Walk and Read Less: Improving the Efficiency of Vision-and-Language Navigation via Tuning-Free Multimodal Token Pruning

Wenda Qin, Andrea Burns, Bryan A. Plummer, Margrit Betke
Boston University
Correspondence: wdqin@bu.edu

Abstract

Large models achieve strong performance on Vision-and-Language Navigation (VLN) tasks, but are costly to run in resource-limited environments. Token pruning offers appealing tradeoffs for efficiency with minimal performance loss by reducing model input size, but prior work overlooks VLN-specific challenges. For example, information loss from pruning can effectively increase computational cost due to longer walks. Thus, the inability to identify uninformative tokens undermines the supposed efficiency gains from pruning. To address this, we propose Navigation-Aware Pruning (NAP), which uses navigation-specific traits to simplify the pruning process by pre-filtering tokens into foreground and background. For example, image views are filtered based on whether the agent can navigate in that direction. We also extract navigation-relevant instructions using a Large Language Model. After filtering, we focus pruning on background tokens, minimizing information loss. To further help avoid increases in navigation length, we discourage backtracking by removing low-importance navigation nodes. Experiments on standard VLN benchmarks show NAP significantly outperforms prior work, preserving higher success rates while saving more than 50% FLOPS[1].

1 Introduction

Vision-and-Language Navigation (VLN) evaluates an AI agent's ability to navigate an environment following a natural language instruction (Fried et al., 2018; Tan et al., 2019; Hong et al., 2021; Chen et al., 2021, 2022; Wang et al., 2024c). However, sometimes the computational costs for high-performing models are too high for hardware-limited agents, underscoring the need to prioritize **navigation efficiency**. Token pruning improves computational efficiency by reducing input size, offering a trade-off between performance and cost. For example, as shown in Fig. 1, methods like FastV (Chen et al., 2025) can reduce the number of floating point operations per second (FLOPS) by 30% while only suffering from a 1.5% drop in navigation success rate (SR) on RxR-English (Ku et al., 2020). Additionally, tuning-free token pruning allows for the same model to flexibly adapt to different hardware constraints by adjusting pruning rates without costly retraining.

A drawback of existing token pruning strategies (Bolya et al., 2022; Zhang et al., 2024; Chen et al., 2025) is that they are designed for general Vision-and-Language Models (VLMs), ignoring the temporal dependence inherent in VLN tasks. This limits their ability to reducing navigation computation cost as pruned tokens may contain useful context to aids the agent's decisions. Without them, the agent becomes less certain and often backtracks to unvisited nodes for additional clues. This results

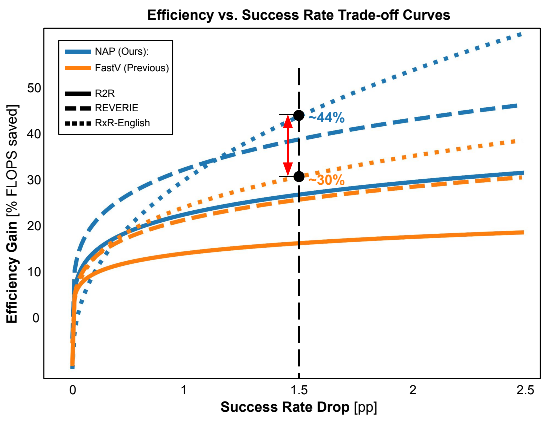

Figure 1: **Efficiency-vs-Success-Rate trade-off curves** for our method NAP (blue) and FastV (Chen et al., 2025) (orange) for 3 VLN datasets. Curves are fitted based on 15 token budget settings (100% to 30% of original budget). NAP consistently achieves greater efficiency for the same success rate loss compared to FastV. For example, for a 1.5% point (pp) drop in the success rate for navigation with RxR data, NAP achieves a 14% point gain in efficiency over FastV (red).

[1]Code available: `https://github.com/wdqin/VLN-NAP`

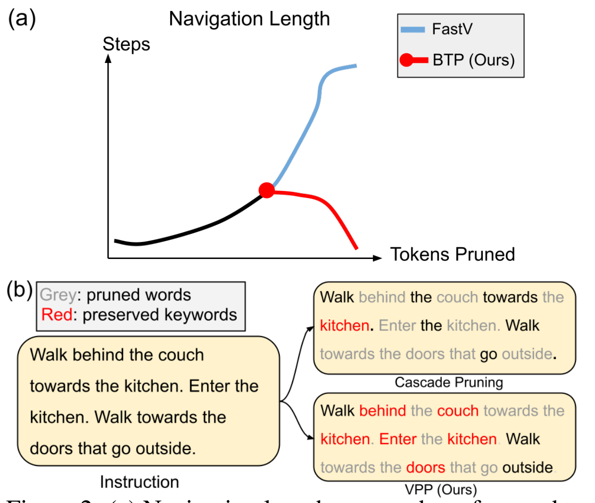
(a)
Navigation Length
Steps
FastV
BTP (Ours)
Tokens Pruned
(b)
Grey: pruned words
Red: preserved keywords
Walk behind the couch towards the kitchen. Enter the kitchen. Walk towards the doors that go outside.
Instruction
Walk behind the couch towards the kitchen. Enter the kitchen. Walk towards the doors that go outside.
Cascade Pruning
Walk behind the couch towards the kitchen. Enter the kitchen Walk towards the doors that go outside
VPP (Ours)

Figure 2: (a) Navigation length vs. number of pruned tokens. As token pruning strategies, BTP reduces navigation length while FastV increases it. (b) Example of instruction pruning. By prioritizing navigation-irrelevant words, VPP preserves more useful information in the instruction than Cascade pruning (Wang et al., 2021) with the same token budget.

in longer paths and additional computation (see blue curve in Fig. 2(a)), ultimately reducing the efficiency gains of pruning. In effect, the widely used attention-based textual pruning (Goyal et al., 2020; Wang et al., 2021) often fails to distinguish relevant from irrelevant instruction tokens. We find VLN models tend to assign high attention scores to punctuation and function words (e.g., "," and "the"), as shown in Fig. 2(b) and supported by prior work (Clark, 2019). As a result, important content tokens may be pruned instead, leaving instructions uninformative and impairing the agent's ability to navigate effectively.

To address these challenges, we propose **N**avigation-**A**ware **P**runing (NAP), a framework tailored for navigation tasks that enhances navigation efficiency by shortening the navigation duration ("walk less"), and prioritizing pruning navigation-irrelevant tokens ("read less"), achieving a significantly improved SR–FLOPS trade-off compared to prior methods (Fig. 1). NAP begins by separating views in the input panorama into action views (e.g., views corresponding to e, f, g in Fig. 3 (a)) and background views (o_1, o_2 in Fig. 3 (a)). We discovered that background views provide contextual information highly amenable to pruning, compared to action views. Our results demonstrate that selectively removing background tokens achieves substantial computational reduction with a minimal loss in SR. This process forms the basis of **B**ack**G**round **P**runing (BGP) in NAP.

We find that selectively pruning unvisited nodes (e.g., b, c in Fig. 3(c)) reduces path length (red in Fig. 2(a)), while preserving backtracking benefits when the number of retained nodes is properly tuned. To enable this, NAP introduces BackTracking Pruning (BTP), which removes unvisited nodes with low-importance scores from previous steps.

To enable the model to "read less," we introduce **V**ocabulary **P**riority **P**runing (VPP) to distinguish and prune uninformative instruction tokens as part of NAP. VPP constructs a *vocabulary of irrelevance* by prompting a large language model (e.g., LLaMA 3 (Dubey et al., 2024)) to identify terms that are non-essential for navigation. This allows VPP to prioritize pruning irrelevant tokens while preserving both LLM-identified and potentially important unseen words. As shown in Fig. 2 (b), important words like "couch," "enter," and "doors" are successfully retained by VPP, whereas the prior method (Wang et al., 2021) fails to do so.

To summarize, our contributions are as follows:

- We propose Navigation-Aware Pruning (NAP), which includes Background Pruning (BGP) to reduce visual inputs with minimal success rate loss, and BackTracking Pruning (BTP) to improve path efficiency by limiting backtracking.
- NAP introduces Vocabulary Priority Pruning (VPP), which leverages an LLM to identify and prune navigation-irrelevant instruction tokens, thus reducing input size.
- By combining BGP, BTP, and VPP, NAP achieves up to 2× speedups over the original model and 1.25× over the state-of-the-art pruning baseline with a smaller success rate drop (2-3 pp). These improvements are observed across multiple VLN models and datasets for both discrete and continuous environments.

2 Related Work

VLN Datasets and Models. Multiple datasets (Anderson et al., 2018; Ku et al., 2020; Qi et al., 2020; Zhu et al., 2021; Hong et al., 2022) have been curated to address VLN challenges that include long paths (Ku et al., 2020), object localization (Qi et al., 2020; Zhu et al., 2021), and continous environment (Krantz et al., 2020). To navigate through the environment, agents rely on transformer-based architectures for action prediction (Chen et al., 2021, 2022; An et al., 2023; Zhao et al., 2022; Huo et al., 2023; Wang et al., 2023a; Li and Bansal, 2023b; Wang et al., 2023b; Qiao et al., 2023a; Li and

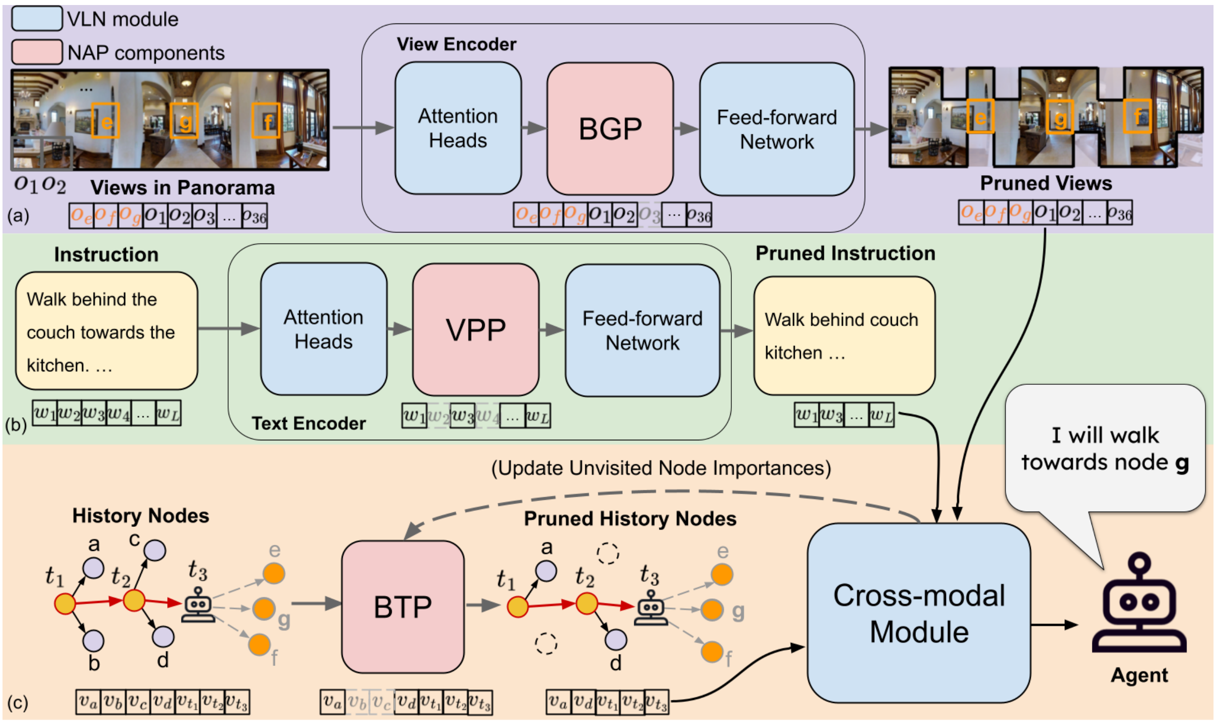

Figure 3: An overview of NAP components (BGP, VPP, BTP) on a VLN model. (a) BGP reduces the size of visual inputs by pruning the non-critical background views from $\{o_1, o_2, ..., o_{36}\}$ (details in Sec. 3.1). (b) VPP prunes the irrelevant word tokens from the instruction tokens $\{w_1, w_2, ..., w_L\}$ to shorten textual inputs (described in Sec. 3.3), and (c) BTP prunes the unvisited history nodes from $\{v_a, v_b, ..., v_h\}$ on the navigation map constructed by the VLN models to discourage backtracking (discussed in Sec. 3.2).

Bansal, 2023a; Wang et al., 2024b).

Advancements include history modules for navigation memory (Chen et al., 2021), backtracking mechanisms with topological mapping (Chen et al., 2022), enabling BTP as an efficiency-performance trade-off, and the use of synthesized visual data (Li and Bansal, 2023b). Wang et al. (2024b) further improved navigation accuracy with a debiasing strategy. Regarding VLN efficiency, Qiao et al. (2023b) explored Parameter-Efficient Transfer Learning, while Zhu et al. (2024) and Wang et al. (2024c) applied knowledge distillation to train smaller VLN models. These methods require new modules or model reconstruction, while our approach does not.

Token Reduction (Pruning and Merging). Recent works (Goyal et al., 2020; Wang et al., 2021; Kim et al., 2022; Liang et al., 2022; Bolya et al., 2022; Wei et al., 2023; Wang et al., 2024a; Zhang et al., 2024; Chen et al., 2025) primarily rely on attention scores to reduce sequence lengths in transformers. PoWER-BERT (Goyal et al., 2020) prunes input tokens based on self-attention. SpAtten (Wang et al., 2021) formally introduced cascade token and head pruning. Liang et al. (2022) addressed information loss by fusing pruned tokens, while Bolya et al. (2022) focused on removing feature redundancy. Chen et al. (2025) prunes all unimportant tokens at a specific layer of the VLM. SparseVLM (Zhang et al., 2024) leverages external LLMs to assess visual token importance. Previous strategies assume attention scores reflect token importance, failing under high text pruning rates. They also ignore the impact of view token pruning on efficiency. We are the first to propose VLN-specific pruning to address these challenges.

3 The NAP Method: BGP, BTP, and VPP

In VLN, an agent navigates based on an instruction of words $I = \{w_1, \ldots, w_L\}$. At each step, it processes views from a panorama $P = \{o_1, \ldots, o_N\}$ (Fried et al., 2018) and selects navigable locations $a \in A$ until it predicts a "STOP" action a_{stop}. Using the Matterport 3D simulator (Anderson et al., 2018; Ku et al., 2020; Qi et al., 2020), navigation is successful if the agent stops within 3 meters of the target; an agent in REVERIE (Qi et al., 2020) must also locate the target object.

Most VLN models split the views in panoramas $P = \{o_1, ..., o_N\}$ into action views $O_{act} = \{o_1, ..., o_n\}$ and background views $O_{bgr} = \{o_{n+1}, ..., o_N\}$ providing visual context. The action views $\{o_1, ..., o_n\}$ correspond to navigable nodes $\{a_1, ..., a_n\}$ indicating navigable locations at the current step. In addition to views, DUET-based (Chen et al., 2022) models (Li and Bansal, 2023b; Wang et al., 2024b,c; Zhu et al., 2024) adopt a topological map as "history" to track the unvisited nodes $V_{unvisited} = \{v_1, v_2, ..., v_m\}$ and visited nodes $V_{visited} = \{v_{m+1}, v_{m+2}, ..., v_M\}$, with $V_{unvisited}$ enabling backtracking actions $\{a_1^{bt}, ..., a_m^{bt}\}$. The model's inputs are thus tri-modal, including I, P, and V. At each step, the model selects an action from stop, navigate, or backtrack using the policy $a = \pi(I, P, V)$, where $a \in \{a_{\text{stop}}, a_1, \ldots, a_n\} \cup \{a_1^{bt}, \ldots, a_m^{bt}\}$.

Given an instruction $I = \{w_1, w_2, ..., w_L\}$, views $P = \{o_1, ..., o_N\}$, and history nodes $V = \{v_1, v_2, ..., v_M\}$, token pruning reduces sequence lengths with a certain process, denoted as function f. We obtain the pruned sequences $I' = f_I(w_1, ..., w_L)$, $P' = f_P(o_1, ..., o_N)$, and $V' = f_V(v_1, ..., w_M)$, such that $|I'| < |I|$, $|P'| < |P|$, $|V'| < |V|$. The VLN model then selects an action $a = \pi(I', P', V')$ with lower computational cost.

Efficient token pruning hinges on accurately assessing token importance. Our NAP framework for this problem includes three components: BackGround Pruning (BGP) for views (Sec. 3.1), BackTracking Pruning (BTP) of unvisited nodes (Sec. 3.2), and Vocabulary Priority Pruning (VPP) for instruction tokens (Sec. 3.3), which are integrated into NAP within VLN models (e.g., GOAT (Wang et al., 2024b)) as shown in Fig. 3.

3.1 Background Pruning (BGP)

In this section we discuss how we boost efficiency by pruning visual tokens. During the navigation process, panoramic views P are projected into a feature space using a pre-trained feature extractor, such as CLIP-B/16 (Radford et al., 2021), and enhanced with learned positional encodings representing the viewing direction. This results in a sequence of N view tokens, each associated with a feature x. For simplicity, we denote the encoded view token sequence as P and each view feature token as o. Our token importance scores come from the transformer blocks processing P. Specifically, each transformer block consists of a self-attention module with attention heads, followed by a feed-forward network. At each head h in layer i, self-attention scores are computed to quantify the dependencies between tokens:

$$\text{Attn}_{h_i} = \text{softmax}[Q_i \times K_i^T], \tag{1}$$

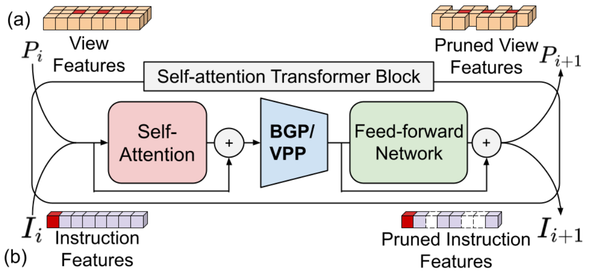

Figure 4: BGP (a) and VPP (b) processes. Both BGP and VPP prune view feature tokens based on their importance score after the self-attention module and the residual block at each transformer layer in a VLN model.

where $Q_i = P_i \times W_q^{h_i}$, $K_i = (P_i \times W_k^{h_i})$, and $W_q^{h_i}$, $W_k^{h_i}$ are trained weight matrices. The resulting Attn_{h_i} is a $N \times N$ matrix, where each entry $\text{Attn}_{h_i}[o_m, o_n]$ represents the normalized dependency of view o_n on view o_m. A higher value of $\text{Attn}_{h_i}[o_m, o_n]$ usually indicates that view o_n has a stronger influence on constructing the updated representation of o_m in the output P_{i+1}.

To determine the overall importance of each token, we aggregate the attention scores by summing each column of Attn_{h_i} across all attention heads H_i at layer i. This gives a vector of scores indicating the influence of each token o' on all tokens:

$$\text{Score}(o') = \sum_{h_i \in H_i} \sum_{o \in P} \text{Attn}_{h_i}[o, o']. \tag{2}$$

We interpret $\text{Score}(o')$ as the **importance score** of view o', which guides the pruning process by identifying less influential background views. Similar scores can be obtained for history nodes v (described in Sec. 3.2) and words w (discussed in Sec. 3.3) from the cross-modal module and language module, respectively. The module concludes by computing the latent feature $Z_i = \text{concatenate}(A_{h_i} \times (P_i \times W_v^{h_i}))$, where the concatenation is performed across all heads and $W_v^{h_i}$ is the weight matrix for each head.

BGP reduces the size of O_{bgr} in P by removing k_{BGP} tokens at each layer of the vision transformer encoder (see Fig. 4). Following Bolya et al. (2022), BGP is applied after computing the attention matrix Attn_{H_i} for P and before passing Z_i to the residual connection and feed-forward network. BGP retains all action view tokens O_{act} while pruning the k_{BGP} tokens with the lowest $\text{Score}(o)$ from O_{bgr} (i.e., removing their o and z values from O_{bgr} and Z_i). Action views are given, and relate to the views where the agent can perform an action. For example, if the agent can move to the right, then the

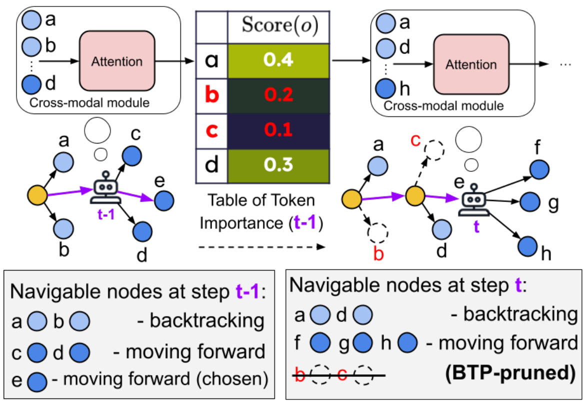

Figure 5: The BTP process. By keeping the number of backtracking nodes in the history less or equal to the threshold $k_{BTP} = 2$, (nodes a, d), and removing the rest (nodes b, c), the agent has fewer backtrack choices, thus is more likely to move forward (f, g, h) or stop.

view in that direction is an action view. The pruned P_i and Z_i are then processed to yield P_{i+1}. Consequently, the final output P_{L+1} contains $k_{\text{BGP}} \cdot L$ fewer tokens than the original P, resulting in a smaller visual input for action prediction.

This procedure can be thought of as separating tokens into foreground (action) and background (context), and then only pruning background tokens. In effect, we leverage *a priori* knowledge to simplify our token selection problem. This helps minimize the possibility that we remove key tokens that would result in longer navigation paths, which is further reduced by BTP in the next section.

3.2 Backtracking Pruning (BTP)

Backtracking in DUET-based VLN models (Chen et al., 2022) refers to returning to unvisited nodes from previous navigation steps. However, as discussed earlier, pruning tokens as done in Sec. 3.1 and Sec. 3.3) can increase navigation length. To address this, we propose BTP, which removes a subset of inconsequential unvisited nodes $V_{unvisited}$ from the history input V (nodes b & c in Fig. 5) to discourage backtracking. Specifically, $V_{unvisited} = \{v_1, v_2, ..., v_m\}$ contains action view tokens from earlier steps that were not selected, i.e., $\{v_1, v_2, ..., v_m\} = \{o_1, o_2, ..., o_m\}$, where $\{o_1, o_2, ..., o_m\} \in O^{t'}_{act}$ at previous steps $t' < t$. To determine the most crucial nodes for successful navigation, we track the importance scores of these tokens from the attention heads of the last cross-modal transformer block at each step (e.g., 0.4 for node a in Fig. 5). At the beginning of the next step, BTP retains the top k_{BTP} unvisited nodes based on their latest Score(o) and discards the remainder, so that $V_{unvisited} = \{v_1, ..., v_{k_{\text{BTP}}}\}$.

BTP offers two main benefits: It reduces the size of the history input V to the cross-modal module, lowering computational cost, and it limits its backtracking options from $\{a'_1, \ldots, a'_m\}$ to $\{a'_1, \ldots, a'_{k_{\text{BTP}}}\}$, enhancing path efficiency by reducing potential unnecessary backtracking.

3.3 Vocabulary Priority Pruning (VPP)

VPP operates similarly to BGP from Sec. 3.1 by pruning instruction tokens $w \in I$ based on the importance Score(w) derived from the attention mechanism of the language transformer block (see Fig. 4). However, we found that the attention scores are a noisy measure of importance. For example, as shown in Table 1, non-informative tokens, such as punctuation, often receive a high Score(w) and are retained, thereby wasting the token budget. Thus, as we did for BGP, we simplify the token selection problem via filtering using task-specific knowledge. However, rather than using action constraints, we prompt an LLM to construct a "vocabulary of irrelevance" $\mathcal{V}$ that identifies unimportant words based on its general knowledge rather than attention scores (illustrated in Fig. 6). We tokenize the training data to form a lexicon and prompt Llama 3 (Dubey et al., 2024) to label each word as "relevant" or "irrelevant" for navigation based on its association with direction/heading, environment description, or indoor/outdoor objects. Words labeled "irrelevant" were compiled into $\mathcal{V}$; see Appendix F for an example prompt and vocabulary.

At each layer of the language transformer encoder, VPP prunes k_{vpp} instruction tokens via a two-step process. First, it filters out all words present in $\mathcal{V}$. If the number of filtered tokens is less than k_{vpp}, the remaining tokens are pruned based on their attention scores Score(w) until the requirement is met. Otherwise, if more tokens are filtered than necessary, some are reinstated according to Score(w) to preserve additional information. The VPP process is detailed in Algorithm. 1.

VPP more effectively retains essential words in instructions, enabling the agent to function normally even at higher pruning rates, than attention-based strategies (Table 1). Importantly, the vocabulary is constructed **prior** to navigation (**offline**) and, thus, **no** LLM computation overhead is introduced during the actual navigation process.

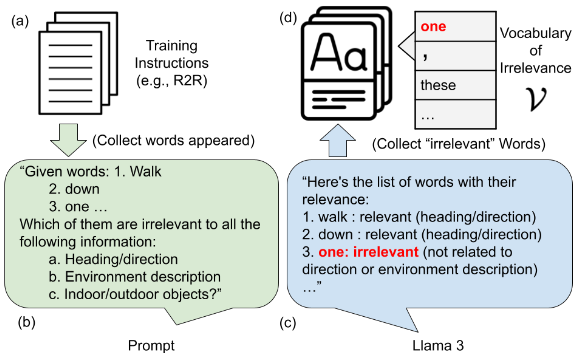
(a)
Training Instructions (e.g., R2R)
(Collect words appeared)
"Given words: 1. Walk
2. down
3. one ...
Which of them are irrelevant to all the following information:
a. Heading/direction
b. Environment description
c. Indoor/outdoor objects?"
(b)
Prompt
(d)
Aa
one
,
these
...
Vocabulary of Irrelevance
$\mathcal{V}$
(Collect "irrelevant" Words)
"Here's the list of words with their relevance:
1. walk : relevant (heading/direction)
2. down : relevant (heading/direction)
3. one: irrelevant (not related to direction or environment description)
..."
(c)
Llama 3

Figure 6: The construction of the "vocabulary of irrelevance" $\mathcal{V}$, as a 4-step process, (a) collecting lexicon of words appeared in the training instructions, (b) prompting a LLM with words in the lexicon, (c) Classifying relevant and irrelevant words by LLM, and (d) grouping words irrelevant into the vocabulary. The resulted vocabulary helps identify words likely to waste computation, and thus should be pruned.

Retain in %	Method	Instruction Tokens (grey ones are pruned)
100	-	<s>Exit the room . Turn right . Start down the stairs and stop 3 steps down . </s>
50	VPP	<s>Exit the room . Turn right . Start down the stairs and stop 3 steps down . </s>
50	Att. Scores	<s>Exit the room . Turn right . Start down the stairs and stop 3 steps down . </s>
25	VPP	<s>Exit the room . Turn right . Start down the stairs and stop 3 steps down . </s>
25	Att. Scores	<s>Exit the room . Turn right . Start down the stairs and stop 3 steps down . </s>

Table 1: Instruction tokens retained under different pruning rates. VPP is more effective in preserving key information than attention-score pruning (Cascade).

Algorithm 1 Vocabulary Priority Pruning

```
1: procedure VPP($I, \mathcal{V}$, Scores, $k_{\text{vpp}}$)
2:     $I_r \leftarrow$ empty sequence                ▷ retained tokens
3:     $I_p \leftarrow$ empty sequence                ▷ pruned tokens
4:     Sort $w$'s in $I$ by Score($w$) from high to low
5:     for $w$ in $I$ do
6:         if $w$ not in $\mathcal{V}$ then
7:             $I_r$.add($w$)
8:         else
9:             $I_p$.add($w$)
10:    if $k_{\text{vpp}} >$ length($I$) - length($I_r$) then
11:        for $i = 1$ to $k_{\text{vpp}}$ do
12:            $I_r$.add($w_i$)
13:    else
14:        for $w$ in $I_p$ do
15:            if length($I$) - length($I_r$) = $k_{\text{vpp}}$ then
16:                break
17:            $I_r$.add($w$)
18:    return $I_r$
```

4 Experiments

Datasets. We evaluated NAP on R2R (Anderson et al., 2018), RxR-English (Ku et al., 2020), and REVERIE (Qi et al., 2020). For each dataset, token pruning was tested on two splits, the validation "seen" split and validation "unseen" split.

Evaluation Metrics. Our primary *efficacy* metric is navigation Success Rate (SR), which measures the agent's ability to navigate correctly. Additional efficacy metrics are reported in our result tables. For *efficiency*, we use a simplified version of FLOPS-based formula from Wang et al. (2024c):

$$G_{\text{total}} = G_{lan}(I) + D \cdot (G_{vis}(P) + G_{cm}(I, P, V)) + c,$$

where D is the number of decision steps, G_{lan} the Giga-FLOPS (GFLOPS) of the language module getting instruction features, G_{vis} the GFLOPS of the visual module processing view features, G_{cm} the GFLOPS of the cross-modal module predicting actions from all input features, and c the cost from other sources, if any. To convey efficiency improvements, we report the ratio of GFLOPS after pruning to GFLOPS before pruning [in %]:

$$\text{FLOPS\%} = G_{\text{pruned}} / G_{\text{original}}. \qquad (3)$$

We compute FLOPS using the Python toolkit thop.

Tested VLN Models. We evaluated our token pruning strategies on three VLN models: HAMT (Chen et al., 2021), DUET (Chen et al., 2022), and GOAT (Wang et al., 2024b), using the pre-tuned parameters provided by the respective authors. We extensively tested NAP with VLN-GOAT (Table 2) due to its outstanding performance across datasets. VLN-GOAT comprises six transformer layers in the language module, two in the view module, and three in the cross-modal module.

Tested Pruning Strategies. We are the first to apply *multimodal* token pruning to VLN. As baselines, we evaluate general strategies applicable to VLN inputs: random pruning, cascade pruning (Goyal et al., 2020; Wang et al., 2021), and FastV (Chen et al., 2025). We also include Token Merging (ToMe) (Bolya et al., 2022) for view pruning. Unless specified, we only prune background views for visual input, due to significant SR drops (see "Visual Input Pruning").

4.1 Results

Table 2 reports that NAP consistently achieves greater FLOPS reductions than baseline methods for VLN-GOAT. For example, at the 50% pruning setting, NAP lowers FLOPS compared to prior work by by 7 (R2R), 2.2 (RxR-English), and 3.1% (REVERIE) points. At the same time, NAP also yields up to 2.5% better navigation success rates.

Method	R2R				RxR-English				REVERIE			
	Seen$\uparrow$	Unseen$\uparrow$	Steps$\downarrow$	FLOPS%$\downarrow$	Seen$\uparrow$	Unseen$\uparrow$	Steps$\downarrow$	FLOPS%$\downarrow$	Seen$\uparrow$	Unseen$\uparrow$	Steps$\downarrow$	FLOPS%$\downarrow$
Upper Bound, 100% Tokens												
VLN-GOAT	84.8	78.1	7.3	100	75.2	69.6	8.1	100	80.7	53.8	10.1	100
Retain 70% $\pm$ 2% Tokens												
Random	80.4	74.1	6.8	76.1	69.2	64.5	8.4	68.6	72.2	49.0	11.1	82.1
FastV[1]	81.7	74.9	6.9	78.3	73.2	68.1	8.4	70.7	78.7	51.1	10.3	76.9
Cascade Pruning[2]	81.2	75.3	6.9	76.7	73.3	68.5	8.4	68.6	77.0	52.1	10.5	76.8
Cascade + ToMe[3]	**82.1**	**75.6**	6.7	74.4	73.3	68.4	8.3	69.7	77.5	51.9	10.4	76.4
NAP (Ours)	81.8	74.5	**6.5**	**68.3**	**73.5**	68.4	8.4	**65.3**	**78.9**	**53.1**	**10.2**	**71.0**
Retain 60% $\pm$ 2% Tokens												
Random	74.2	68.2	8.2	70.4	65.5	59.9	8.5	59.7	71.7	47.3	10.8	66.6
FastV[1]	78.7	68.8	8.0	71.6	70.9	65.8	8.5	61.8	77.0	51.0	10.3	65.8
Cascade Pruning[2]	78.0	72.1	7.7	65.8	72.1	66.9	8.6	59.7	77.2	50.7	10.3	63.6
Cascade + ToMe[3]	78.2	71.6	7.4	66.9	72.2	66.8	8.5	61.0	76.7	51.0	10.4	65.6
NAP (Ours)	**82.6**	**75.2**	**7.0**	**60.0**	**73.3**	**68.0**	8.5	**56.2**	**79.1**	**52.2**	**10.1**	**58.9**
Retain 50% $\pm$ 2% Tokens												
Random	69.3	65.4	10.0	71.6	59.3	54.7	**8.6**	51.5	71.1	44.9	10.8	53.0
FastV[1]	74.3	67.0	9.8	75.6	64.8	60.4	8.7	54.3	75.6	47.9	10.6	55.9
Cascade Pruning[2]	74.1	70.7	9.8	70.0	68.3	63.6	8.8	52.9	75.2	46.3	10.5	51.4
Cascade + ToMe[3]	77.4	71.1	7.6	62.6	68.3	63.7	8.7	53.8	63.3	42.8	15.2	75.7
NAP (Ours)	**81.4**	**73.8**	**7.0**	**55.6**	**70.1**	**63.9**	8.9	**49.3**	**76.3**	**49.6**	**10.1**	**48.3**

Table 2: Performance of VLN-GOAT under various token pruning strategies. Metrics include navigation success rate ("Seen","Unseen"), average steps from both splits, and average FLOPS% relative to the base model. FastV[1](Chen et al., 2025) (textual & visual), Cascade pruning[2] (textual & visual) (Goyal et al., 2020; Wang et al., 2021), Cascade pruning and Token Merging (Bolya et al., 2022) (Cascade + ToMe[3], visual) are pruning techniques that are compared to NAP. NAP consistently yields better FLOPS reduction while maintaining comparable or higher SRs.

BGP	BTP	VPP	SR$\uparrow$	SPL$\uparrow$	Steps$\downarrow$	FLOPS%$\downarrow$
✗	✗	✗	69.6	61.8	8.2	100.0
✓	✗	✗	65.5	55.0	9.1	76.6
✗	✓	✗	68.2	60.6	8.2	96.0
✗	✗	✓	68.2	60.0	8.4	81.9
✓	✓	✗	64.9	55.3	8.9	69.2
✓	✗	✓	65.4	54.3	9.3	56.5
✗	✓	✓	66.6	58.8	8.4	77.5
✓	✓	✓	63.9	54.3	9.0	49.3

Table 3: Ablation of BGP, BTP, and VPP on RxR-English Unseen. Each component lowers navigation cost while maintaining most SR and SPL.

These advantages are retained even with larger token budget. For instance, in R2R at a 60% pruning rate, NAP achieves nearly the same SR as Cascade + ToMe (a 0.4 pp difference) while reducing FLOPS by an additional 14 pp (from 74.4% to 60.0%). This improvement aligns with the trade-off curves shown in Fig. 1.

In all R2R and REVERIE settings, NAP completes navigation with the fewest steps. Notably, under a 50% token budget in R2R, NAP reduces the average number of steps from 7.3 to 7.0, while other strategies increase it.

In Appendix B we also validated our method on HAMT (Chen et al., 2021) and DUET (Chen et al., 2022) using the R2R and REVERIE datasets, compared to FastV (Chen et al., 2025). In DUET, our method outperforms FastV by 5pp to 10pp in SR, SPL, RGS, and RGSPL with similar FLOPS. In HAMT, where BTP cannot be applied, our approach still shows superior SR and SPL on R2R and outperforms FastV in both efficacy and efficiency on REVERIE.

4.2 Model Analysis

Table 3 assesses the effectiveness of BGP, BTP, and VPP under a 50% token budget. We find BGP achieves the largest FLOPS reduction (23.4pp), highlighting the high prunability of background views. BTP contributes an additional 7 pp FLOPS reduction and shortens navigation by 0.3 steps on average (more than 1 on R2R), while VPP reduces FLOPS by 20pp. We find that FLOPS improvement in VPP is closely related to the instruction length (see Appendix D).

Visual Input Pruning (BGP and BTP). Table 4 compares SR–FLOPS trade-off between BGP and full view pruning with with FastV (Chen et al., 2025) across 5 pruning rates. Pruning action views severely degrades efficacy (0% SR at high rates) and can increase FLOPS (108% at 20%). This highlights the importance of preserving action views, a core aspect of BGP. However, pruning background views increases decision steps (Fig. 7(a), from 7.0 to 10.5). BTP reverses this trend by keeping

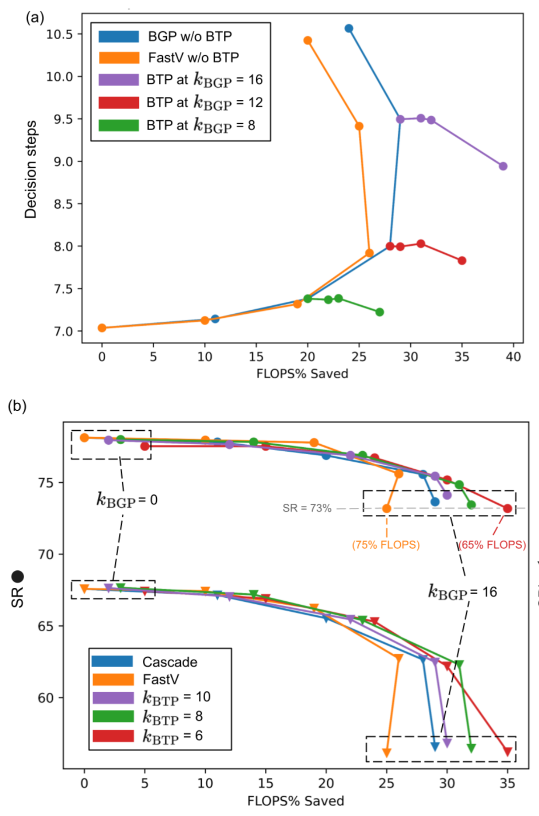

Figure 7: (a) Averaged decision steps for navigation applying BGP, FastV, and BTP. (b) SR and FLOPS saved of different pruning rates for Cascade pruning, FastV, and BGP with various BTP settings in R2R. BTP saves additional FLOPS by shortening the navigation length while keeping SR almost unaffected.

just 6–10 unvisited nodes per step. As shown in Fig. 7(b), doing so saves 10 pp more FLOPS than FastV (pruning backgrounds), with minimal SR loss. Appendix C reports that BTP also improves performance on our other datasets.

Textual Input (VPP). VPP prunes the textual tokens in the instruction I. We compare VPP with Cascade Pruning and FastV for instruction token removal on different datasets (Fig. 8). VPP (green) consistently outperforms the other two strategies across most pruning rates, with improvements >10 pp in the 40–60% range.

Table 5 evaluates whether a vocabulary constructed from one VLN dataset can be reused on other datasets. We can see from the result that cross-dataset vocabularies still effectively mitigate performance loss. For example, a vocabulary built from RxR achieves 75.5% SR under 50% pruning, close to the 76.0% SR that is from the R2R dataset. This suggests that navigation-relevant words are largely shared across datasets.

Continuous Environment. While our navigation

Pruning	Full Views (FastV)			BGP		
Rate%	SR↑	Steps↓	FLOPS%↓	SR↑	Steps↓	FLOPS%↓
0	78	7	100	78	7	100
20	51	11	108	77	7	80
40	3	2	20	74	9	71
60	0	0	10	69	11	76
80	0	0	9	69	11	73

Table 4: Comparison of BGP and pruning full views including action views across different pruning rates on the R2R dataset. Pruning action views leads to substantial drops in SR or longer, less efficient navigation paths, indicating the importance of preserving action views.

Source / Target	R2R	RxR	REVERIE
R2R (68.8)	+7.2 (76.0)	+6.7 (75.5)	+5.2 (74.0)
RxR (66.4)	+1.6 (68.0)	+1.8 (68.2)	+1.8 (68.2)
REVERIE (36.5)	+13.3 (49.8)	+11.4 (47.9)	+13.5 (50.0)

Table 5: Navigation success rates (in parentheses) using vocabularies of irrelevance built from source datasets (columns) and applied to target datasets (rows). The leftmost column (e.g., 68.8) shows success rates under a 50% instruction pruning rate with Cascade Pruning as a reference. Performance remains consistent across vocabularies from different sources, suggesting that irrelevant vocabulary is largely dataset-independent.

so far is based on a node-to-node discrete setting, NAP also applies to continuous environments through a waypoint predictor. The waypoint predictor is a commonly adopted module (Krantz et al., 2021; Hong et al., 2022; An et al., 2024) that converts low-level actions (e.g., “go left”) into high-level node traversals (e.g., “go to viewpoint A”), enabling discrete VLN models to operate in continuous settings. NAP can therefore leverage the outputs of the waypoint predictor for input pruning in continuous environments.

To validate this, Table 6 compares NAP and Fast-V on VLN-CE-R2R (Krantz et al., 2020) under a 50% visual+text token budget, integrating them into the ETPNav (An et al., 2024) model. Given that the waypoint predictor introduces an additional computational overhead of approximately 10 pp FLOPS, NAP reduces 37.6 pp FLOPS with an SR drop of 4.2 pp, which is 14.1 pp higher than Fast-V. Meanwhile, we observe that NAP leads to roughly one additional step per navigation compared to Fast-V, which diminishes the FLOP savings from pruning backtracking nodes. We will leave the investigation of this phenomenon for further FLOPS improvement as future work.

Table 7 also provides an ablation study of NAP

Strategy	k_{BGP}	k_{BTP}	VPP	Steps↓	SR↑	FLOPS%↓
NAP	3	4	40%	9.6	52.7	62.4
FastV	3	-	40%	8.3	38.6	63.9

Table 6: A comparison between NAP and Fast-V on VLN-CE-R2R dataset (Krantz et al., 2020) with a total token budget of 50%. NAP preserves 14.1 pp more SR while achieving similar FLOPS deduction.

BGP	BTP	VPP	SR↑	Steps↓	FLOPS%↓
✗	✗	✗	56.9	8.8	100.0
✗	✓	✗	55.6	9.2	90.8
✓	✗	✗	56.2	8.8	97.4
✓	✓	✗	54.0	9.3	88.9
✗	✗	✓	55.6	9.2	75.9
✗	✓	✓	53.0	9.7	65.3
✓	✗	✓	56.1	9.2	73.8
✓	✓	✓	52.7	9.6	62.4

Table 7: Ablation study of NAP components on the VLN-CE dataset. The pruning parameters are the same as Table 6. The result shows that all components reduce FLOPs, with BTP and VPP proving substantially more effective than BGP (9.2 pp and 24.1 pp vs. 2.6 pp, rows 2, 5, and 3).

components in the continuous setting. VPP is the most effective strategy in this case: 24.1 pp FLOPS reduced with only a 1.3 pp SR drop. Compared to the discrete setting, FLOP reductions shift: BGP contributes far less (3 pp), while BTP and VPP contribute substantially more (10 pp and 24 pp, respec). This difference arises because VLN models in the continuous setting use a smaller view encoder to enable faster image processing.

Illustrative Result. The impact of NAP is illustrated in a navigation example in Fig. 9. VPP preserves key instruction words, enabling the agent to complete the route. Meanwhile, BTP "forces" the agent to stop at the correct destination by removing potential backtracking nodes; without it, unpruned unvisited nodes would cause backtracking, resulting in longer paths and navigation failure.

5 Conclusions

Our work introduces NAP, a navigation-specific token pruning framework that integrates three strategies: BGP, BTP, and VPP. Compared to NAP, general pruning methods could cause longer navigation paths due to the view removal. Also, as methods designed for tasks other than navigation, they also fail to reliably preserve navigation-critical instruction tokens. NAP addresses these issues by tailoring pruning to the navigation task, reducing the input size of VLN models across all modalities: text, visual views, and history nodes. Specifically, BGP prunes irrelevant background views while preserving action views; BTP removes redundant backtracking nodes, shortening both history inputs and navigation paths; and VPP employs an LLM-guided vocabulary to distinguish and retain navigation-relevant tokens, enabling more aggressive yet precise pruning. Experiments on the R2R, RxR-English, and REVERIE datasets show that NAP achieves deeper token reduction while maintaining higher success rates in VLN tasks. Our model- and dataset-agnostic framework further reduces computational cost while improving navigation efficiency, enabling the deployment of VLN models in resource-constrained environments.

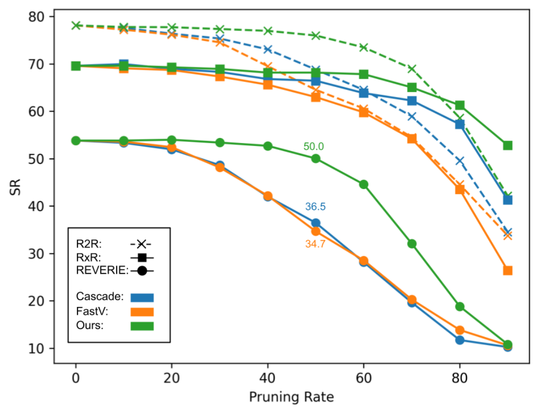

Figure 8: Success rates of VLN applying VPP, Cascade Pruning, and FastV on instruction input. VPP preserves noticeably more SR with the same pruning rate than Cascade Pruning and FastV.

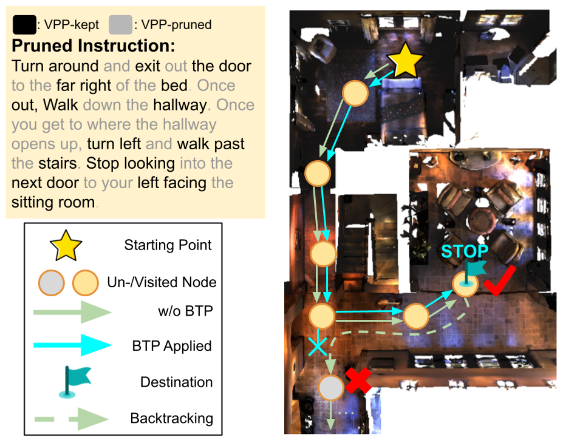
■: VPP-kept ■: VPP-pruned
Pruned Instruction:
Turn around and exit out the door to the far right of the bed. Once out, Walk down the hallway. Once you get to where the hallway opens up, turn left and walk past the stairs. Stop looking into the next door to your left facing the sitting room.

Figure 9: An example of navigation with NAP. BTP prevents a long and failed navigation path by pruning unvisited nodes, while VPP preserves most key information and removes tokens unhelpful to navigation.

6 Limitations

A majority of generic token pruning strategies, including our BGP and BTP, assume that attention scores accurately reflect the true importance of token features for navigation success. However, this assumption is inherently flawed. While VPP partly mitigates this issue for instruction pruning, an analogous solution for view pruning would incur expensive costs, such as those associated with gradient- or perturbation-based saliency. Moreover, as shown in Table 2, random token pruning performs nearly as well as attention-based methods—except for our approach. This suggests that many VLN tokens may contribute similarly to navigation, yielding comparable efficiency gains and success rate losses regardless of the pruning method. We hope our work serves as a starting point for understanding token importance in VLN for both text and visuals, and inspires further development of effective, cost-efficient indicators for VLN token pruning.

References

Dong An, Yuankai Qi, Yangguang Li, Yan Huang, Liang Wang, Tieniu Tan, and Jing Shao. 2023. BEVBERT: Multimodal Map Pre-training for Language-guided Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2737–2748.

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. 2024. ETP-Nav: Evolving Topological Planning for Vision-Language Navigation in Continuous Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language Navigation: Interpreting Visually-grounded Navigation Instructions in Real Environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token Merging: Your ViT But Faster. *arXiv preprint arXiv:2210.09461*.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2025. An Image is Worth 1/2 Tokens After Layer 2: Plug-and-play Inference Acceleration for Large Vision-language Models. In *European Conference on Computer Vision*, pages 19–35. Springer.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. 2021. History Aware Multimodal Transformer for Vision-and-language Navigation. *Advances in neural information processing systems*, 34:5834–5847.

Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. 2022. Think Global, Act Local: Dual-scale Graph Transformer for Vision-and-language Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16537–16547.

Kevin Clark. 2019. What Does Bert Look At? An Analysis of Bert's Attention. *arXiv preprint arXiv:1906.04341*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower Models for Vision-and-language Navigation. *Advances in neural information processing systems*, 31.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. 2022. Bridging the Gap between Learning in Discrete and Continuous Environments for Vision-and-language Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15439–15449.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. 2021. VLN BERT: A Recurrent Vision-and-language BERT for Navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1643–1653.

Jingyang Huo, Qiang Sun, Boyan Jiang, Haitao Lin, and Yanwei Fu. 2023. GeoVLN: Learning Geometry-enhanced Visual Representation with Slot Attention for Vision-and-language Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23212–23221.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned Token Pruning for Transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.

Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. 2021. Waypoint Models for Instruction-guided Navigation in Continuous Environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15162–15171.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. 2020. Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments. In *European Conference on Computer Vision*, pages 104–120. Springer.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-across-room: Multilingual Vision-and-language Navigation with Dense Spatiotemporal Grounding. *arXiv preprint arXiv:2010.07954*.

Jialu Li and Mohit Bansal. 2023a. Improving vision-and-language navigation by generating future-view image semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10803–10812.

Jialu Li and Mohit Bansal. 2023b. PanoGen: Text-conditioned Panoramic Environment Generation for Vision-and-language Navigation. *Advances in Neural Information Processing Systems*, 36:21878–21894.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint arXiv:2202.07800*.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. 2020. REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991.

Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, and Qi Wu. 2023a. Hop+: History-enhanced and order-aware pre-training for vision-and-language navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):8524–8537.

Yanyuan Qiao, Zheng Yu, and Qi Wu. 2023b. VLN-PETL: Parameter-Efficient Transfer Learning for Vision-and-Language Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15443–15452.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning Transferable Visual Models from Natural Language Supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout. *arXiv preprint arXiv:1904.04195*.

Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. IEEE.

Hongjie Wang, Bhishma Dedhia, and Niraj K Jha. 2024a. Zero-TPrune: Zero-shot Token Pruning Through Leveraging of the Attention Graph in Pre-trained Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16070–16079.

Liuyi Wang, Zongtao He, Ronghao Dang, Mengjiao Shen, Chengju Liu, and Qijun Chen. 2024b. Vision-and-Language Navigation via Causal Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13139–13150.

Liuyi Wang, Zongtao He, Mengjiao Shen, Jingwei Yang, Chengju Liu, and Qijun Chen. 2024c. Magic: Meta-ability Guided Interactive Chain-of-distillation for Effective-and-efficient Vision-and-language Navigation. *arXiv preprint arXiv:2406.17960*.

Liuyi Wang, Zongtao He, Jiagui Tang, Ronghao Dang, Naijia Wang, Chengju Liu, and Qijun Chen. 2023a. A Dual Semantic-aware Recurrent Global-adaptive Network for Vision-and-Language Navigation. *arXiv preprint arXiv:2305.03602*.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. 2023b. GridMM: Grid Memory Map for Vision-and-language Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15625–15636.

Siyuan Wei, Tianzhu Ye, Shen Zhang, Yao Tang, and Jiajun Liang. 2023. Joint Token Pruning and Squeezing towards More Aggressive Compression of Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2092–2101.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. 2024. SparseVLM: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*.

Yusheng Zhao, Jinyu Chen, Chen Gao, Wenguan Wang, Lirong Yang, Haibing Ren, Huaxia Xia, and Si Liu. 2022. Target-driven Structured Transformer Planner for Vision-language Navigation. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4194–4203.

Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun Chang, and Xiaodan Liang. 2021. Soon: Scenario

Oriented Object Navigation with Graph-based Exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12689–12699.

Junyou Zhu, Yanyuan Qiao, Siqi Zhang, Xingjian He, Qi Wu, and Jing Liu. 2024. MiniVLN: Efficient Vision-and-Language Navigation by Progressive Knowledge Distillation. *arXiv preprint arXiv:2409.18800*.

Dataset	R2R		RxR-english		REVERIE	
	Seen	Unseen	Seen	Unseen	Seen	Unseen
Size	1020	2349	2939	4551	1423	3521
History nodes size	14		19		16	
View tokens	38		38		42	
Instruction tokens	32		127		20	

Table 8: Statistics of R2R, RxR-English, and REVERIE regarding token pruning. The numbers are averaged over all navigation tasks and steps. R2R is easier compared to RxR-English and REVERIE given its path and instruction lengths.

Method	Tokens	Validation Unseen				
		SR↑	SPL↑	RGS↑	RGSPL↑	FLOPS%↓
HAMT	-	32	29	19	17	100
HAMT (FastV)	70%	**32**	29	18	16	73
	60%	30	27	16	14	68
	50%	27	24	14	12	65
HAMT (NAP^{-})	70%	31	29	**19**	**17**	**69**
	60%	**31**	**28**	**18**	**17**	**64**
	50%	**29**	**26**	**17**	**15**	**59**
DUET	-	47	34	32	23	100
DUET (FastV)	70%	39	28	23	16	82
	60%	34	24	19	13	73
	50%	30	20	16	11	64
DUET (NAP)	70%	**47**	**33**	**31**	**22**	**80**
	60%	**45**	**32**	**31**	**22**	**70**
	50%	**43**	**30**	**28**	**20**	**62**

Table 9: Performance of pruning strategies on REVERIE data NAP^{-} indicates pruning without BTP. Efficacy is measured in Success Rate (SR) Per Length (SPL), Remote Grounding Success rate (RGS), and Remote Grounding Success rate Per Length (RGSPL). The results show that NAP is also effective for different navigation tasks such as remote object localization.

A VLN Datasets Statistics

The statistic of VLN dataset statistic is given in Table 8. The tokens numbers are averaged over steps of all navigation tasks. The statistics show that R2R tasks are easier with shorter paths and instructions. On the other hand, RxR-English tasks contain longer instructions and paths.

B NAP Performance of HAMT and DUET Models on R2R and REVERIE Datasets

We also compared the NAP performance of HAMT and DUET models on the REVERIE (Table 9) and R2R (Table 10) datasets against FastV. In the HAMT setting, we find that NAP achieves comparable or superior results, even without BTP (since HAMT lacks a history mapping module). For example, on REVERIE (Table 9), NAP achieves greater FLOP reductions than FastV (6 pp to 4 pp) while maintaining similar or better performance

Method	Retain Rate	Validation Seen			Validation Unseen		
		SR↑	SPL↑	FLOPS%↓	SR↑	SPL↑	FLOPS%↓
HAMT	-	70	67	100	63	58	100
HAMT (FastV)	0.7	67	64	80	**60**	**54**	82
	0.6	63	60	72	56	50	75
	0.5	58	55	**65**	50	45	68
HAMT (NAP^{-})	0.7	**71**	**66**	**79**	59	53	**81**
	0.6	**69**	**65**	72	56	**51**	**74**
	0.5	**63**	**60**	66	**52**	**46**	68
DUET	-	79	73	100	72	60	100
DUET (FastV)	0.7	68	62	73	60	50	**73**
	0.6	62	56	66	54	44	**65**
	0.5	59	52	**56**	50	40	**55**
DUET (NAP)	0.7	**77**	**65**	**71**	**68**	**57**	75
	0.6	**75**	**69**	**64**	**66**	**55**	68
	0.5	**72**	**71**	57	**63**	**52**	59

Table 10: Model performance on the R2R dataset with NAP and FastV. NAP^{-} indicates pruning without BTP. The result shows that NAP can be adapted to different models while still outperforming FastV.

in both navigation (SR, SPL) and object localization (RGS, RGSPL) across token budgets ranging from 70% to 50%. The advantage of NAP is even more pronounced for the DUET model: NAP consistently yields a better trade-off, with navigation performance (SR) surpassing FastV by 8–13 pp, and FLOP reductions exceeding FastV by 2–3 pp. Similar observation can be found on R2R dataset as well (Table 10), showing that NAP benefits different VLN models, even without a history mapping.

C BTP combined with different view pruning strategies

We evaluated the FLOPS improvements when applying BTP respectively with cascade token pruning, FastV, and Token Merging (ToMe) across the R2R, RxR, and REVERIE datasets (see Fig. 10). Our findings show that incorporating BTP reduces FLOPS by approximately 5 percentage points at high BGP rates (e.g., 80%), while resulting in a success rate drop of less than 0.5 percentage points.

D Instruction Length Influence on Efficiency Improvement

VPP's efficiency gains are closely related to instruction length. Under the default settings for R2R, RxR, and REVERIE (36 views, 14 history nodes, and a 7-step navigation), we evaluated the efficiency improvement from a 50% instruction pruning rate across different instruction lengths (see Fig. 11). The results show that a 50-token instruction achieves a 10 percentage point FLOPS reduction, which increases to 20 percentage points for a 150-token instruction.

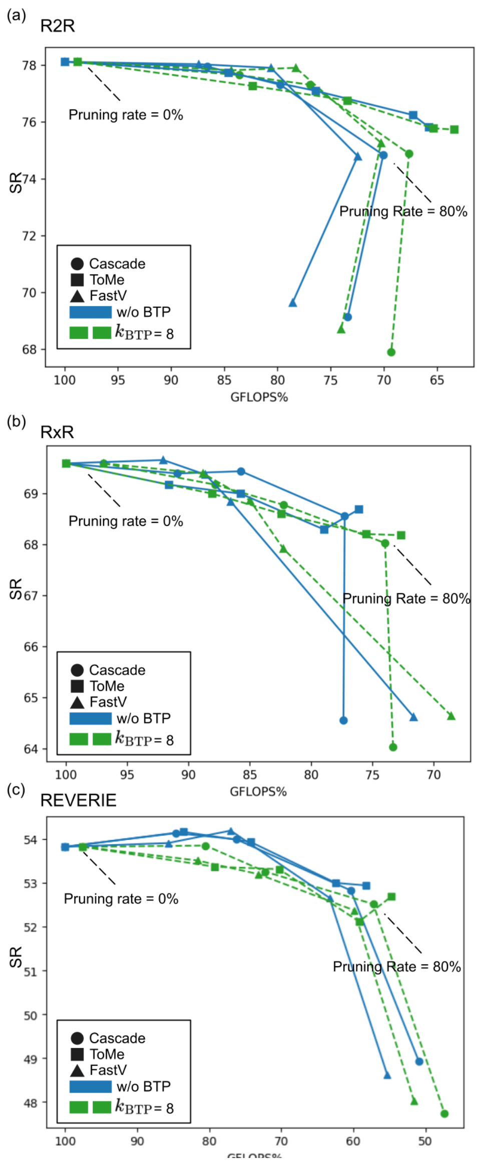

Figure 10: SR and FLOPS curves for different view pruning rates with and without BTP. Performance is evaluated on the R2R, RxR and REVERIE datasets.

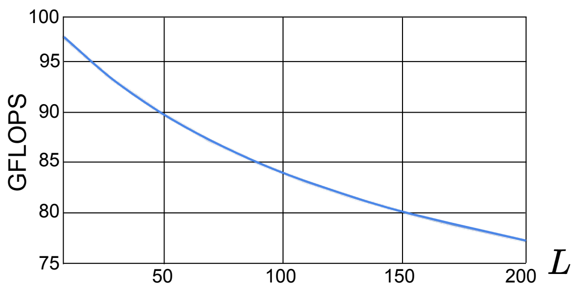

Figure 11: FLOPS curve over different instruction lengths with 50% token pruning rate. The curve shows that FLOPS reduced from textual input is heavily dependent on the instruction length, varying from 10 pp given 50 tokens to 20 pp given 150 tokens.

Method	Steps↓	SR↑	FLOPS%↓
No pruning	7	78.1	100.00%
SAS	6.8	76.8	96.45%
NAP (70%)	7.2	74.5	69.04%
NAP+SAS (70%)	7.5	72.3	71.22%
NAP	7.8	75.2	60.92%
NAP+SAS (60%)	7.9	72.2	61.53%
NAP	7.9	73.8	57.03%
NAP+SAS (50%)	8	71.2	57.72%

Table 11: A comparison with SAS and NAP under different pruning rates. NAP preserves higher SR than SAS, while FLOPS reduction is similar for both.

Method	Cross Attention	Transformer Layer
	FLOPS	FLOPS
SAS	$4.4 * 10^{-4}$	0.114
No SAS	$4.0 * 10^{-3}$	0.117

Table 12: Analysis of FLOPS reduced by applying SAS on a single transformer layer. SAS only saves very little (0.003) FLOPS in a VLN transformer layer.

E Comparison between Pruning Tokens and Skipping Attention Calculation Only

We observe that token-related computational cost arises from two sources: the linear projection ($O(ld^2)$) and the attention computation ($O(l^2d)$), where l denotes the token count and d the hidden feature size. Token pruning reduces both by decreasing l. As a lighter alternative, one can reduce FLOPs by shrinking l only during the attention computation while retaining the full token set in the projection layers. We term this approach Selective Attention Sum (SAS). SAS restricts attention from background (non-explorable) views to action (explorable) views, thereby skipping unnecessary computations.

We evaluate SAS in combination with NAP on the R2R unseen split (Table 11). SAS introduces a 1–3 pp drop in success rate (SR) and slightly increases navigation steps under tight token budgets, which offsets its marginal FLOPS savings.

To further analyze the impact of saving computation during attention computation, we compared the computation of a transformer layer with and without SAS in terms of GFLOPS, shown in Table 12. While SAS reduces attention score computations (from $4.0 * 10^{-3}$ to $4.4 * 10^{-4}$ GFLOPS across 36 views), the majority of computation in a transformer layer comes from linear projections.

Prompt	Given the following set of words: “walk”, “down”, ..., “landing”, can you point out which of them are irrelevant to the following types of information: 1. A direction to go; 2. Describing the environment; 3. Object(s) in the indoor/outdoor environments. Please don’t change the word in the quotation mark and explain why. Please answer in the following: format: “{word} : relevant/irrelevant {explanation}”	
Word	Relevance	Explanation
walk	relevant	direction to go
down	relevant	direction to go
one	irrelevant	not related to direction or environment description
flight	irrelevant	not related to indoor environment navigation
of	irrelevant	preposition, not related to direction or environment description
stairs	relevant	object in the environment
and	irrelevant	conjunction, not related to direction or environment description
stop	relevant	direction to go
the	irrelevant	article, not related to direction or environment description
landing	relevant	object in the environment

Table 13: Our prompt to Llama 3, and examples of words in the vocabulary of irrelevance, explained by the LLM why they are relevant/irrelevant.

As a result, applying SAS only reduces the FLOPS of a transformer layer from 0.117 to 0.114, which is just a 2.5% gain.

We attribute the limited efficiency gain of SAS to the relatively small number of view tokens ($l = 36$) compared to the hidden size ($d = 768$), which makes the cost of linear projections ($O(ld^2)$) dominant over attention computation ($O(l^2d)$). In contrast, token pruning reduces both attention and projection costs, making it more effective than SAS for improving efficiency.

F Vocabulary Example

We provide an example of how we prompt the LLM to construct the vocabulary of irrelevance, with explanations from Llama 3 why such tokens are classified to be relevant or irrelevant (see Table 13).