DODO : Dynamic Contextual Compression for Decoder-only LMs

Guanghui Qin[η*], **Corby Rosset**[μ], **Ethan C. Chau**[μ], **Nikhil Rao**[μ], **Benjamin Van Durme**[η,μ]
[η]Johns Hopkins University [μ]Microsoft
{gqin2,vandurme}@jhu.edu

Abstract

Transformer-based language models (LMs) are inefficient in long contexts. We propose DODO , a solution for context compression. Instead of one vector per token in a standard transformer model, DODO represents text with *a dynamic number* of hidden states at each layer, reducing the cost of self-attention to a fraction of typical time and space. Moreover, off-the-shelf models such as LLAMA can be adapted to DODO by efficient parameter tuning methods such as LoRA. In use, DODO can act as either an autoregressive LM or a context compressor for downstream tasks. We demonstrate through experiments in language modeling, question answering, and summarization that DODO retains capabilities in these tasks, while drastically reducing the overhead during decoding. For example, in the autoencoding task, DODO shrinks context at a 20x compression ratio with a BLEU score of 98% for reconstruction, achieving nearly lossless encoding.

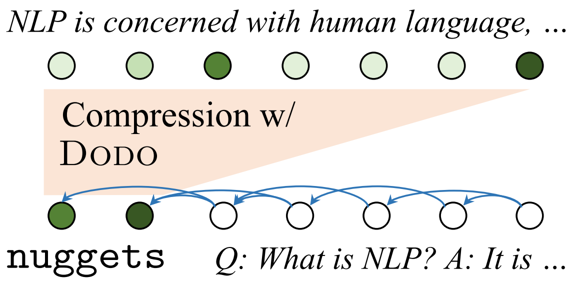

Figure 1: DODO efficiently maps long inputs into a compressed set of vectors named `nuggets` , which can then be attended to when processing a query.

1 Introduction

Transformer-based LMs (Vaswani et al., 2017) suffer from quadratic computational complexity w.r.t. sequence length, making it challenging to scale to long sequences. Proposed solutions (Tay et al., 2022) include sparsifying attention patterns (Beltagy et al., 2020; Ding et al., 2023) or approximating the attention computation with kernel methods (Choromanski et al., 2021). However, not all these approaches are proven effective for NLP tasks (Qin et al., 2023), and very few of them are applied to large language models (LLMs), such as LLaMA (Touvron et al., 2023a).

We propose DODO , a solution for dynamic contextual compression for decoder-only LMs. While a standard transformer represents a text with vector sequences of the same length as tokens, the intuition of DODO is to use *a smaller, variable number* of vectors as a contextual representation. Past research indicates that a subset of token embeddings, named `nuggets` , in an encoder with global attention may carry enough information to reconstruct surrounding context (Qin and Van Durme, 2023), and upon inspection those authors observed these `nuggets` tended to account for *preceding* text. This suggests a decoder-only model might be dynamically capable of deriving such a representation online (Fig. 1). To enable DODO requires addressing a selection process that is not differentiable: we adopt the straight-through estimator (Bengio et al., 2013) to make the model end-to-end trainable.

Past work on context compression, such as Ge et al. (2024) and Mu et al. (2023), appends *fixed additional tokens*. DODO *grows* the representation with sequence length and *re-uses* existing token embeddings. Moreover, unlike pattern-based methods that evenly chunk the text (Rae et al., 2020), experiments show that DODO spontaneously learns to use *textual delimiters* as `nuggets` , naturally splitting the text into subsentential units (Section 4.3).

DODO supports causal masking and can be naturally used as an autoregressive LM. We experimentally demonstrate that DODO can achieve a perplexity score lower than the original LM with restricted memory, outperforming the baseline model of Rae

[*]Work done in part during Guanghui Qin's internship at Microsoft Research.

et al. (2020). For tasks with a fixed context, e.g. long-form QA, DODO works as a context compressor: It encodes a token sequence into a shorter vector sequence, achieving a configurable compression ratio. In experiments on autoencoding, we demonstrate that DODO can achieve near lossless encoding with a compression ratio as high as 20x, a marked improvement over ICAE (Ge et al., 2024). After fine-tuning, DODO is effective in downstream NLP tasks such as question answering (QA) and summarization, where it performs on par with or even better than the original LMs while achieving a compression ratio as high as 10x.

In summary, we propose DODO for contextual compression for decoder-only transformers. It learns to subselect a fractional number of tokens as context representation. A straight-through estimator ensures that DODO is differentiable and can be trained with the next-token prediction objective. DODO achieves a remarkable compression ratio of up to 20x and is shown to be effective in tasks such as autoencoding, language modeling, and applications including QA and summarization.

2 Approach

In this paper, we study the language modeling problem $p(w_t \mid w_{<t})$, where $w_i \in V$ is a sequence of tokens and V is the vocabulary. The common Transformer (Vaswani et al., 2017) approach encodes a token sequence $w_{1:n}$ into a sequence of vectors and then predicts the next token:

$$\left(\mathbf{x}_1^L, \mathbf{x}_2^L \ldots, \mathbf{x}_n^L\right) = \texttt{Transformer}_\theta(w_{1:n}), \quad (1)$$

$$p(w_{n+1} \mid w_{1:n}) \sim \texttt{LMHead}_\theta(\mathbf{x}_n^L), \quad (2)$$

where θ is the parameter, L is the number of transformer layers, $\mathbf{x}_t^L \in \mathbb{R}^d$ is the hidden state of the t-th token in the L-th layer, d is the hidden state dimension, and LMHead is a feedforward neural network that defines a categorical distribution over the vocabulary. In the decoder-only transformers, $\mathbf{x}_t^{l+1}$ is encoded by attending to past token representation in the l-th layer:

$$\mathbf{x}_t^{l+1} = \texttt{Attn}_\theta(\mathbf{x}_t^l, \mathbf{x}_{1:t}^l), \; l = 1, 2, \ldots, L{-}1 \quad (3)$$

where the Attn function takes query and key (value) vectors as arguments. Eq. (3) can be inefficient with long sequences as its computation grows quadratically with the sequence length. In this paper, we aim to answer: *Can we find an alternative method to efficiently approximate* $\mathbf{x}_t^l$ *?*

2.1 Representing texts with DODO

In Eq. (3), context information up to the t-th token is encoded into t vectors as hidden states. Intuitively, we can reduce the computational overhead by controlling the size of hidden states. Formally, we want to encode t tokens $w_{1:t}$ into k vectors: $(\mathbf{z}_1^l, \ldots, \mathbf{z}_k^l)$, where $k \leq t$. Following prior work (Qin and Van Durme, 2023) we refer to these vectors as `nuggets`. Then $\mathbf{x}_t^{l+1}$ is derived by

$$\mathbf{x}_t^{l+1} = \texttt{Attn}_\theta(\mathbf{x}_t^l, \mathbf{z}_{1:k}^l), \; l = 1, 2, \ldots, L{-}1. \quad (4)$$

Please note that k *is not a fixed number* (Zhang et al., 2022; Ge et al., 2024) *but a dynamic number that depends on the input sequence* $w_{1:t}$. We will discuss the choice of k later.

We observe that $\mathbf{x}_{1:t}^l$ encodes the information of tokens $w_{1:t}$, thus one may derive $\mathbf{z}_{1:k}^l$ from $\mathbf{x}_{1:t}^l$. We therefore select $\mathbf{z}_{1:k}^l$ by *subselecting vectors* from $\mathbf{x}_{1:t}^l$. Formally, we have (c.f. §3.3 in Zeng et al., 2023b and §3.1 in Qin and Van Durme, 2023):

$$\{\mathbf{z}_1^l, \ldots, \mathbf{z}_k^l\} = \{\mathbf{x}_i^l \mid \alpha_i = 1, 1 \leq i \leq t\}, \quad (5)$$

$$p(\alpha_i = 1) = \sigma(\texttt{Scorer}_\varphi(\mathbf{x}_i^\iota)), \quad (6)$$

where α_i is a binary variable indicating if $\mathbf{x}_i^l$ is selected, $p(\alpha_i = 1)$ refers to a Bernoulli distribution, $\texttt{Scorer}_\varphi$ is a feedforward neural network parameterized by φ, and σ is the sigmoid function. $\texttt{Scorer}_\varphi$ takes as input $\mathbf{x}_i^\iota$, the hidden state of w_i in the ι-th layer, where ι is a hyperparameter. [1] That is, tokens that were assigned with higher scores by Scorer is more likely be selected as `nuggets`.

Note that ι in Eq. (6) does not depend on l, thus it selects the same set of indices for all the layers. In the remainder of this paper, we abstract the process of Eqs. (1) and (4) to (6) into a Dodo operator:

$$\mathbf{z}_{1:k}^{1:L} = \texttt{Dodo}_{\theta,\varphi}(w_{1:t}), \quad 1 \leq k \leq t. \quad (7)$$

We may omit the superscript and use $\mathbf{z}_i$ ($\mathbf{x}_i$) to indicate $\mathbf{z}_i^{1:L}$ ($\mathbf{x}_i^{1:L}$), the i-th `nuggets` in all layers.

So far, we only assume that k is a dynamic number depending on $w_{1:t}$. In general, we set k to be roughly proportional to t, controlled by a compression ratio $r \approx t/k$. Depending on the task, k can either grow with t when $w_{1:t}$ is incrementally observed (Section 2.2), or be strictly proportional to t when $w_{1:t}$ is fully observed (Section 2.3).

[1] We empirically set $\iota = 3$ in all experiments.

2.2 Dodo as an autoregressive LM

Not all efficient LMs support causal masking (Peng et al., 2022). Many context compression methods (Mu et al., 2023; Ge et al., 2024) only apply to fixed-sized texts. However, each hidden state $\mathbf{z}_i$ in `nuggets` only conditions on its past tokens. Thus Dodo can be naturally integrated into an autoregressive LM, where tokens $w_{1:t}$ are sequentially fed into an LM. Instead of saving all past hidden states $\mathbf{x}_{1:t}$, Dodo only retains a subset of tokens as `nuggets`, which are selected by `Scorer`. The stochastic selection process in Eq. (5) is made deterministic by settings a threshold Λ in Eq. (6):

$$\alpha_i = \mathbb{1}\left\{\texttt{Scorer}_{\varphi}(\mathbf{x}_i^{\iota}) > \Lambda\right\}, \tag{8}$$

where $\mathbb{1}$ is the indicator function. That is, token w_i is retained as `nuggets` $\mathbf{z}_j$ if its score is above the threshold Λ. Because Eq. (8) does not depend on future tokens, $\mathbf{z}_{1:k}$ can be autoregressively encoded with causal masking.

To set a proper threshold Λ, we define a compression ratio $r \geq 1$ and let $r \approx t/k$. That is, Λ should be set such that after t tokens are fed into Dodo, roughly $k \approx t/r$ hidden states $\mathbf{x}_i$'s should be selected as $\mathbf{z}_j$'s. In practice, we estimate the threshold Λ by running a trained $\texttt{Scorer}_{\varphi}$ on sampled tokens. [2]

Parameter configuration Intuitively, as a compressed representation, $\mathbf{z}_j$ should encode a broader range of tokens than $\mathbf{x}_i$ does. We therefore separate their attention parameters: Once a token w_t is selected by Eq. (8), it uses $\texttt{Attn}_{\phi}$ to attend past tokens. Otherwise, it uses $\texttt{Attn}_{\theta}$.

A mixed resolution Though $\mathbf{z}_{1:k}$ is more efficient than $\mathbf{x}_{1:t}$, information loss is inevitable during the subselection process. Intuitively, the tokens closer to the target token w_{t+1} contain more relevant information. We propose to revise Eq. (4) with a mixed resolution, where $\mathbf{x}_t$ *attends to recent* τ *tokens without compression*. Suppose we split the sequence $w_{1:t}$ at index $(t-\tau)$, we have

$$\mathbf{x}_t^{l+1} = \texttt{Attn}_{\theta}\left(\mathbf{x}_t^l, \left[\mathbf{z}_{1:k}^l; \mathbf{x}_{t-\tau:t}^l\right]\right), \tag{9}$$

$$\mathbf{z}_{1:k} = \texttt{Dodo}_{\phi,\varphi}(w_{1:t-\tau}) \tag{10}$$

where $\mathbf{z}_{1:k}$ are the compressed representation of $w_{1:t-\tau}$, $[\ ;\ ]$ indicates the concatenation of vector sequences, and τ is a hyperparameter. An illustration of our method can be seen in Fig. 2.

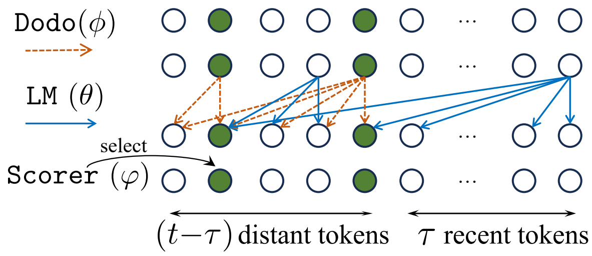

Figure 2: An illustration of the autoregressive Dodo, where `Scorer`(φ) selects `nuggets` tokens, `Dodo`(ϕ) aggregates the information of $(t-\tau)$ distant tokens into `nuggets`. When predicting a new token, the `LM`(θ) has direct access to recent τ tokens but needs to use `nuggets` to access the distant information.

Learning To train Dodo as an autoregressive LM, we estimate the parameters (θ, ϕ, φ) to maximize the log likelihood of $p(w_{1:n})$:

$$\max_{\theta,\phi,\varphi} \sum_{w_{1:n}\in\mathcal{D}} \sum_{i=1}^{n-1} \log p(w_{i+1} \mid w_{1:i}), \tag{11}$$

where $\mathcal{D}$ is the corpus and $p(w_{i+1} \mid w_{1:i})$ is defined by Eqs. (2), (9) and (10).

Learning with Eq. (11) can be inefficient: The computation cannot be parallelized on the sequence dimension because they have different splitting index $(i-\tau)$. As an efficiency optimization, we chunk the texts into segments, and tokens in a segment share the same splitting index.

2.3 Dodo as a contextual compressor

In some tasks, such as long-form question answering, a fixed segment text, say $w_{1:n}$, is treated as the context and is fully observed before the text generation. In this case, one can use Dodo as an encoder [3] to encode the input text into hidden states $\mathbf{z}_{1:k}$ where $k \leq n$.

Formally, suppose $w_{1:n}$ and $y_{1:m}$ are the input and output sequences separately, the probability distribution of $y_{1:m}$ is defined as

$$p(y_i \mid y_{<i}, w_{1:n}) \sim \texttt{LMHead}_{\theta}\left(\mathbf{y}_i^L\right), \tag{12}$$

$$\mathbf{y}_i^{l+1} = \texttt{Attn}_{\theta}\left(\mathbf{y}_i^l, \left[\mathbf{z}_{1:k}^l; \mathbf{y}_{1:i}^l\right]\right), \tag{13}$$

where we slightly abuse the notation to use $\mathbf{y}_i$ as the hidden states of token y_i. Refer to Fig. 3 for an illustration of Eq. (13).

[2] Training $\texttt{Scorer}_{\varphi}$ requires a determined Λ, but setting Λ needs a trained $\texttt{Scorer}_{\varphi}$. To prevent the chicken-and-egg problem, we initialize the $\texttt{Scorer}_{\varphi}$ here from Section 2.3.

[3] We use the term "encoder" because it encodes an input sequence. It is technically a decoder-only transformer model.

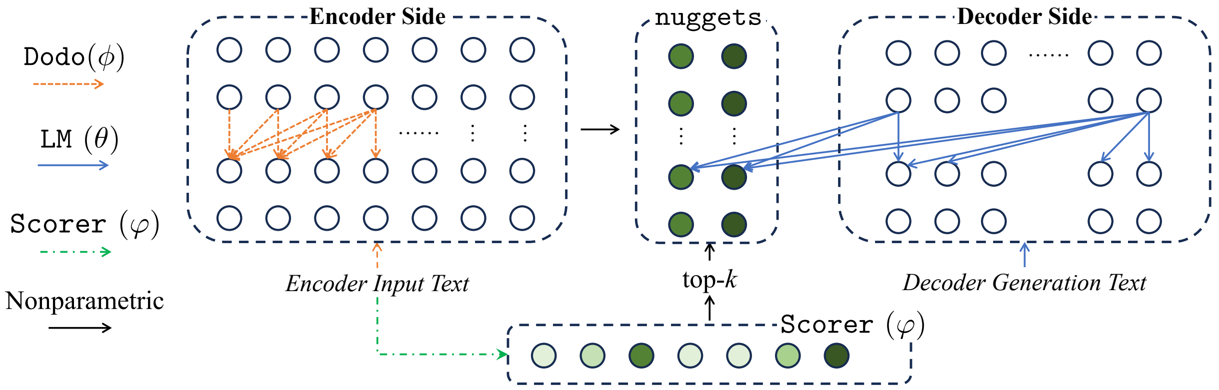

Figure 3: DODO as context compressor. From left to right, **Encoder side**: $\texttt{Dodo}_\phi$ encodes texts into vectors representations; **Scorer**: $\texttt{Scorer}_\varphi$ computes a score for eaceh encoder token and then select the top-k tokens as `nuggets`; **Decoder side**: Language model $\texttt{LM}_\theta$ autoretressively decodes text conditioned on `nuggets`.

Because n, the number of input tokens, is known, we could maintain a fixed compression $r = n/k$ by setting $k = \lceil n/r \rceil$. We therefore make the stochastic selection in Eq. (6) deterministic by:

$$\{\mathbf{z}_1, \ldots, \mathbf{z}_k\} = \texttt{TopK}(\mathbf{x}_{1:n}, s_{1:n}, k), \tag{14}$$

$$s_i = \texttt{Scorer}_\varphi(\mathbf{x}_i^l), \tag{15}$$

where TopK selects k vectors from $\mathbf{x}_{1:n}$ with the highest s_i, the score of token w_i. [4]

Parameter configuration We assign separate parameters to the attention modules in the encoder and decoder: The parameters of the encoder (decoder) are indicated by ϕ (θ).

Learning To train DODO as an encoder, we learn it through maximum likelihood estimation:

$$\max_{\theta,\phi,\varphi} \sum_{w,y\in\mathcal{D}} \sum_{i=1}^{m} \log p\left(y_i \mid y_{<i}, w_{1:n}\right),$$

where input and output sequence pairs $(w_{1:n}, y_{1:m})$ are sampled from a corpus $\mathcal{D}$, and the next-token probability is defined by Eqs. (12) to (15).

2.4 Learning with straight-through estimator

The selection of $\mathbf{z}$ is discrete: the selection process, Eqs. (8) and (14), is *not differentiable*. Here we show how to back-propagate the gradients so the parameter φ in $\texttt{Scorer}_\varphi$ can be learned.

Previous work proposed approaches to make TopK differentiable (e.g., Xie et al., 2020 and Sander et al., 2023). To avoid unnecessary complexity, we adopt the biased but simpler straight-through estimator of Bengio et al. (2013). Suppose the token $\mathbf{x}_j$ attends to the compressed representation $\mathbf{z}_i$, and let $\xi_{i,j}$ denote the logit of the attention token $\mathbf{x}_i$ to the compressed hidden state $\mathbf{z}_j$. Then we have (c.f. §3.2 in Qin and Van Durme, 2023 and §2.2 in Jang et al., 2017):

$$\xi_{i,j}^l = \left(\mathbf{W}_\text{Q}\mathbf{x}_j^l\right)^\top \left(\mathbf{W}_\text{K}\mathbf{z}_i^l\right), \tag{16}$$

$$\frac{\partial \ell}{\partial s_i} \leftarrow \sum_j \sum_{l=1}^{L} \frac{\partial \ell}{\partial \xi_{i,j}^l}, \tag{17}$$

where $\mathbf{W}_\text{Q}$ and $\mathbf{W}_\text{K}$ are parameters of the self-attention, and $\partial\ell/\partial s_i$ is set to be the aggregation of the gradients of $\xi_{i,j}^l$ from future tokens in all layers. Intuitively, $\texttt{Scorer}_\varphi$ learns to select tokens that are more attended by future tokens. To implement Eq. (17), we replace $\xi_{i,j}^l$ in Eq. (16) with:

$$\overline{\xi}_{i,j}^l = \xi_{i,j}^l + s_i - \texttt{StopGrad}(s_i), \tag{18}$$

where the $\texttt{StopGrad}(s_i)$ detaches s_i from backward pass and ensures that the addition of s_i to $\xi_{i,j}^L$ does not affect the forward pass.

3 Overall experiment setup

We adopt the decpder-only transformer architecture of LLAMA (Touvron et al., 2023a,b) as our base model. For the autoencoding experiment, we use the checkpoint of `LLaMA-7B` following the baseline model ICAE (Ge et al., 2024). We use the checkpoint of `LLaMA-2-7B` for the autoregressive language modeling experiments (Section 5) and `LLaMA-2-7B-chat` (Section 6) for the downstream NLP tasks.

We adopt LORA (Hu et al., 2022) with a rank of 32 to fine-tune the parameters of the LM, namely

[4] Because $\mathbf{x}_i$ only encodes texts before w_i, the last token w_n is always selected to the information in $w_{1:n}$ is completely encoded in $\mathbf{z}_{1:k}$.

θ and ϕ. We employ mixed precision to save GPU memory. The training is scaled up to 16 NVIDIA V100 cards with DeepSpeed (Rasley et al., 2020). See Appendix B for further training details, including hyperparameters, and parameter counts.

4 Autoencoding experiment

4.1 Task, dataset, and experiment setups

In this section, we use DODO as a context compressor (Section 2.3) and apply it to the autoencoding task. As a comparison, we use In-Context AutoEncoder (Ge et al., 2024, ICAE) as a baseline model. In this task, a model is asked to reconstruct the input text from a compressed representation. Following ICAE, we fine-tune the `LLaMA-7B` model on the Pile (Gao et al., 2020) dataset. We manually split the corpus into train, dev, and test splits, and train the model until convergence.

As stated in Section 2.3, we use DODO to compress the input text into fewer hidden states $\mathbf{z}$, and then use the LM to decode the input sequence. The size of hidden states $\mathbf{z}$, i.e. k, is set to be proportional to the length of the input sequence: $k = n/r$, and we set $r = 20$ and 10. We prepend a trainable soft token to the decoding sequence to signal the model to reconstruct inputs (Ge et al., 2024).

The key idea of ICAE is to append 128 tokens to the input sequence as "memory slots," and train the decoder to reconstruct the input from the memories:

$$(\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \ldots, \tilde{\mathbf{m}}_{128}) = \mathtt{LM}\left([w_{1:n}; m_{1:128}]\right)$$
$$p(w_{i+1} \mid w_{1:i}) = \mathtt{LM}\left([w_{1:i}; \tilde{\mathbf{m}}_{1:128}]\right).$$

We measure using BLEU (Papineni et al., 2002) score on pairs of input and decoded texts. [5]

4.2 Experiment results

In Fig. 4 we see DODO has comparable performance with the ICAE baseline for short sequences and better performance for long sequences. Moreover, DODO successfully handles longer inputs: performance improves on longer sequences because the number of `nuggets` is proportional to the sequence length, unlike ICAE's constant-sized memory. Despite its variable memory, DODO maintains an advantage over ICAE in computational time and space. First, DODO *encodes* sequences more efficiently: while ICAE always *appends* 128 tokens, DODO *reuses* a fraction of the already-encoded tokens. Also, DODO *uses fewer tokens* than ICAE: even for the longest sequences, DODO only uses 25 or 50 tokens, while ICAE uses 128 for all sequences. [6] Lastly, DODO is more efficient than ICAE during *decoding* because it uses fewer tokens and does not need to re-encode them. In short, compared to the baseline, DODO demonstrates comparable or better performance, successful handling of long sequences, and much more efficient encoding and decoding.

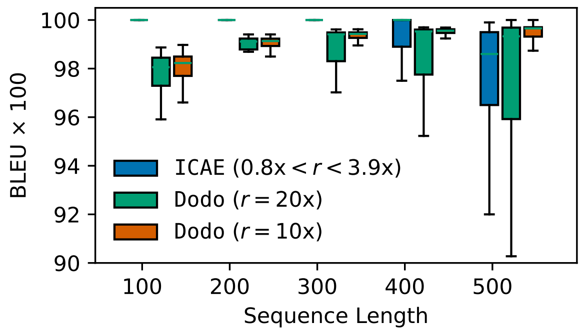

Figure 4: BLEU scores for autoencoding. Each group corresponds to a sequence length (±5 tokens). Note the performance of ICAE is nearly 100% for sequence lengths shorter than 300.

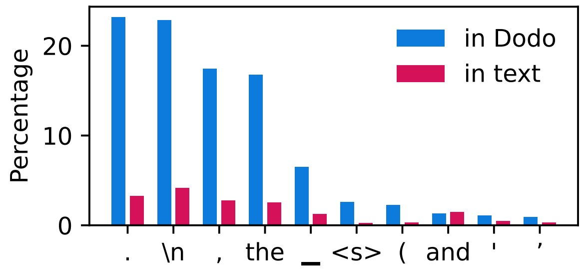

Figure 5: Token frequency of tokens selected by DODO and the formal texts. These top 10 token types cover 95% of the observed selection.

4.3 DODO selects clausal text delimiters

In Section 2.1, we employ `Scorer` to pick out `nuggets`, but what are the actual tokens selected? We empirically sampled 128 documents with 50k tokens and run the `Scorer` from the checkpoint in Section 4 with a compression ratio of 10, and the results are shown in Fig. 5. Readers may refer to Appendix C for case studies on sampled texts. From Fig. 5, we observe similar phenomena as Qin and Van Durme (2023), where the tokens preferred by DODO are mostly clausal text delimiters, such as punctuation marks and conjunction words.

[5] We report ICAE results per the §3.3.1 in Ge et al. (2024).

[6] DODO uses all layers while ICAE only uses the last layer. However, ICAE needs to encode their memory tokens into hidden states during decoding, while DODO can save this step.

. . . In the 1890s, armed standoffs were avoided narrowly several times. The Great Northern Railway, under the supervision of president . . . (omitted 230 tokens) . . . The railway also built Glacier Park Lodge, adjacent to the park on its east side, and the Many Glacier Hotel on the east shore of Swiftcurrent Lake. Louis Hill personally selected the sites for all of these buildings, choosing each for their dramatic scenic backdrops and views. Another developer, John Lewis, built the Lewis Glacier Hotel on Lake McDonald in 1913–1914. **The Great Northern Railway** bought the hotel in 1930 and it was later . . .

Figure 6: An example of a setting of our LM experiment. Here, compressive models access 320 tokens of history (italics) which they must compress to 32 states, along with 32 explicit tokens of most recent history (final portion of red, normal text). FULL gets explicit access only to the entirety of the red text (64 tokens), with no access to longer history. Models need to complete the sequence starting with **The Great Northern Railway**.

model	total states	compressed tokens	context tokens	ppl. on WikiText *subword*	ppl. on WikiText *word*	ppl. on Pile *subword*
FULL	256	0	256	6.39	10.65	4.94
COMPRESSIVE	256	1280	128	6.88	11.62	4.82
DODO	256	1280	128	**6.30**	**10.55**	**4.01**
FULL	128	0	128	6.87	11.69	5.35
COMPRESSIVE	128	640	64	7.09	12.18	4.93
DODO	128	640	64	**6.58**	**11.06**	**4.49**
FULL	64	0	64	7.95	14.08	5.80
COMPRESSIVE	64	320	32	7.64	13.39	5.65
DODO	64	320	32	**6.91**	**11.78**	**5.01**

Table 1: Perplexity on the Pile and WikiText-103, contrasting two 10x compressed solutions against no use of compression. **Compressed tokens**: the number of compressed tokens that precede **context tokens**: the uncompressed context immediately before the token to be predicted. This adds up to **total state**, which is directly comparable between systems, using three settings (256, 128, and 64). DODO trades off explicit context for larger history, with better perplexity results.

5 Autoregressive LM experiment

5.1 Experiment setup

In this task, the model is asked to *autoregressively* decode a sequence of texts. We therefore use DODO as an autoregressive LM (Section 2.2). We introduce a baseline method Compressive Transformers (Rae et al., 2020) (denoted by COMPRESSIVE), which evenly chunks the text into segments and uses a pooling algorithm [7] to compress the hidden states of each segment into a single vector. We also conduct experiments with the original LLAMA, denoted by FULL . In experiments, COMPRESSIVE has the save compression ratio as DODO does. FULL does not support compression, so we limit its context length to make sure all models use the same number of hidden states.

We use the Pile (Gao et al., 2020) and WikiText-103 (Merity et al., 2017) as the corpus. We randomly split the Pile into train, dev, and test sets, where the test set contains 100k tokens. All models are initialized from the checkpoint `Llama-2-7b`, and trained on the training set of the Pile until convergence. The compression ratio for DODO and COMPRESSIVE is 10x. The evaluation is conducted on the test set of the Pile and WikiText-103.

[7]In experiments, we adopt the mean pooling.

Perplexity (PPL) is used as the evaluation metric. Following previous work, we exclude the words that are defined as out-of-vocabulary by Merity et al. (2017) from the evaluation on WikiText-103. Because WikiText-103 is a tokenized corpus, we take production over the probabilities of subwords for each complete word to measure the word PPL. Note our algorithm underestimates the model performance for the complete word PPL.

We illustrate the intuition of DODO via an example in Fig. 6. For such an example, DODO should retain both topical and explicit vocabulary information (e.g., the underlined text) in the compressed history, in order to be less surprised by subsequent text such as bolded there.

5.2 Experiment results

The experiment results are shown in Table 1. We conduct experiments with 3 context configurations, where an LM has access to up to 64, 128, or 256 past hidden states. For DODO and COMPRESSIVE , the first 32, 64, or 128 states are compressed representation of the past 320, 640, or 1280 tokens. DODO outperforms both COMPRESSIVE and FULL , showing that with a restricted size of hidden states, DODO is an effective method to encode history information.

Dataset	Split sizes			Text length		
	train	dev	test	doc	query	answer
SQuAD (Rajpurkar et al., 2016)	88k	10.5k	-	231	17.0	-
CNN/DailyMail (See et al., 2017)	287k	13.4k	12k	878	-	68.9

Table 2: Dataset statistics. The text lengths are counted by the LLaMA tokenizer.

6 Downstream task experiments

We pick downstream tasks where a document as context is followed by a query. The model is asked to encode the document and decode the answer conditioned on the document encoding and question. In these tasks, we use DODO as a context compressor (Section 2.3), and we set the compression $r = 5$ or 10. To train DODO to perform these tasks, we consider 2 scenarios. a) **Fine-tuning**: DODO is trained on the training set of the downstream tasks. b) **Zero-shot**: DODO is trained on normal texts randomly sampled from the Pile and directly tested on the downstream task. In this case, each text is split into 2 parts, containing up to 512 and 128 tokens, and the model is asked to decode the second part conditioned on the encoding of the first part.

We consider the tasks of question answering and summarization. Datasets used in this section are SQuAD (Rajpurkar et al., 2016) and CNN/DailyMail v3.0.0 (See et al., 2017) for summarization. Their statistics are listed in Table 2.

We use the following baseline methods:

- FULL : Results of the original LM.
- NODOC : LM is used to do the task without any documents. Only the question is provided.
- LMSUMM : Use the LM to summarize the text into fewer tokens with prompts, which asks the LM to compress the texts into 10% of its length. LM uses the summary instead of documents to do the task. (Appendix D.1) [8]

6.1 Question answering

Model	cmpr.	accuracy
NODOC	∞	1.4
LMSUMM	10x	30.9
FULL	1x	**64.5**
DODO	5x	59.1
DODO	10x	49.8

Table 3: The accuracy of all 4 models on the task of SQuAD. Cmpr. is the compression ratio of the method.

In SQuAD a model is asked to extract a phrase from the passage to answer the query. We reformulate this problem as a text-to-text task instead of annotation and prompt the model to answer the question (Appendix D.2). We use accuracy to evaluate the model performance. As the model tends to generate tokens more than the answer itself or using different forms (e.g. using "two" instead of "2"), we normalize the output to match the answer. Readers may refer to Appendix E for the algorithm used to calculate the accuracy.

We consider all models: FULL , LMSUMM , DODO , and NODOC (Table 3). All models are evaluated in a zero-shot manner without fine-tuning. FULL and DODO easily outperform the NODOC and LMSUMM , and we observe that LMSUMM often omits details that are needed by the question. The performance of DODO can be improved by lowering its compression ratio, and the performance of DODO ($r = 5$) is close to FULL , confirming a compressed representation can still support LLM reasoning.

6.2 Summarization

model	cmpr.	R1	R2	RL
FULL (zero-shot)	1x	32.5	9.7	28.2
FULL (fine-tuning)	1x	37.7	**15.6**	35.3
DODO	10x	**39.9**	14.6	**37.0**

Table 4: The Rouge scores (F_1 of Rouge-1, Rouge-2, LCS) of FULL and DODO on CNN/DailyMail.

CNN/DailyMail contains news articles, where a model is required to generate a short summary. As no query is involved, we propose a prompt as a statement of the task requirement (Appendix D.3).

We consider FULL and DODO ($r = 10$). FULL is evaluated in both zero-shot and fine-tuning settings and DODO is fine-tuned. The results are shown in Table 4. We find that DODO can achieve similar or even better performance than FULL after compression. We speculate that as the context of CNN/DailyMail is long, this may lead the LM to be "lost in the middle" (Liu et al., 2024), whereas the `nuggets` generated by DODO is only 10% of the original length and perhaps less susceptible. This is an interesting avenue for future exploration.

[8] In practice, LM uses 10.9% of its original length to summarize the text on average, counted by subwords.

7 Discussion

7.1 Optimal `nuggets` selection

In DODO, `Scorer` selects tokens as `nuggets` and is learned through the residual connection introduced in Section 2.4. With gradient signal from the self-attention, `Scorer` tends to select the tokens that are mostly attended by the decoder. Isolating the other parts of the model, how can we evaluate the performance of `Scorer` itself?

To facilitate the discussion, let $\mathcal{I}$ be the selection conducted by `Scorer`. We use $\mathcal{I}^*$ to denote the *theoretically optimal* `nuggets` *selection*, which is defined as the selection that achieves the best performance in a task, e.g. the lowest perplexity in the LM task. To evaluate $\mathcal{I}$, we ask: How similar are $\mathcal{I}$ and $\mathcal{I}^*$? What is their performance gap?

Unfortunately, finding the optimal selection $\mathcal{I}^*$ is a non-trivial combinatorial problem, so we propose a greedy algorithm to approximate $\mathcal{I}^*$. Due to the space limit, we leave the details of this algorithm and our experiment design to Appendix A. As the results, the overlapping between $\mathcal{I}$ and $\mathcal{I}^*$ is roughly 75.3%, meaning the `nuggets` selected by `Scorer` are very close to the theoretical optimal selection. Replacing $\mathcal{I}^*$ with $\mathcal{I}$ will sacrifice 7.9% of the performance in terms of LM perplexity, so we conclude that `Scorer`, though not being optimal, can achieve a close-to optimal performance through the straight-through estimator.

7.2 Related work

Scaling transformers to long sequences is a popular topic in the NLP community (Tay et al., 2022). Existing work includes sparsify the attention patterns (Beltagy et al., 2020; Zaheer et al., 2020; Khalitov et al., 2023; Ding et al., 2023; Ainslie et al., 2023; Rae et al., 2020), employing low-rank or kernel methods to approximate the attention matrix computation (Choromanski et al., 2021; Katharopoulos et al., 2020), or applying recurrence (Dai et al., 2019; Yang et al., 2019; Bulatov et al., 2022). Another line of work tries to extrapolate the ability of LMs to long contexts, such as using linear bias (Press et al., 2022) or rotary position embeddings (Su et al., 2024). Recently, Bertsch et al. (2023); Tworkowski et al. (2023) applied kNN search to select a subset of tokens for attention at each layer of an encoder-decoder transformer, effectively extending the attention range of transformers. Zeng et al. (2023b) proposed to compress the context by prioritizing the "VIP tokens", which are important to certain tasks and can be saved in specialized data structure.

Past work on efficient transformers, as shown above, mainly improve the efficiency of the self-attention. DODO instead addresses a language representation problem: It shortens the length of the sequences in the space of hidden states. From this perspective, the idea of DODO is orthogonal to most of the efficient self-attention methods, and thus can be jointly applied with most of them, e.g. kNN based methods (Tworkowski et al., 2023).

In the context of large language models, recent work focuses on compressing the prompt tokens into soft embeddings (Mu et al., 2023; Wingate et al., 2022) or encoding the supporting documents (Ge et al., 2024; Chevalier et al., 2023) into fewer vectors. Some recent work tries to train LLMs with longer contexts, such as Li et al. (2023), GLM (Zeng et al., 2023a), and Claude 2 (Anthropic, 2023). Notably, Xiong et al. (2023) continue to train LLAMA to study the relationship between model performance and context length.

Researchers also explored retrieval-based methods that infuse knowledge into LM decoding, some notable work in this field includes FiD (Izacard and Grave, 2021), REALM (Guu et al., 2020), KNN-LM (Khandelwal et al., 2020), and RAG (Lewis et al., 2020). From the angle of the LLMs, Zheng et al. (2023) found that providing contexts to LLMs can help them generate truthful answers.

8 Conclusion

In this work, we propose DODO, a method for contextual compression for decoder-only transformers. In language modeling (Section 5) and summarization (Section 6.2), DODO is shown to generate a highly condensed representation of the context, while the results in autoencoding (Section 4) and question answering (Section 6.1) reflect that the details of the contexts can be recovered from `nuggets`. Moreover, in Section 6.1 we show that DODO trained with text continuation preserves the capability of instruction following. This demonstrates LLMs can encapsulate more of their input into fewer hidden states than previously realized, suggesting a new direction for efficient foundation models. Future work will explore more specialized versions of this proposal for optimizing results on individual applications, such as in dialog, supervised fine-tuning, reinforcement learning with human feedback, and in-context learning.

9 Ethical statement and limitations

Used artifacts In this work, we used the publicly released codes and checkpoints of LLaMA. Per the license attached to LLaMA, we agree not to re-distribute their parameters and limit the usage of the models for the research purpose only.

Potential societal risks Because we only trained LLaMA on general texts, we do not think that our paper will have any additional societal impacts beyond the checkpoints, except for the privacy issues mentioned below.

Privacy issues on the datasets Our method further fine-tunes LLaMA on the Pile (Gao et al., 2020). Given the size of the Pile (Gao et al., 2020) is huge (around 800GB), we are unable to conduct effective investigation on the privacy issue on the corpus. We refer readers to Gao et al. (2020) for the discussion of the potential issues inside the data.

Language bias We adopt the checkpoint of LLaMA, which was mainly trained on English corpus. For that reason, our method is biased toward English. However, we do not believe our findings are specific to English, but its effectiveness on other languages remains for future exploration.

References

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. 2023. CoLT5: Faster Long-Range Transformers with Conditional Computation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Anthropic. 2023. Claude 2.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. 2022. Recurrent Memory Transformer. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting Language Models to Compress Contexts. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2021. Rethinking Attention with Performers. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, and Furu Wei. 2023. LongNet: Scaling Transformers to 1,000,000,000 Tokens.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context Autoencoder for Context Compression in a Large Language Model. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. In *Proceedings of International Conference on Machine Learning (ICML)*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of Annual Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Fran¸cois Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear

Attention. In *Proceedings of International Conference on Machine Learning (ICML)*.

Ruslan Khalitov, Tong Yu, Lei Cheng, and Zhirong Yang. 2023. ChordMixer: A Scalable Neural Attention Model for Sequences with Different Lengths. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Quentin Lhoest, Albert Villanova Del Moral, Yacine Jernite, Abhishek Thakur, Patrick Von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario ˇ Saˇ sko, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, Fran¸cois Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A Community Library for Natural Language Processing. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023. How Long Can Context Length of Open-Source LLMs truly Promise? In *Proceedings of Workshop on Instruction Tuning and Instruction Following*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics (TACL)*.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to Compress Prompts with Gist Tokens. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. 2022. ABC: Attention with Bounded-memory Control. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Guanghui Qin, Yukun Feng, and Benjamin Van Durme. 2023. The NLP Task Effectiveness of Long-Range Transformers. In *Proceedings of Annual Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Guanghui Qin and Benjamin Van Durme. 2023. Nugget: Neural Agglomerative Embeddings of Text. In *Proceedings of International Conference on Machine Learning (ICML)*.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2020. Compressive Transformers for Long-Range Sequence Modelling. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD)*.

Michael E. Sander, Joan Puigcerver, Josip Djolonga, Gabriel Peyre, and Mathieu Blondel. 2023. Fast, Differentiable and Sparse Top-k: A Convex Analysis Perspective. In *Proceedings of International Conference on Machine Learning (ICML)*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, page 127063.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient Transformers: A Survey. *ACM Computing Surveys*, pages 1–28.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Roziere, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and Efficient Foundation Language Models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open Foundation and Fine-Tuned Chat Models.

Szymon Tworkowski, Konrad Staniszewski, Mikoł aj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Mił oś. 2023. Focused Transformer: Contrastive Training for Context Scaling. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

William A. Falcon and The PyTorch Lightning team. 2019. Pytorch Lightning.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt Compression and Contrastive Conditioning for Controllability and Toxicity Reduction in Language Models. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick Von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. 2020. Differentiable Top-k Operator with Optimal Transport. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. 2023. Effective Long-Context Scaling of Foundation Models.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023a. GLM-130B: An Open Bilingual Pre-trained Model. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Zhanpeng Zeng, Cole Hawkins, Mingyi Hong, Aston Zhang, Nikolaos Pappas, Vikas Singh, and Shuai

Zheng. 2023b. VCC: Scaling Transformers to 128K Tokens or More by Prioritizing Important Tokens. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*.

Shunyu Zhang, Yaobo Liang, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Multi-View Document Representation Learning for Open-Domain Dense Retrieval. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shen Zheng, Jie Huang, and Kevin Chen-Chuan Chang. 2023. Why Does ChatGPT Fall Short in Providing Truthful Answers? In *Proceedings of ICBINB Workshop*.

A Optimal nuggets selection

The nuggets selection module, i.e. Scorer, is learned through the residual connection introduced in Section 2.4. With gradient signal from the self-attention, Scorer tends to select the tokens that are mostly attended by the decoder (parameterized by θ). However, it remains a question whether the selection is optimal. Here we provide an empirical estimate of the gap between the optimal nuggets selection and Scorer.

Suppose we select k nuggets out of n tokens, we define a selection as a set of indices

$$\mathcal{I} = \{i_1, i_2, \ldots, i_k\}, \quad 1 \leq i_j \leq n.$$

From the definition, we can see that

$$\mathcal{I} \subseteq \{1, 2, 3, \ldots, n\}.$$

We further define the optimal selection $\mathcal{I}^*$ as the selection that achieves *the best performance* on a downstream task, e.g. lowest perplexity for language modeling. We denote the selection of Scorer as $\bar{\mathcal{I}}$. We want to answer two questions: How similar are $\mathcal{I}^*$ and $\bar{\mathcal{I}}$, and what is the performance gap between $\mathcal{I}^*$ and $\bar{\mathcal{I}}$?

Finding $\mathcal{I}^*$ is a non-trivial combinatorial optimization problem. The only possible solution, as we know, is to enumerate $\binom{n}{k}$ different selections, which is infeasible for large n and k. Therefore, we approximate $\mathcal{I}^*$ with a greedy algorithm. The basic idea is to start with $\mathcal{I} \leftarrow \bar{\mathcal{I}}$. Iteratively, for each index $i \in \mathcal{I}$, we replace it with an optimal index from the un-chosen indices so that it achieves the best downstream performance. We formalize it in Algorithm 1 with an example downstream task of language modeling.

We conduct experiments with the checkpoints in Section 5. We compress a sequence of up to 128 tokens into nuggets with a compression ratio of 10x. We present the model with another 64 tokens without compression. The model is required to predict the next 64 tokens, and we measure the subword-level perplexity of DODO. Because Algorithm 1 contains 2 for loops and is expensive to execute, we only sample 1000 documents from the test set of WikiText-103 (Merity et al., 2017).

To measure the difference between $\bar{\mathcal{I}}$ and $\mathcal{I}^*$, we count how many elements are replaced in $\bar{\mathcal{I}}$ with Algorithm 1. On average, 24.7% nuggets tokens are replaced, meaning Scorer is roughly 75.3% "correct". After replacing $\bar{\mathcal{I}}$ with $\mathcal{I}^*$, the overall subword-level perplexity is improved from 7.74 to 7.13, or $\mathcal{I}^*$ is roughly 7.9% better than $\bar{\mathcal{I}}$ in terms of downstream task performance.

Algorithm 1 A greedy algorithm to find the "optimal" selection $\mathcal{I}^*$.

```
Input: k (number of nuggets) and n (number of tokens) (0 < k ≤ n), encoder outputs x_{1:n}
Output: A selection I and the corresponding LM perplexity b
  Initialize I = {i_1, i_2, ..., i_k} with Scorer.
  Perplexity b ← Decoder(x_{1:n}, I)        ▷ Lowest perplexity so far
  for i ∈ I do
      for i' ∈ {1, 2, ..., n}\I do        ▷ All possible replacements from unchosen indices
          I' ← (I\{i}) ∪ {i'}        ▷ Replace i in I with i'
          Perplexity b' ← Decoder(x_{1:n}, I')
          if b' < b then        ▷ If i' is better than i, make the replacement permanent
              b ← b', I ← I'
          end if
      end for
  end for
```

In conclusion, we conduct experiments to show that Scorer is adequate to select nuggets as it can achieve similar performance as a decoder-aware optimal selector.

B Implementation & training details

B.1 Implementation

The training pipeline of DODO is implemented with the PyTorch (Paszke et al., 2019) and Pytorch Lightning package (William A. Falcon and The PyTorch Lightning team, 2019). We use the ZeRO stage-2 provided by the DeepSpeed Rasley et al. (2020) package with mixed precision to accelerate the training. The implementation of DODO is based on the huggingface/transformers package (Wolf et al., 2020). We used the implementation of huggingface/peft (Sourab Mangrulkar et al., 2022) for LoRA (Hu et al., 2022). Our dataset reader uses huggingface/datasets (Lhoest et al., 2021).

A soft prompt is involved for the training objectives of autoencoding. The soft prompt is treated as part of the model parameters and one soft token to be trained. We empirically found that the number of soft prompt tokens does not have a major impact on model performance.

B.2 Hyperparameters and training devices

For all the experiments, we follow the training setup of Touvron et al. (2023b) and use an Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-5}$. We use a cosine learning rate scheduler (Loshchilov and Hutter, 2017) with warmup of 2k steps, and the period of the cosine annealing function is set as 150k steps.

All the text generation processes in this paper are implemented as greedy decoding.

We train the models on 16 NVIDIA Tesla V100 GPUs (32 GiB), each with a batch size of 1. Gradients are accumulated for 2 batches before the execution of the optimizers. All the models are trained until early stopping because of the convergence of the loss on the validation set.

B.3 Number of parameters

In this section, we enumerate the number of parameters in DODO , as shown in Table 5. Except for the frozen LLAMAmodel, DODO has an encoder and decoder, which contains additional parameters to the Llama model with LoRA (Hu et al., 2022) (rank = 32), a scorer (2-layer feedforward neural networks), and a soft prompt that adds a special token to the embedding matrix.

For the experiments in Section 5, we use LoRA to train COMPRESSIVE , which contains a decoder and a soft prompt as we shown in Table 5. However, compared to the size of LLAMA, the trainable parameters of both DODO and COMPRESSIVE are significantly fewer (<1%).

C Example text for `nuggets` selection analysis

We sample a passage from Wikipedia and run `Scorer` on the text, where we set the compression ratio $r = 10$. The results are shown in Fig. 7.

D Prompts used in the paper

Here we list all the prompts used in Section 6.

D.1 Compress texts with LMs

The prompt used by the LMSUMM method to generate a summary for a given text is:

```
[INST]
Please summarize the following
text into $WORD words: $TEXT
[/INST]
```

We replace `$WORD` with $\lceil n \cdot r \rceil$, where n is the number of words (counted by spaces) and r is a desired ratio (in Section 6, r is 10).

D.2 Question answering on SQuAD

In the SQuAD experiment (Section 6.1), a prompt is used to answer a question given a document:

```
[INST]
$DOCUMENT
Based on the provided document,
answer the following question:
$QUESTION
[/INST]
```

We replace `$DOCUMENT` with the context document and `$QUESTION` with the question.

D.3 Summarization

In the summarization experiment (Section 6.2), we use the following prompt:

```
[INST]
$DOCUMENT
Please summarize the above
document in one sentence.
[/INST]
```

We replace `$DOCUMENT` with the document to be summarized.

E Normalization algorithm for SQuAD answers

The output of the language model tends to have tokens other than the answer or have different forms. For each pair of model output and SQuAD answer, we apply the following rules:

- Convert all English numbers to digits. E.g. convert "two" to "2".
- Replace all punctuation marks with spaces.
- Remove side spaces on both sides.
- Lowercase the string.

After these steps, a program is used to check if the model output contains the answer. We restrict the model to generate up to 64 tokens in case they generate many tokens to hit the answer. [9]

[9] They rarely do, as they are not optimized to cheat SQuAD.

module	#params	percentage	trainable
LLAMA-7B	6.74B	99.01%	no
encoder (w/ LoRA, ϕ)	25.2M	0.37%	yes
decoder (w/ LoRA, θ)	25.2M	0.37%	yes
`Scorer` (φ)	16.8M	0.25%	yes
soft prompt (θ)	4,096	<0.0001%	yes

Table 5: Parameter count of DODO. We do not distinguish `Llama-7b`, `Llama-2-7b`, and `Llama-2-7b-chat` here as they have the same architecture. The parameters of the encoder and decoder are counted as additional parameters with LoRA compared to the base model.

> The Brooklyn Nets have built themselves up from next to nothing. Devoid of anything close to an asset before 2015, the Nets had to make something out of nothing. They have done so indeed, loading the roster and asset cupboards simultaneously. Unfortunately, just as quickly as Marks acquired youngsters, he must also decide which ones should stick around. It's an arduous exercise, and even tougher for a team far from contention. Most teams reach this stage just as they are close to playoff-caliber. The Nets do not have this luxury, and must evaluate with a much longer view than the average young team. Put simply, they must think like a contender before becoming one. Luckily, the current roster has distinct tiers of young players in terms of their long-term potential. Eight of the nine under-25 players can be split into two tiers. Locks The group of definite keepers is relatively simple. These players have the most potential of the current Nets. Although D'Angelo Russell has gone through some rough patches, he has displayed enough promising signs to warrant the "keeper" status. His crafty ball-handling, scoring off the dribble, shooting off the catch, and great passing vision all make him an ideal fit for Kenny Atkinson's attack. Being the No. 2 overall selection in a draft is typically enough credibility to keep a player around, but Russell has shown legitimate flashes of star potential as well. Giving up on him now would be a fatal mistake. Jarrett Allen, a rookie center from the University of Texas, has done a wonderful job in his specialized role. With superb athleticism that allows him to protect the rim and switch onto perimeter attackers, Allen is quite capable of captaining a modern defense. This athleticism helps him on offense as well, as he gets plenty of lobs to finish pick-and-roll plays. When in doubt, the guards can chuck it up to him for an easy deuce. The vertical dimension of basketball is rarely appreciated .

Figure 7: Example texts processed by the `Scorer` of DODO. Darker texts have a higher score than light texts. The tokens in green background are selected as `nuggets`.