trade-off between generation quality and deployment efficiency.

2. Related Work

2.1. Semantic compression

Early research primarily focused on acquiring compact sentence vectors or instruction embeddings via contrastive learning [6], [7]; these semantic representations are typically utilized for data retrieval and matching. Recent works have explored diverse semantic compression paradigms. For instance, GIST compresses prompts into a set of Transformer activations via meta-learning [8], while the LLMingua model series [9], [10] achieves hard compression by eliminating tokens with low PPL. Furthermore, ICAE demonstrates the feasibility of employing an encoder-decoder architecture for semantic compression [11]. Building upon this trajectory, 500xCompressor [12] enhances the compression ratio through KV cache compression, whereas C3 [13] compresses semantics using cascaded small-parameter LLMs. SentenceVAE [4], on the other hand, focuses on sentence-level compression, encoding an entire sentence into a single sentence-level latent representation

that can be reversibly reconstructed by a decoder. Another prominent direction involves continuous latent space modeling, which achieves semantic compression via diffusion models [14], [15], [16], [17]. Within this domain, AR-Diffusion [18] improves parallelism and coherence through local causal constraints, while two-stage methods such as LD4LG [19] and PLANNER [20] enhance controllability and efficiency via a compression-then-refinement approach. Unlike fixed sentence-level compression, DSEI uses length-adaptive segmentation and token-level semantic weighting to avoid over-compressing long sentences.

2.2. Latent-space inference

Latent reasoning methods aim to perform inference within continuous latent spaces and can be broadly categorized into three primary directions: knowledge internalization, architectural modification, and autoregressive latent reasoning. Knowledge internalization seeks to directly embed logic and reasoning capabilities into model parameters, thereby reducing reliance on explicit chains of thought or explanatory prompts. For instance, iCoT-SI [21] forces the model to internalize reasoning structures by progressively eliminating explicit reasoning steps during training, whereas TwT [22] emphasizes the roles of multi-teacher distillation and habitual reasoning. Pause [23] encodes reasoning steps into dedicated token embeddings, while CoCoMix [24] demonstrates that reasoning capabilities can be implanted through continuous concept mixing during the pre-training phase. Furthermore, [25] reveals the critical impacts of reasoning granularity, representation format, and teacher model selection on distillation efficiency. Architectural modification leverages the hierarchical structure of Transformers to enable selective reasoning and dynamic computation [26], [27], [28], [29], [30]. Approaches like [31] allow models to dynamically adjust computational steps according to problem difficulty, whereas [32], [33] explore supervision and guidance oriented toward implicit trajectories. Other studies [34], [35], [36], [37] focus on novel architectures and parallel inference. Autoregressive latent reasoning utilizes hidden states to represent the thought process [1], [2]. In this vein, CODI [3] constructs an autoregressive latent variable model combined with self-distillation, while [38], [39] concentrate on enhancing the stability of these hidden states. In contrast to latent inference methods that primarily compress intermediate reasoning trajectories into continuous representations, DSEI extends latent-space inference to the complete input-output stream through segment-level semantic representations.

3. Method

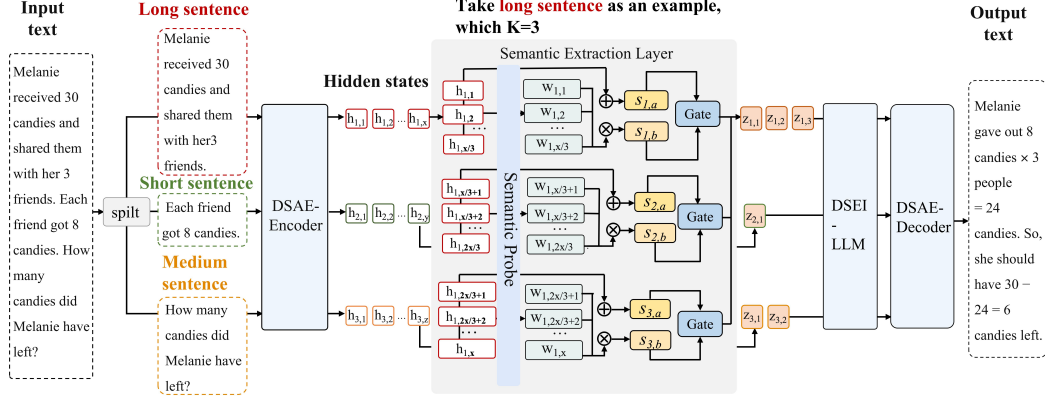


Figure 2: Overall architecture of the proposed DSEI framework. The input text is first split into sentences, and each sentence is partitioned into one or more segments according to token length. The DSAE encoder extracts segment-level latent representations through token-level semantic weighting and gated fusion. These latent representations are then processed by the DSEI-LLM for latent-space inference, and the generated latent outputs are reconstructed into natural language by the DSAE decoder.

In this section, we introduce the proposed DSEI. Given a vanilla LLM $\mathcal{L}$ and an input text consisting of n sentences $\mathbf{x} = \{x^1, x^2, \dots, x^n\}$, the text is tokenized and embedded to obtain m token embedding vectors $\mathbf{e} = \{e_{1:l_1}^1, e_{1:l_2}^2, \dots, e_{1:l_n}^n\}$, where $e_{1:l_n}^n$ denotes the l_n embedding vectors of the n -th sentence x^n , and $m = \sum_{i=1}^n l_i$. $\mathcal{L}$ processes these embeddings and predicts the next token, generating j sentences comprising t tokens $\mathbf{o} = \{o_{1:b_1}^1, o_{1:b_2}^2, \dots, o_{1:b_j}^j\}$, where $o_{1:b_j}^j$ represents the b_j tokens of the j -th sentence, and $t = \sum_{i=1}^j b_i$.

To address the issue of excessively long inputs and outputs in LLMs, we propose enabling the LLM to perform inference within a segment-level latent space, thereby reducing the input length from m to $\sum_{i=1}^n K_i$, where each sentence is partitioned into K_i segments. Similarly, the output length is reduced from t to $\sum_{r=1}^j \hat{K}_r$, where $\hat{K}_r$ denotes the number of latent segments associated with the r -th output sentence. This requires our method to: i) extract the semantics of sentences and efficiently compress them into the latent space; ii) enable the LLM to comprehend segment-level latent representations and perform segment-by-segment inference. DSEI is designed precisely to achieve these two objectives.

3.1. Semantic Autoencoder

As shown in Fig. 2, the input to DSAE is a single sentence, which can be represented as $e_{1:l}^1$ after tokenization and embedding. The encoder is composed of stacking Transformer encoder layers [40], which map these embedding vectors into l hidden states $\mathbf{h} = h_{1:l}$. These hidden states $\mathbf{h}$ are partitioned into K segments according to the token length. Specifically, sentences with fewer than 15 tokens are treated as a single segment, those with 15–30 tokens are partitioned into two, and longer sentences are partitioned into three segments. Formally, the number of segments is determined by the token length l :

$$\mathbf{h}^k = \left\{ h_i \mid \left\lfloor \frac{k \cdot l}{K} \right\rfloor \leq i < \left\lfloor \frac{(k+1) \cdot l}{K} \right\rfloor \right\}, \quad k = 0, 1, \dots, K-1, \quad (1)$$

where $\mathbf{h}^k$ denotes the hidden states belonging to the k -th segment.

To extract the segment semantics based on $\mathbf{h}^k$, a straightforward approach is to sum $\mathbf{h}^k$, as shown in Eq. 2. Although the resulting latent representation s_a retains global information, compressing hidden states containing varying amounts of information equally into one vector also introduces redundancy. To address this issue, we employ a fully connected layer called the Semantic Probe to map $\mathbf{h}^k$ to one dimension, obtaining a sequence of token weights $\mathbf{w}^k$. A softmax function is applied to normalize these weights into attention probabilities. Since $\mathbf{h}^k$ obtained by the self-attention mechanism is context-dependent, $\mathbf{w}^k$ can dynamically identify the semantic informativeness of tokens. The latent representation s_b is obtained by the weighted average of $\mathbf{w}^k$ and $\mathbf{h}^k$, as shown in Eq. 3:

$$s_{k,a} = \text{LayerNorm} \left(\sum \mathbf{h}^k \right) \quad (2)$$

$$s_{k,b} = \text{LayerNorm} \left(\text{softmax}(\text{linear}(\mathbf{h}^k)) \mathbf{h}^{k\top} \right) \quad (3)$$

For the reconstruction task of the decoder, the signal provided by the dynamically extracted $s_{k,b}$ is too sparse, making it difficult for the decoder to predict a reconstructed output consistent with the input. To address this issue, we combine $s_{k,a}$, which contains global information, with s_b through a learnable scalar w_{gate} . This scalar is mapped to a value between 0 and 1 via a sigmoid function, implementing a simple yet effective gating mechanism that

balances the contribution of the two representations. The resulting segment latent representation s_k thus incorporates global information while reducing redundancy. This process can be formally represented as:

$$s_k = \text{sigmoid}(w_{gate}) \cdot s_{k,a} + (1 - \text{sigmoid}(w_{gate})) \cdot s_{k,b} \quad (4)$$

Given a sentence represented by a sequence of segment-level latent representations $\mathbf{s} = s_{1:K}$, the decoder reconstructs the original sentence from all segment representations jointly. The decoder consists of stacked Transformer decoder layers, where $\mathbf{s}$ is used as the key and value in cross-attention. Starting from the start token, the decoder autoregressively generates the reconstructed token probabilities $\mathbf{p} = p_{1:y}$, where p_y denotes the predicted probability of the y -th target token. The reconstruction probabilities are then compared with the original input tokens using focal loss [41], which serves as the training objective in the first stage. This process can be formulated as:

$$\mathcal{L}_{\text{comp}} = \sum_{i=1}^y \text{FL}(p_i) = \sum_{i=1}^y -(1 - p_i)^2 \log(p_i) \quad (5)$$

The initial parameters of the encoder and decoder in DSAE are transferred from the LLM backbone of DSEI. The encoder and decoder in DSAE maintain the same number of layers. For a DSAE with i layers, since the encoder is responsible for extracting features from text embeddings, its parameters are transferred from the first i layers of the functionally equivalent LLM. Similarly, the decoder is responsible for generating text from latent representations, and its parameters are transferred from the last i layers of the functionally equivalent LLM. Compared to random initialization, parameter transfer endows the model with semantic comprehension capabilities, allowing training to proceed without starting from basic grammatical structure construction. This enables efficient training of the model’s semantic extraction abilities, leading to convergence at better performance.

3.2. Segment-by-segment inference

To enable the LLM to perform segment-by-segment inference in the latent space, we integrate DSAE into the LLM and optimize the resulting model in an end-to-end manner. In DSEI, the token embedding layer of the vanilla LLM is replaced by the DSAE encoder and the semantic extraction layer, while the output hidden states of the LLM are decoded into natural language

by the DSAE decoder. Given an input text, we first split it into n sentences by pattern matching based on punctuation marks. For the i -th sentence x^i , the DSAE encoder and semantic extraction layer determine the number of segments K_i according to its token length and encode it into a sequence of segment-level latent representations $s_{1:k_i}^i$. The latent representations of all sentences are then concatenated in their original textual order to form the LLM input $\mathbf{s} = \{s_{1:k_1}^1, s_{1:k_2}^2, \dots, s_{1:k_n}^n\}$, where the total number of latent vectors is $\sum_{i=1}^n K_i$. This encoding process is performed in parallel across sentences in a batched manner. The LLM takes $\mathbf{s}$ as input and autoregressively generates a sequence of output latent representations $\hat{\mathbf{s}} = \hat{s}_{1:T}$.

Since the LLM produces a flat sequence of segment-level latent representations, DSEI further introduces a segmentation head to recover sentence-level structure before decoding. Specifically, for each generated latent representation $\hat{s}_t$, the segmentation head predicts whether it marks the end of a sentence-level latent group. These predicted boundaries partition $\hat{\mathbf{s}}$ into several groups, each corresponding to one output sentence. The DSAE decoder then reconstructs a natural-language sentence from each latent group, and the concatenation of all reconstructed sentences forms the final output of DSEI.

To determine when the LLM should stop generating segment-level latent representations, we train a fully connected layer called the termination head. During training, the termination objective is computed only for generated latent representations that correspond to ground-truth sentence-level latent group boundaries. Given such a boundary representation $\hat{s}_i$, the termination head maps it to a two-dimensional logit vector $\mathbf{d}_i = [d_i^1, d_i^2]$, where the two dimensions indicate whether generation should continue or terminate. If the vector signals an end state, the generation iteration terminates; otherwise, the LLM continues to generate the next latent representation.

During training, we use focal loss [41] to optimize the segmentation head, the termination head, and the latent-generation process jointly. Formally, the total loss is defined as:

$$\mathcal{L}_{\text{infer}} = \underbrace{\frac{\lambda}{j} \sum_{i=1}^t \text{FL}(p_i)}_{\text{generation term}} + \underbrace{\sum_{i=1}^t \text{FL}(b_i)}_{\text{segment term}} + \underbrace{\sum_{t \in \mathcal{B}} \text{FL}(d_i)}_{\text{stop term}}, \quad (6)$$

where λ is a weighting coefficient used to balance the token-level generation term with the stop and segment terms, b_t denote the predicted probabil-

ity assigned to the ground-truth boundary label of $\hat{s}_t$, d_i denote the predicted probability assigned to the ground-truth termination label of $\hat{s}_t$, and $\mathcal{B}$ is the set of boundary positions at the end of sentence.

4. Experiments

We first train DSAE via self-supervised learning, and subsequently integrate the trained DSAE into the LLM for joint end-to-end training. We analyze DSAE and DSEI separately, comparing them with baseline methods and token-by-token LLMs, exploring the characteristics of dynamic semantics, and analyzing the contributions of different components.

4.1. Experimental setup

Dataset and metrics. For both DSAE and DSEI, all experiments are trained and validated on the English subset of the Wanjuan-1.0 dataset [5]. For DSAE, the training set samples approximately 155M sentences, and the validation set consists of 1K non-overlapping sentences. For DSEI, the training set samples approximately 6.4M paragraphs, and the validation set consists of 1K non-overlapping paragraphs. Following [4], we use PPL as the evaluation metric for DSAE and DSEI. Additionally, we employ mean input throughput and mean GPU memory to assess the computational efficiency of DSEI.

Baseline methods. We compare our method with the baseline approach [4], which includes SVAE, an autoencoder with a static fusion mechanism for sentence compression and reconstruction, and SLLM, a sentence-level LLM that integrates SVAE. In SVAE, the sentence-level latent representation is obtained by summing the hidden states from the last layer of the encoder, and SLLM performs sentence-level inference based on this static latent representation. Additionally, we also compare with token-level inference LLMs from the OPT [42] series.

Implementation details. (1) Base Models: We adopt three OPT [42] series models of sizes 125M, 350M, and 1.3B as our base LLMs to validate the effectiveness of our method across different model scales. The dimension of hidden states in DSAE is kept consistent with that of the base LLM. During weight transfer, both the encoder and decoder only utilize the weights of the self-attention modules from the base LLM, excluding the feed-forward network (FFN) weights. (2) Hyperparameters: For DSAE, the maximum input token length is set to 64, batch size to 512, learning rate to 1e-7, and

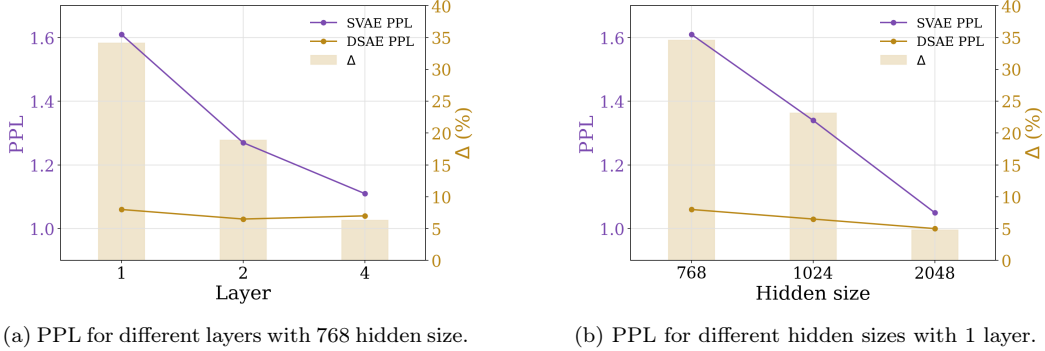


Figure 3: Comparison of PPL for semantic extraction between DSAE and baseline method across varying (a) layers and (b) hidden sizes. Δ denotes percentage decrease in PPL of DSAE compared to the baseline method.

the gating scalar w_{gate} is initialized to 0. In experiments where the base LLM is 125M, the number of layers in DSAE is set to 1, 2, and 4 respectively, while for base LLMs of other sizes it is set to 1. For DSEI, the maximum input sentence length is set to 64, batch size is 4, and the learning rate is 1e-6. For all experiments, λ is set to 0.01. We use the AdamW optimizer [43] with a weight decay of 1e-2 and gradient clipping with a maximum L2 norm of 1. The learning rate follows a linear schedule for the first 5,000 iterations, and then uses a cosine annealing schedule for subsequent iterations. All experiments are conducted on a single RTX 5880 Ada Generation GPU.

4.2. Semantic extraction results

Fig. 3 compares the performance of DSAE with the baseline method SVAE under different model sizes. DSAE demonstrates consistent performance improvements over the static representation fusion-based semantic extraction method across various model depths (varying layers) and widths (varying hidden sizes). Notably, in the smallest model configuration, DSAE achieves a 34% reduction in PPL compared to the baseline method, which is the largest performance gain among all model sizes. This is particularly advantageous for resource-constrained edge devices, which typically employ smaller-scale models.

Tab. 1 presents the sentence reconstruction performance of DSAE compared with the baseline method across multiple test cases. DSAE demonstrates more accurate sentence reconstruction compared to the baseline method. The token weight results for input sentences show that DSAE pays greater

Table 1: Test examples of sentence reconstruction between baseline method and DSAE. The darker the color, the greater the weight of the token. Text in red denotes mismatches between reconstructed sentence and input sentence, and $\triangle$ denotes missing output.

Weight of input sentence	Output sentence	
	SVAE	DSAE
Some also have performance venues for various kinds of artists .	Some also have performance venues for various kinds of artists.	Some also have performance venues for various kinds of artists.
Intense pulsed light device emit a range wavelength (515 - 1200) of light .	Intense pulsed light device emit a range of puls(660-1200) $\triangle$ light.	Intense pulsed light device emit a range wavelength(515-1200) of light.
Makes the perfect gift for Schylling enthusiasts that are at least 3 years old .	Makes the perfect gift for Schreferably that enthusiasts at $\triangle$ 3 years old.	Makes the perfect gift for Schylling enthusiasts that are at least 3 years old.
His hat is accented with holly and holly berries .	His hat is accented with holly holly and berries.	His holly accented with holly and holly berries.

Table 2: The ablation experiment results of DSAE. $\triangle$ denotes percentage increase in PPL compared to the default setting. “First layer weights” indicates that the initial parameters for both the encoder and the decoder are transferred from the first layer of the base LLM, while “Last layer weights” are derived from the final layer.

Method	PPL↓	$\triangle$
Default	1.06	0.00%
w/o Segmentation	1.40	32.07%
w/o s_a	1.14	7.55%
w/o s_b	1.16	9.43%
w/o gate	1.21	14.15%
w/o LLM weights	1.24	16.98%
First layer weights	1.13	6.60%
Last layer weights	1.17	10.38%

attention to content words rich in semantic information, such as “various” in sample 1 and “pulsed” in sample 2, while assigning lower weights to function words that lack substantial meaning, such as “for”, “a”, and “of”. This indicates that DSAE can efficiently identify key information in sentences based on context and compress it into a compact latent representation, exhibiting particular advantages when processing longer sentences or those containing uncommon words. For instance, in sample 2, SVAE reconstructed “515” as “660”, while DSAE assigned greater weight to this token during semantic extraction and successfully reconstructed the original token. In sample 3, which contains the proper noun “Schylling”, DSAE assigned it the highest weight and accurately generated the word.

To validate the effectiveness of each component in DSAE, we conducted ablation studies on a model with hidden size 768 and 1 layer, with results shown in Tab. 2. Four key findings emerged:

(1) Among all ablated components, sentence segmentation yields the most pronounced effect on semantic compression. Without segmentation, PPL increases by 32% relative to the default setting. This demonstrates that forcing an entire sentence into a single latent representation creates a severe information bottleneck. In contrast, the proposed segmentation strategy assigns multiple latent vectors to longer sentences according to their token length, allowing the compressed representation to preserve richer semantic details.

(2) Static and dynamic latent representations provide complementary semantic information. We directly used the static latent representation s_a and dynamic latent representation s_b separately as sentence semantics for the decoder to reconstruct sentences. We found that when the static latent representation is absent, reconstructed sentences tend to miss words, while when the dynamic latent representation is absent, inconsistent words are reconstructed—both leading to performance degradation.

(3) The gating fusion mechanism is beneficial. In the gating ablation experiment, we simply summed the static latent representation s_a and dynamic latent representation s_b as the sentence latent representation. This simple summation strategy increases PPL by 14%. This confirms the importance of gated fusion of global and dynamic semantic information when extracting sentence semantics.

(4) LLM pre-trained parameters are suitable for semantic extraction. When initializing model parameters randomly instead of transferring from the LLM, performance dropped by 17%. This indicates that the language understanding capabilities acquired from LLM pre-training are well-suited for semantic extraction tasks. Additionally, we investigated the suitability of different LLM layers. When initializing encoder and decoder parameters with the first and last layers of the LLM respectively, performance decreased by 6% and 10%. This suggests that shallow layers align better with the encoder—both are responsible for extracting features from text—while deep layers align better with the decoder—both are responsible for generating text based on extracted features.

To further analyze how the decoder utilizes segment-level latent representations, we visualize the cross-attention weights between latent vectors and tokens for sentences of different lengths, as shown in Fig. 4. For short

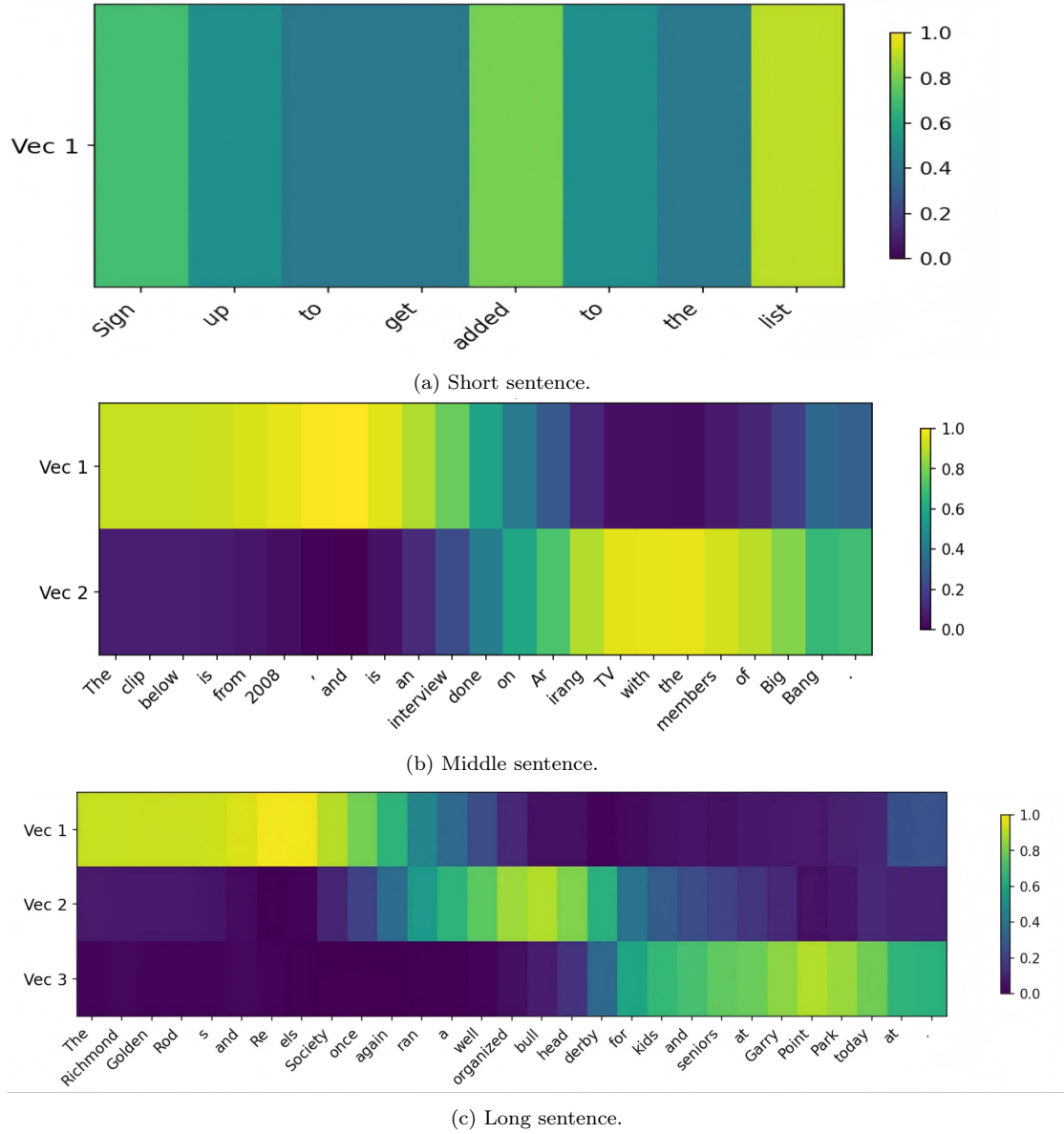


Figure 4: Attention heatmap comparison of decoder cross-attention weights on DSAE tokens for (a) short, (b) middle, and (c) long sentences during semantic compression. Lighter colors indicate higher attention weights, signifying that the decoder assigns greater importance to those tokens when reconstructing the original sentence.

Table 3: Experiment results of baseline methods and DSEI. DSEI-b-l denotes an OPT model with parameter size b, integrated with a DSAE consisting of l layers.

Model	Parameters(M)	PPL↓	Mean input throughput (k tokens/s)↑	Mean GPU memory (KB/token)↓
OPT-125M	125.23	26.94	13.93	131.45
SLLM-125M-H1	214.32	21.84	35.90	10.97
DSEI-125M-H1	214.92	11.17	35.98	13.05
SLLM-125M-H2	226.14	21.20	37.49	9.94
DSEI-125M-H2	226.76	15.52	37.76	13.18
SLLM-125M-H4	249.78	20.99	24.36	10.60
DSEI-125M-H4	250.37	14.57	23.50	13.56
OPT-350M	331.19	22.60	7.71	184.72
SLLM-350M-H1	429.46	21.51	99.20	25.21
DSEI-350M-H1	432.66	14.12	98.30	27.85
OPT-1.3B	1315.75	16.02	7.9	285.24
SLLM-1.3B-H1	1605.67	19.30	60.88	50.44
DSEI-1.3B-H1	1609.88	9.86	59.53	52.07

sentences, the single latent vector exhibits a broad attention pattern over the entire sentence, indicating that one compact representation is sufficient for reconstruction when the semantic content is compact. For medium and long sentences, different latent vectors exhibit clear and complementary attention patterns over different token regions. This suggests that length-based segmentation enables multiple latent vectors to capture localized semantic information, thereby reducing the burden on a single sentence-level vector and improving the fidelity of sentence reconstruction, thus pointing to a more effective design for long-text semantic compression.

4.3. Latent inference results

Tab. 3 presents a comparison between DSEI and existing baseline methods in terms of PPL, throughput, and GPU memory overhead. Experimental results demonstrate that DSEI achieves an optimal balance between inference efficiency and accuracy. Across various parameter scales of LLM backbones, DSEI consistently exhibits improved PPL performance compared to static sentence-level latent inference baseline SLLM.

It is worth noting that the quality improvement brought by DSEI is particularly pronounced on smaller-scale LLMs, while its throughput remains comparable to SLLM and its memory overhead remains substantially lower than token-level inference. On a 125M-parameter model, compared with SLLM, a single-layer DSEI configuration achieves a 48% reduction in PPL,

although the average input throughput increases only marginally by 0.22% and the mean GPU memory overhead increases by 18%. These observations suggest that DSEI may be particularly suitable for enhancing LLM inference performance in resource-constrained, practical deployment scenarios.

Furthermore, DSEI relies on a lightweight dynamic semantic extraction mechanism and introduces only approximately 0.6M additional parameters compared with SLLM. Despite this marginal parameter increase, it preserves the inherent efficiency advantages of semantic-level inference over token-by-token LLM inference in terms of both throughput and GPU memory overhead. Overall, these results demonstrate that DSEI provides a favorable trade-off between generation quality and inference efficiency.

5. Conclusion

In this paper, we propose the Dynamic Semantic Autoencoder (DSAE), which dynamically compresses sentences into segment-level latent representations. We integrate DSAE into Large Language Models (LLMs) via the Dynamic Semantic Extraction and Inference (DSEI) framework, enabling the LLM to conduct the entire inference process in the latent space. Our approach incorporates three key innovations: (1) employing a sentence segmentation mechanism to divide entire sentences into segments, thereby enhancing the model’s semantic extraction capability for long sentences; (2) efficiently extracting dynamic semantics by assigning semantic weights to tokens through a lightweight contextual awareness mechanism; and (3) utilizing a gating mechanism to fuse dynamic and static semantics, which preserves global information while eliminating redundancy, thereby striking a balance between performance and efficiency. Experimental results demonstrate that, compared to static semantic extraction baselines, DSAE achieves a maximum improvement of 34% in PPL, and DSEI reduces perplexity by 48%. These results demonstrate that DSEI provides a practical trade-off between generation quality and deployment efficiency. It improves latent-space inference quality while preserving the computational advantages of compressed semantic representations.

References

- [1] S. Hao, S. Sukhbaatar, D. Su, X. Li, Z. Hu, J. Weston, Y. Tian, Training large language models to reason in a continuous latent space, in: COLM, 2025.

- [2] W. Tan, J. Li, J. Ju, Z. Luo, R. Song, J. Luan, Think silently, think fast: Dynamic latent compression of llm reasoning chains, arXiv preprint arXiv:2505.16552 (2025).
- [3] Z. Shen, H. Yan, L. Zhang, Z. Hu, Y. Du, Y. He, Codi: Compressing chain-of-thought into continuous space via self-distillation, in: EMNLP, 2025.
- [4] H. An, Y. Chen, Z. Sun, X. Li, Sentencevae: Enable next-sentence prediction for large language models with faster speed, higher accuracy and longer context, arXiv preprint arXiv:2408.00655 (2024).
- [5] C. He, Z. Jin, C. Xu, J. Qiu, B. Wang, W. Li, et al., Wanjian: A comprehensive multimodal dataset for advancing english and chinese large models, arXiv preprint arXiv:2308.10755 (2023).
- [6] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: EMNLP, 2019.
- [7] T. Gao, X. Yao, D. Chen, Simcse: Simple contrastive learning of sentence embeddings, in: EMNLP, 2021.
- [8] J. Mu, X. Li, N. Goodman, Learning to compress prompts with gist tokens, in: NeurIPS, 2023.
- [9] Z. Pan, Q. Wu, H. Jiang, et al., Llmilingua-2: Data distillation for efficient and faithful task-agnostic prompt compression, in: ACL Findings, 2024.
- [10] H. Jiang, Q. Wu, C.-Y. Lin, et al., Llmilingua: Compressing prompts for accelerated inference of large language models, in: EMNLP, 2023.
- [11] T. Ge, H. Jing, L. Wang, et al., In-context autoencoder for context compression in a large language model, in: ICLR, 2024.
- [12] Z. Li, Y. Su, N. Collier, et al., 500xcompressor: Generalized prompt compression for large language models, in: ACL, 2025.
- [13] F. Liu, H. Qiu, Context cascade compression: Exploring the upper limits of text compression, arXiv preprint arXiv:2511.15244 (2025).

- [14] V. Meshchaninov, E. Chimbulatov, A. Shabalin, et al., Cosmos: Compressed and smooth latent space for text diffusion modeling, in: NeurIPS, 2025.
- [15] S. Gong, M. Li, J. Feng, Z. Wu, L. Kong, Diffuseq: Sequence to sequence text generation with diffusion models, in: ICLR, 2023.
- [16] H. Yuan, Z. Yuan, C. Tan, F. Huang, S. Huang, Seqdiffuseq: Text diffusion with encoder-decoder transformers, arXiv preprint arXiv:2212.10325 (2022).
- [17] A. Shabalin, V. Meshchaninov, E. Chimbulatov, et al., Tencdm: Understanding the properties of the diffusion model in the space of language model encodings, in: AAAI, 2025.
- [18] T. Wu, Z. Fan, X. Liu, H.-T. Zheng, Y. Gong, J. Jiao, J. Li, J. Guo, N. Duan, W. Chen, et al., Ar-diffusion: Auto-regressive diffusion model for text generation, NeurIPS (2023).
- [19] J. Lovelace, V. Kishore, C. Wan, E. Shekhtman, K. Q. Weinberger, Latent diffusion for language generation, NeurIPS (2023).
- [20] Y. Zhang, J. Gu, Z. Wu, S. Zhai, J. Susskind, N. Jaitly, Planner: Generating diversified paragraph via latent language diffusion model, NeurIPS (2023).
- [21] Y. Deng, Y. Choi, S. Shieber, From explicit cot to implicit cot: Learning to internalize cot step by step, arXiv preprint arXiv:2405.14838 (2024).
- [22] J. Xu, M. Zhou, W. Liu, H. Liu, S. Han, D. Zhang, Twt: Thinking without tokens by habitual reasoning distillation with multi-teachers' guidance, arXiv preprint arXiv:2503.24198 (2025).
- [23] S. Goyal, Z. Ji, A. S. Rawat, A. K. Menon, S. Kumar, V. Nagarajan, Think before you speak: Training language models with pause tokens, arXiv preprint arXiv:2310.02226 (2023).
- [24] J. Tack, J. Lanchantin, J. Yu, A. Cohen, I. Kulikov, J. Lan, S. Hao, Y. Tian, J. E. Weston, X. Li, Llm pretraining with continuous concepts, in: NeurIPS, 2025.

- [25] X. Chen, Z. Sun, G. Wenjin, M. Zhang, Y. Chen, Y. Sun, H. Su, Y. Pan, D. Klakow, W. Li, et al., Unveiling the key factors for distilling chain-of-thought reasoning, in: ACL, 2025.
- [26] N. Saunshi, N. Dikkala, Z. Li, S. Kumar, S. J. Reddi, Reasoning with latent thoughts: On the power of looped transformers, in: ICLR, 2025.
- [27] Y. Chen, J. Shang, Z. Zhang, et al., Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking, in: ACL, 2025.
- [28] J. Cheng, B. Van Durme, Compressed chain of thought: Efficient reasoning through dense representations, arXiv preprint arXiv:2412.13171 (2024).
- [29] D. Su, H. Zhu, Y. Xu, J. Jiao, Y. Tian, Q. Zheng, Token assorted: Mixing latent and text tokens for improved language model reasoning, in: ICML, 2025.
- [30] A. Mohtashami, M. Pagliardini, M. Jaggi, Cotformer: More tokens with attention make up for less depth, in: NeurIPS, 2023.
- [31] A. Mohtashami, M. Pagliardini, M. Jaggi, Cotformer: A chain of thought driven architecture with budget-adaptive computation cost at inference, in: ICLR, 2025.
- [32] B. Zeng, S. Song, S. Huang, Y. Wang, H. Li, Z. He, X. Wang, Z. Li, Z. Lin, Pretraining language models to ponder in continuous space, arXiv preprint arXiv:2505.20674 (2025).
- [33] H. Du, Y. Dong, X. Ning, Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts, arXiv preprint arXiv:2509.26314 (2025).
- [34] T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, T. Dao, Medusa: Simple llm inference acceleration framework with multiple decoding heads, in: ICML, 2024.
- [35] Z. Chen, A. May, R. Svirschevski, Y. Huang, M. Ryabinin, Z. Jia, B. Chen, Sequoia: Scalable, robust, and hardware-aware speculative decoding, arXiv preprint arXiv:2402.12374 (2024).

- [36] J. Ye, S. Gong, L. Chen, L. Zheng, J. Gao, H. Shi, C. Wu, X. Jiang, Z. Li, W. Bi, et al., Diffusion of thought: Chain-of-thought reasoning in diffusion language models, NeurIPS (2024).
- [37] Z. Huang, Z. Chen, Z. Wang, et al., Reinforcing the diffusion chain of lateral thought with diffusion language models, in: NeurIPS, 2025.
- [38] H. Wu, Z. Teng, K. Tu, Parallel continuous chain-of-thought with jacobi iteration, in: EMNLP, 2025.
- [39] X. Wei, X. Liu, Y. Zang, X. Dong, Y. Cao, J. Wang, X. Qiu, D. Lin, Sim-cot: Supervised implicit chain-of-thought, arXiv preprint arXiv:2509.20317 (2025).
- [40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, NeurIPS (2017).
- [41] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Doll, Focal loss for dense object detection, in: ICCV, 2017.
- [42] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al., Opt: Open pre-trained transformer language models, arXiv preprint arXiv:2205.01068 (2022).
- [43] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: ICLR, 2019.