EfficientNav: Towards On-Device Vision-Language Navigation with Navigation Map Caching and Retrieval

Anonymous ACL submission

Abstract

Embodied agents equipped with large language models (LLMs) and online constructed navigation maps have demonstrated promising capability for zero-shot vision-language navigation (VLN) in unseen environments. However, existing agents heavily rely on giant LLMs on the cloud, e.g., GPT-4, while directly switching to small LLMs, e.g., LLaMA3.2-11b, suffer from significant success rate drops due to limited model capacity for understanding complex navigation maps, which prevents deploying VLN on local devices. At the same time, the long prompt introduced by the navigation map description will cause high planning latency on local device. In this paper, we propose EfficientNav to enable on-device zero-shot VLN for the first time. To help the smaller LLMs better understand the environment, we propose semantics-aware memory retrieval to prune redundant information in navigation maps. To reduce planning latency, we propose discrete memory caching and attention-based memory clustering to efficiently save and re-use the KV cache. Extensive experimental results demonstrate that EfficientNav achieves 11.1% improvement in success rate on Habitat ObjNav Challenge benchmark over GPT-4-based baselines, and demonstrates 6.7× real-time latency reduction and 4.7× end-to-end latency reduction over GPT-4 planner. Our code is available on Anonymous Github.

1 Introduction

Vision-language navigation (VLN) in an unseen environment is a crucial task for embodied agents (An et al., 2024; Sridhar et al., 2024). To enable zero-shot VLN, recent studies have proposed to leverage the common-sense reasoning and decision-making abilities of large language models (LLMs) (Chen et al., 2024; Cao et al., 2024). Specifically, LLMs act as planners and determine a series of navigation sub-goals based on the observed environmental information to guide the agent to the final destination (Dorbala et al., 2022). However, for long-horizon tasks, simple LLM-based agents are still insufficient as they make decisions based solely on the observations of local environments (Cai et al., 2024). To solve this, navigation maps are usually constructed online and incorporated into the prompts for LLM planning (Chen et al., 2024), thus providing a memory augmentation of the explored area and improving planning performance.

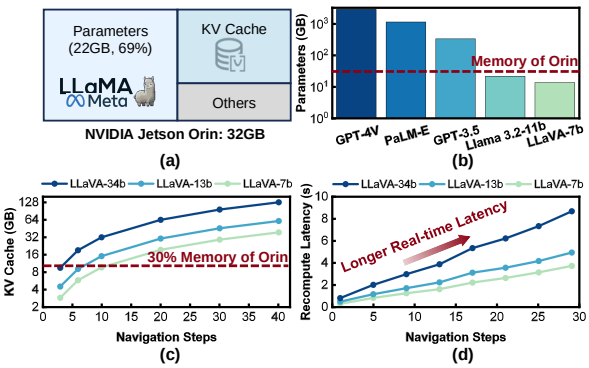


Figure 1: (a) Local devices have limited memory resources. Hence, (b) only smaller LLMs can be used as the planner, and (c) the KV cache of map information can not be fully saved. (d) While re-computing the KV cache will cause long real-time latency.

Though promising, existing LLM-based methods usually depend on giant LLMs, e.g., GPT-4, to achieve high performance, thus heavily relying on cloud offloading (Wake et al., 2023; Cai et al., 2024; Zhou et al., 2024). Hence, it suffers from high communication latency, and a heavy reliance on WiFi or cellular networks. This will introduce long real-time latency, negatively impacting robotic performance. (Khatib, 1986; Mei et al., 2006; Shah et al., 2023a). There will also be privacy concerns (Lin et al., 2024; Ren et al., 2023) and high energy cost (Bahrini et al., 2023; Wu et al., 2024b). Hence, there is a growing demand to develop on-device VLN systems, e.g., on local GPUs or application-specific chips (Karumbunathan, 2022).

However, running zero-shot VLN on local de-

vices is very challenging, as local devices have limited memory resources. For example, a single Jetson AGX Orin (Karumbunathan, 2022) only has 32GB DRAM. As shown in Figure 1, to meet this constraint, on the one hand, only smaller LLMs, e.g., LLaMA3.2-11b, instead of giant LLMs, can be used, while directly switching to smaller LLMs will cause significant success rate drop because of lower model capacity (Cao et al., 2024; Long et al., 2024). On the other hand, the KV cache of the navigation map description can not be fully saved in device memory. In multi-turn dialogues with the planner, this will cause high re-computation cost and long real-time latency (Jin et al., 2024).

In this paper, we propose EfficientNav, the first on-device memory-augmented VLN system that enables zero-shot in-door navigation. We observe that the navigation map description will cause long prompt length, whose KV cache can not be fully saved due to memory constraints. Hence, we only select part of the map description and load their KV cache for LLM. However, the KV cache will change with the context order. To enable reusing the saved KV cache, we propose discrete memory caching, clustering the map information into groups, and calculating KV cache for each group individually. Then only select partial groups for LLM. To reduce the impact of ignoring cross-attention between groups, we propose attention-based memory clustering, using LLM attention to cluster related information into the same group. We further observe that smaller LLMs can not fully understand complex navigation maps. So we propose semantics-aware memory retrieval to prune redundant map information in the group selection process, thus improving the performance of small LLMs. We summarize our contributions as follows:

- We propose discrete memory caching to prevent saving the KV cache of the whole map description and meet the memory constraints.
- We propose attention-based memory clustering to reduce the impact of ignoring cross-attention between groups.
- We propose semantics-aware memory retrieval to efficiently select the relevant groups and prune redundant map information.
- We conduct extensive experiments and show that EfficientNav can achieves 11.1% success rate improvements over GPT-4-based methods on Habitat ObjNav Challenge dataset, and show 6.7× real-time latency and 4.7× end-to-end latency reduction over GPT-4 planner.

Table 1: Comparison with prior-art methods.

Method	Zero-shot	LLM	On-device Inference	Memory Augmented
ViKiNG (Shah and Levine, 2022)	✗	-	✓	✓
NaVid (Zhang et al., 2024)	✗	Vicuna	✓	✗
Skip-SCAR (Liu and Zhang, 2024)	✗	-	✓	✓
Pixel Navigation (Cai et al., 2024)	✓	GPT-4	✗	✗
InstructNav (Long et al., 2024)	✓	GPT-4V	✗	✓
MapGPT (Chen et al., 2024)	✓	GPT-4	✗	✓
LFG (Shah et al., 2023b)	✓	GPT-4	✗	✓
EfficientNav **(Ours)**	✓	LLaMA	✓	✓

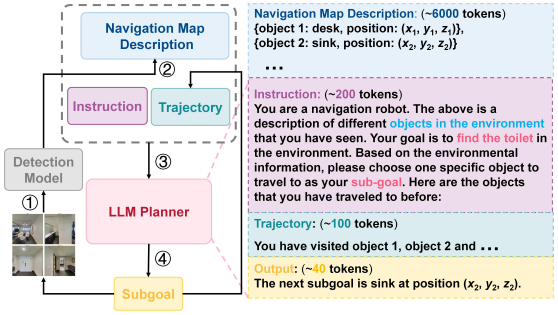
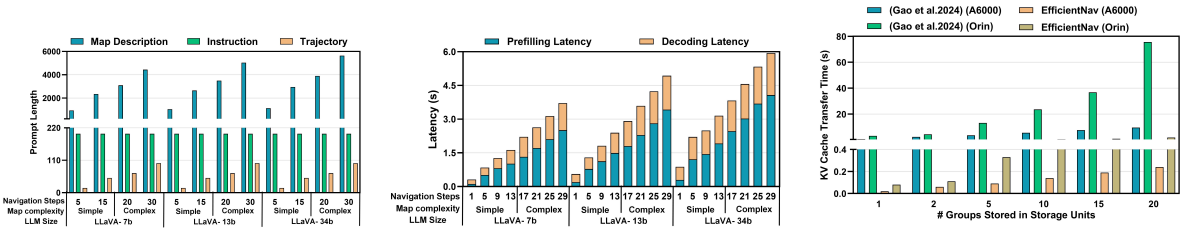


Figure 2: Typical flow of one navigation step.

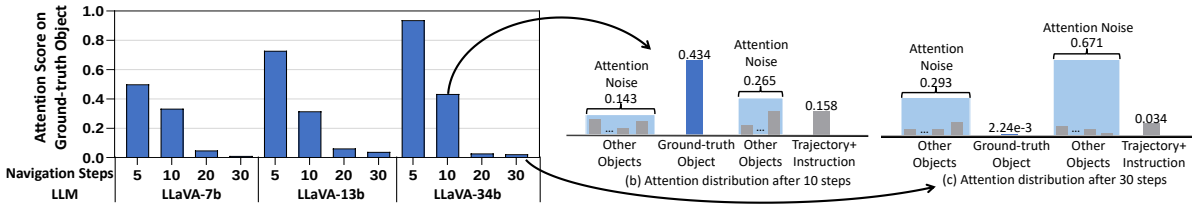
2 Background

Different from classical navigation with pre-built maps (Shah and Levine, 2022), in zero-shot navigation tasks, the robot faces inefficiencies during sub-goal planning due to its limited field of view (Cai et al., 2024). To overcome this, a navigation map is usually constructed online to provide a memory of the explored area (Chen et al., 2024). The flow of a navigation step (planning and achieving one sub-goal) is illustrated in Figure 2 (Chen et al., 2024; An et al., 2024; Zhou et al., 2024). First, after achieving the last sub-goal, the robot detects the surrounding objects. Second, the information of newly detected objects, e.g., position and semantics, is added to the navigation map. Then the navigation map description and the instructions are given to LLM as prompts for sub-goal planning. Finally, the LLM chooses one object in the navigation map as the next sub-goal. To achieve a high success rate, existing works (Chen et al., 2024; Long et al., 2024; Shah et al., 2023b) usually use giant models, e.g., GPT-4, as the planner, which prevents the navigation system deployed on local devices. While directly switching to smaller LLMs will cause significant success rate drops (Cao et al., 2024; Kim et al., 2024). At the same time, the amount of map information will increase in each step and introduce a long prompt length. Traditional LLM serv-



(a) (b) (c)

Figure 3: (a) Average length of different parts of prompt in different navigation steps. (b) On-device LLM planning latency in different navigation steps. (c) Memory transfer time with LLaVA-7b when using low-speed storage units.



(a)

Figure 4: (a) Attention score on the most promising sub-goal (ground-truth object) in different navigation steps, and the attention distribution after (b) 10 steps and (c) 30 steps using LLaVA-34b in sub-goal planning. As the amount of map information increases with the progression of navigation, the LLM can not fully understand the navigation map and focus on the most promising sub-goal. We also find this problem on other small open-source LLMs such as LLaMA and Vicuna. The ground-truth object is chosen by GPT-4o.

ing methods (Kwon et al., 2023; Zheng et al., 2023) re-computing the KV cache in each inference will cause long prefilling latency, while directly saving and reusing the KV cache of history information (Jin et al., 2024; Gao et al., 2024) can not meet the memory constraints of local devices.

As shown in Table 1, EfficientNav is the first work to enable accurate and efficient on-device zero-shot VLN. EfficientNav designs a memory selection mechanism to remove redundant information in the navigation map. This helps the smaller LLM focus on relevant information and improve success rate. For efficiency, EfficientNav designs a novel memory caching and clustering mechanism, which adaptively clusters objects in the navigation map into groups and only selects relevant groups for the planner, thus meeting memory constraints and enabling KV cache reuse. Note that our method is orthogonal and naturally compatible with quantization (Frantar et al., 2022), pruning (Ma et al., 2023), and other LLM compression methods.

3 EfficientNav: Memory Augmented On-device Navigation System

3.1 Motivation and Overview

Observation 1: tight memory constraints of local device limit the saving of KV cache of navigation map description. As discussed in Section 2 and shown in Figure 3 (a), with the progression of navigation, the amount of map information rapidly increases, thus introducing long prompt length, usually up to thousands of tokens. And re-computing the KV cache in each planning process will cause long prefilling time, which is shown in Figure 3 (b). To avoid this cost, (Jin et al., 2024) saves the KV cache of history information in server memory, while as shown in Figure 1 (c), this can not meet the memory constraints of local devices. So a new memory caching mechanism is needed.

Observation 2: the tight memory constraint forces to use smaller LLMs, which have lower model capacity and cause success rate drop. To meet the memory constraints of local device, usually tens of gigabyte, we need to use smaller LLMs, e.g., LLaVA-7b. However, directly using them to replace giant LLMs, such as GPT-4, will cause significant success rate drop (Cao et al., 2024; Kim et al., 2024). In practice, we find this comes from that the smaller LLM with lower capacity can not fully understand complex navigation maps. With the progression of navigation, the LLM can not pay its attention to the important information among the huge amount of map information and choose the correct sub-goal, which is shown in Figure 4. So a memory selection strategy is needed to remove the redundant information for the smaller LLM.

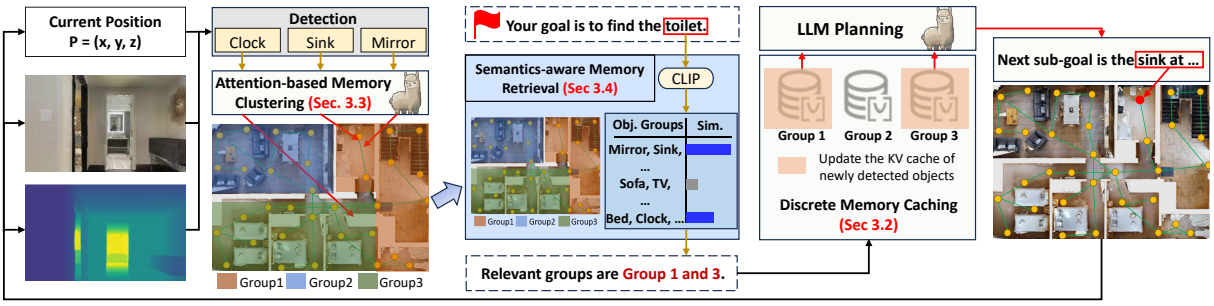


Figure 5: Overview of EfficientNav. Sim. stands for similarity.

EfficientNav Overview Based on these observations, we propose EfficientNav, the first work enabling on-device zero-shot VLN. Figure 5 shows the overview of EfficientNav. To meet the memory constraints, we propose discrete memory caching to cluster objects into groups and calculate the KV cache of each group individually. Then only select a portion of groups to LLM and load their KV cache to device memory. To accurately cluster information and improve success rate, we propose attention-based memory clustering, using LLM attention to guide group clustering. To help the smaller LLMs better understand the navigation map, in group selection process, we propose semantic-aware memory retrieval to efficiently prune redundant map information.

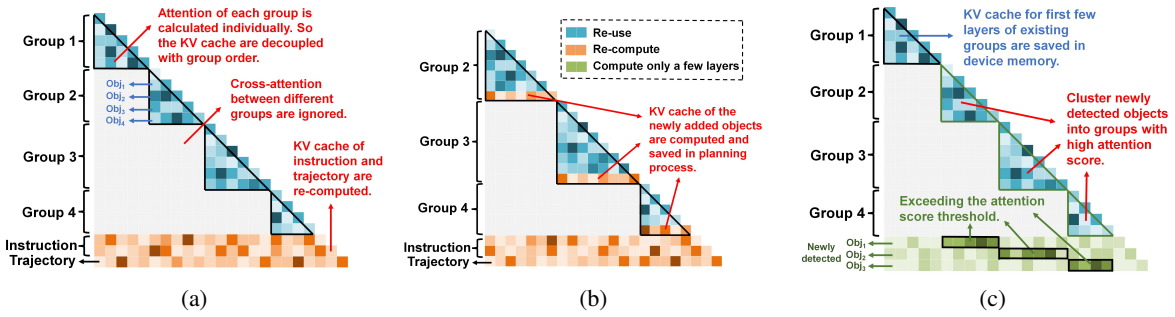
3.2 Discrete Memory Caching

As discussed in Section 3.1, the memory constraints of local devices prevent saving KV cache of all navigation map descriptions. To solve this problem, (Gao et al., 2024) saves the KV cache in storage units with low speed but large storage capacity, e.g., CPU host memory for NVIDIA RTX A6000 or disk for Jetson Orin, and load the KV cache to the device memory when a request comes. However, as local memory can not hold all the KV cache at the same time, this method needs to re-load the KV cache to device memory in each decoding phase. However, in each planning process, LLM needs to decode multiple tokens (around 40 in practice). The frequent communication between device memory and storage units will also cause huge latency, which is shown in Figure 3 (c).

To meet the memory constraint and avoid re-computation and frequent memory transfer, we propose discrete memory caching. Following (Gao et al., 2024), we also save the KV cache in low-speed storage units and load the required KV cache into device memory. However, as the navigation map contains redundancy, we only select partial important information according to the memory budget and load KV cache only once, thus reducing transfer cost.

However, when selecting information, the order of context in prompt will change. When pre-calculating the KV cache, existing works (Liu et al., 2024; Zheng et al., 2023) calculate the full attention of the whole context. Then only the KV cache of its longest common prefix with the selected information can be used. The KV cache after the changed position needs to be re-computed. To solve this, shown in Figure 6 (a), we cluster objects in navigation map into groups and compute KV cache of each group individually. Then only select partial groups to prompt the LLM planner. Hence, we can decouple the group KV cache and the group order, thus making full usage of saved KV cache and avoiding re-computation. As shown in Figure 5, in each navigation step, newly detected objects will be added to existing groups, and we need to add their KV cache to the group KV cache. To avoid changing context order within the group and impacting the existing KV cache, we concatenate the information of new objects at the end of the group information. As shown in Figure 6 (b), in the planning process, when groups with newly added objects are selected, the KV cache of these objects can be directly calculated and saved, without introducing extra computation. We follow the position encoding strategy in LLM inference as (Gim et al., 2024).

However, discrete memory caching will cause the ignorance of cross-attention between groups, and this may cause LLM performance drop (Yao et al., 2024; Hu et al., 2024). We also need to ensure that the groups with important information are not removed in group selection. So correct memory clustering and memory selection is very important. We will explain our design in Section 3.3 and 3.4.



(a) (b) (c)

Figure 6: (a) When using discrete memory caching, the KV cache of existing groups can be re-used in LLM planning. (b) When groups with newly added objects are selected, in LLM planning process, the KV cache of these objects can be directly calculated and saved without impacting existing KV cache. (c) When using attention-based memory clustering, the newly detected objects will be clustered into different groups according to attention distribution.

3.3 Attention-based Memory Clustering

As discussed in Section 3.2, to reduce the impact of ignoring cross-attention and maintain a high success rate, we need to cluster the information accurately and efficiently. Existing works (Jin et al., 2024; Yao et al., 2024) meanly divide the entire context into several chunks with the same length. However, this does not take into account the relationships between related objects in the navigation map. For example, the relationship between the oven and the pot is closer than the relationship between the oven and the bed. By grouping objects with closer relationships together and calculating their cross-attention, LLMs can better understand the navigation map and abstract the environment of a larger area. At the same time, the grouping granularity is also important. If the granularity is too fine, more cross-attention will be ignored, thus impacting accuracy. If the granularity is too coarse, the KV cache size of each group will become large. Hence, only a few groups can be selected for the LLM, which will enhance the difficulty of selection and may neglect important information.

To adaptively control object clustering and group granularity, we use the attention of LLM itself to cluster the newly detected objects into different groups. As shown in Figure 6 (c), in the clustering process, we input the information of existing groups and the newly detected objects into LLM, and only infer a few layers (in practice, we find around $\frac{1}{10}$ layers of the whole model is enough). If the mean attention between a newly detected object and an existing group exceeds a specific threshold, we cluster this object into the group. Otherwise, the remaining objects will be added to a new group. For the first group, we simply cluster the detected objects in the first navigation step into a group.

By using attention-based memory clustering, we can reduce the difference between discrete attention and full attention. Note that the clustering process will not introduce much extra computation and latency. This is because we only compute the first few layers of LLM, and the KV cache of existing groups has been pre-computed and its cache of the first few layers are kept in device memory.

3.4 Semantics-aware Memory Retrieval

As discussed in Section 3.1, the smaller LLM cannot fully understand complex navigation maps. So a memory selection mechanism is needed to remove redundant information. (Shah et al., 2023b) empirically selects environmental information based on object positions. However, we find when the final goal changes, the relevance of each group to the final goal also changes, which is shown in Figure 7. The empirical selection can not adapt to different final goals, thus showing lower performance. At the same time, it does not consider different device memory budgets. (Cao et al., 2024) uses LLM to select relevant information, while this will introduce extra LLM calls and long real-time latency.

Based on this, we propose semantics-aware memory retrieval, efficiently selecting groups according to semantic information to adapt to different sub-goals. As the task of removing redundant information is much easier than finding a specific sub-goal, to improve efficiency, we conduct group selection and sub-goal planning in a small-large model collaboration mode. For the easier group selection, we use CLIP model (Dorbala et al., 2022) instead of LLM, which only has around 100M parameters and lower inference latency. For the harder sub-goal planning, we use LLM to select one specific object as the sub-goal. In the information se-

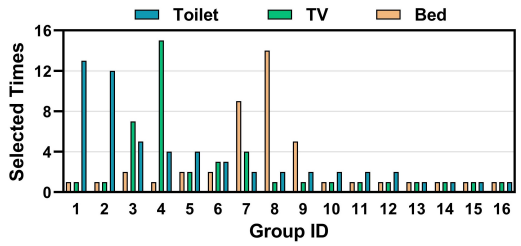


Figure 7: For different final goals, the selected times of different groups are different, thus showing different relevance. For each final goal, the navigation task is conducted 15 times with different starting points. The groups are selected by GPT-4o.

lection process, we use CLIP to encode the object information in each group and the final goal. And then calculate the similarity between the encoding results between the final goal and each group, which is viewed as the possibility for a group including potential sub-goals. We consider the group relevant to the final goal only if the possibility is larger than a threshold. To adapt to different devices with different memory budgets, we formulate the group selection as a knapsack problem:

$$\text{maximize} \quad P = \sum_{i=1}^{n} (P_i - threshold) \cdot x_i \quad (1)$$

$$\text{subject to} \quad \sum_{i=1}^{n} M_i \cdot x_i \leq M, \;\; x_i \in \{0, 1\} \quad (2)$$

where P_i denotes the possibility to include sub-goals in group i, M_i and M denotes the memory usage of KV cache of group i and device memory budget, x_i denotes whether to select group i, n denotes the number of groups. After solving the knapsack problem, we can select truly relevant information for LLM and meet the memory budget at the same time.

By using semantics-aware memory retrieval, we can efficiently select relevant groups adapting to different final goals and devices. Note that in each planning process, we only update the decoding results of the groups with newly detected objects. As the inference time of CLIP is much lower than LLM, the latency of the selection process is negligible. At the same time, between two adjacent planning processes, the navigation map will not change much. So it is likely that some group are both selected in the current step and last step. Hence, their KV cache has been loaded into device memory, thus reducing the KV cache loading time, which we will further show in our experiments.

4 Experiments

4.1 Experiment Setup

Dataset and Evaluation metric. We evaluate EfficientNav on the Habitat ObjNav Challenge dataset (Andrew et al., 2022) based on the Habitat simulation platform (Szot et al., 2021). In the simulation platform, the robot can access RGBD observation of the environment. In each task, the robot is placed at a different starting point in the environment and is only instructed to find a specific object, e.g., "TV", "chair", "sofa" etc., which is harder than tasks giving detailed, step-by-step directions (Anderson et al., 2018; Qi et al., 2020). We evaluate our method on the evaluation set, which has 20 scenes. For each scene, we evaluate 15 tasks.

For accuracy, we report two metrics: i) the average success rate (SR), and ii) the success rate penalized by path length (SPL), which both evaluates the accuracy and the efficiency of robot trajectory. For system efficiency, we report two metrics: i) real-time latency (RtL): the average robot planning time in one navigation step. ii) End-to-end latency (E2EL): the average latency for the robot to complete one navigation task (including multiple navigation steps and robot moving time).

Implementation In EfficientNav, we use smaller open-source planners, such as LLaVA and LLaMA3.2. We use a single NVIDIA RTX A6000 GPU to deploy LLaVA-7b and LLaMA3.2-11b. When using LLaVA-13b and LLaVA-34b, we deploy our system on 2 and 4 NVIDIA RTX A6000 GPUs respectively. We also deploy LLaVA-7b on a single Jetson AGX Orin.

4.2 Main Results

Comparison with state-of-the-art. The comparison of SR and SPL is shown in Table 2. Compared with learning-based methods, EfficienNav achieves 18.0% SR and 14.7% SPL improvements over the prior-art method OVRL (Yadav et al., 2023), without any training cost. This shows the advantage of LLM-based navigation system, which we have discussed in Section 1. Compared with zero-shot methods, EfficientNav achieves 11.1% SR and 5.5% SPL improvements over LFG (Shah et al., 2023b), which uses GPT-4. Compared with naive LLaVA-34b planner (Chen et al., 2024), EfficientNav achieves 37.3% SR and 20.5% SPL improvements. Because we use semantics-aware memory retrieval, helping the LLM focus on the most rele-

vant groups and removing redundant information.

The latency comparison is shown in Table 3. Compared with GPT-4 planner (Chen et al., 2024), EfficientNav achieves 6.7× real-time latency reduction and 4.7× end-to-end latency reduction, by saving the communication time to the cloud server. Compared with naive LLaVA planner, EfficcientNav achieves 8.8× and 6.5× real-time latency reduction and 3.7× and 4.4× end-to-end latency reduction on LLaMA3.2-11b and LLaVA-34b. This mainly comes from using discrete memory caching to avoid re-computation in the prefilling stage of planning. Prior-art solutions such as vllm can accelerate LLM inference, but they can not solve the problem of high re-computation cost, which is the main bottleneck. Compared with vllm, EfficientNav achieves 6.5× and 5.1× real-time latency reduction and 3.1× and 3.8× end-to-end latency reduction on LLaMA3.2-11b and LLaVA-34b.

Real-time latency in different navigation steps. The amount of map information increases in each navigation step. As shown in Figure 8, when using traditional serving methods (Kwon et al., 2023), the increasing prompt length will introduce high recomputation cost and increasing real-time latency. However, when using EfficientNav, the real-time latency stabilizes after a certain navigation step, as we use semantics-aware memory retrieval to select partial groups for LLM according to memory budget. With the same amount of information given to LLM, EffcientNav also shows lower real-time latency, as discrete memory caching saves the re-computation time of LLM prefilling.

Planning latency breakdown. Figure 9 shows the latency breakdown of one navigation step. Compared to traditional serving methods (Kwon et al., 2023), by avoiding re-computation, discrete memory caching can significantly reduce the prefilling time of planning by around 20×. And semantics-aware memory retrieval removes the redundant information, thus reducing the prompt length and reducing the computation cost and latency of the decoding stage of planning. The latency of memory clustering and memory selection is negligible, which we discussed in Section 3.3 and 3.4.

Memory caching distribution and cache hit rate of EfficientNav. When using EfficeintNav, in each planning process, we only select relevant groups for LLM. With larger device memory budgets for KV cache, we can store more groups in device memory. As discussed in Section 3.4, some selected groups may be just stored in device memory, so we do not need to re-load these groups from storage units. We define the proportion of these groups as cache hit rate. A high hit rate will lead

Table 2: SR and SPL comparison on Habitat ObjNav Challenge dataset.

Method	Zero-shot	LLM	SR	SPL
DD-PPO (Wijmans et al., 2019)	✗	-	27.9	14.2
SemExp (Chaplot et al., 2020)	✗	-	37.9	18.8
Habitat-web (Ramrakhya et al., 2022)	✗	-	41.5	16.0
L3MVN (Yu et al., 2023)	✗	-	50.4	23.1
OVRL (Yadav et al., 2023)	✗	-	62.0	26.8
ZSON (Majumdar et al., 2022)	✓	-	25.5	12.6
PixelNav (Cai et al., 2024)	✓	GPT-4	37.9	20.5
ESC (Zhou et al., 2023)	✓	-	39.2	22.3
VoroNav (Wu et al., 2024a)	✓	GPT-3.5	42.0	26.0
LLaVA Planner-34b (Chen et al., 2024)	✓	LLaVA-34b	42.7	21.0
InstructNav (Long et al., 2024)	✓	GPT-4V	58.0	20.9
LFG (Shah et al., 2023b)	✓	GPT-4	68.9	36.0
EfficientNav-11b	✓	LLaMA3.2-11b	**74.2**	**39.5**
EfficientNav-34b	✓	LLaVA-34b	**80.0**	**41.5**

Table 3: Average latency comparison on Habitat ObjNav Challenge dataset.

Method	LLM	RtL	E2EL
GPT-4 Planner (Chen et al., 2024)	GPT-4	5.80s	59.34s
LLaMA Planner-11b (Chen et al., 2024)	LLaMA3.2-11b	3.07s	46.40s
vllm (Kwon et al., 2023)	LLaMA3.2-11b	2.27s	39.78s
EfficientNav-11b (**Ours**)	LLaMA3.2-11b	0.35s	12.70s
LLaVA Planner-34b (Chen et al., 2024)	LLaVA-34b	5.63s	55.32s
vllm (Kwon et al., 2023)	LLaVA-34b	4.43s	47.95s
EfficientNav-34b (**Ours**)	LLaVA-34b	0.87s	12.51s

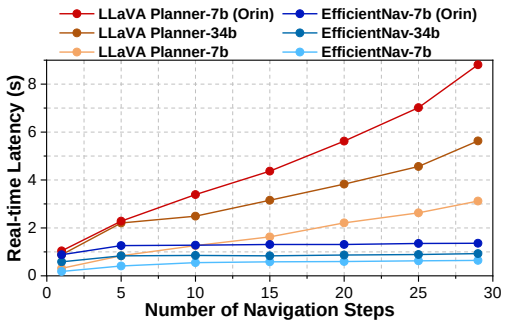


Figure 8: Comparison of real-time latency in different navigation steps.

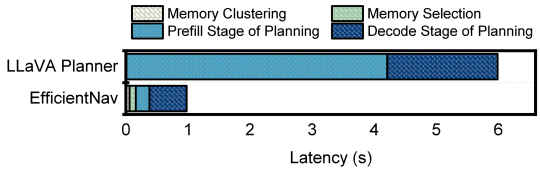


Figure 9: Planning latency (s) breakdown for LLaVA planner and EfficientNav. The cache loading time in EfficientNav is included in prefilling stage of planning.

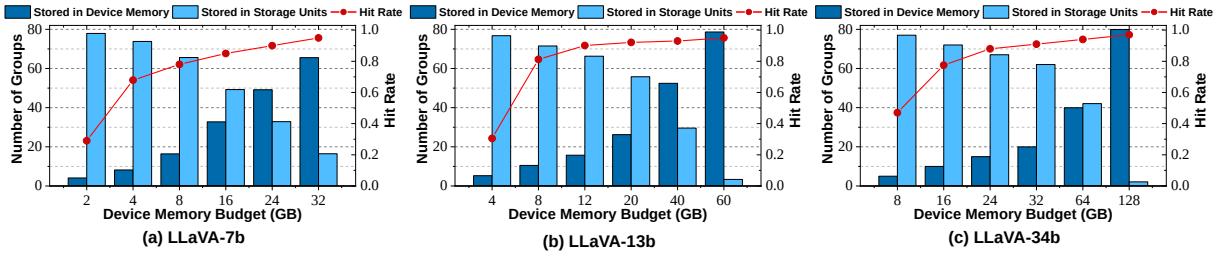


Figure 10: Comparison on the memory caching distribution and memory cache hit rate with different device memory budgets for KV cache, using (a) LLaVA-7b, (b) LLaVA-13b, and (c) LLaVA-34b. We define the proportion of selected groups already stored in device memory as cache hit rate.

to low caching loading time. As shown in Figure 10, the cache hit rate of EfficientNav increases with the device memory budget and rapidly reaches a high level. This is because when the memory budget increases, more relevant groups are stored in device memory. At the same time, the semantics of each group will not change much between adjacent planning processes. So it is likely that some relevant groups are just loaded to device memory in the last few steps, leading to a high cache hit rate.

4.3 Ablation Study

Individual influence of our methods. Table 4 shows the individual influence of our proposed methods. In naive LLaVA planner, we empirically cluster and select environmental information according to object positions (Zhang and Ma, 2024; Shah et al., 2023b). By using discrete memory caching, compared with naive LLaVA planner (Chen et al., 2024), we achieve 2.3× real-time latency and 1.5× end-to-end latency reduction, by re-using KV cache of navigation map description and avoiding re-computation. By using attention-based memory clustering, we further achieve 20.2% SR and 13.2% SPL improvement, by fully considering the relationship between objects and reducing the impact of ignoring cross-attention. By using semantics-aware memory retrieval, we achieve 16.7% SR and 7.3% SPL improvement, by selecting groups according to semantics and pruning redundancy. We also achieve 2.7× real-time latency and 2.6× end-to-end latency reduction, by selecting groups according to memory budget and avoiding frequent communication with storage units.

Impact of device memory budget for KV cache. In semantics-aware memory retrieval, we select relevant groups according to the device memory budget. Here we evaluate the impact of memory budget on success rate and latency using LLaVA-34b. As shown in Table 5, on most cases, our method shows good robustness. However, when the memory budget for KV cache is too small, fewer relevant groups can be selected. Hence, the potential sub-goal may be ignored, which causes success rate drops and longer end-to-end latency. For a larger memory budget, more relevant groups can be selected. While the longer prompt length may cause longer planning time, which will slightly increase the real-time latency.

Table 4: Ablation study on EfficientNav methods with LLaVA-34b.

Method	SR	SPL	RtL	E2EL
LLaVA Planner	42.7	21.0	5.63s	55.32s
+Discrete Memory Caching	43.1	21.0	2.42s	36.94s
+Attention-based Memory Clustering	63.3	34.2	2.32s	32.58s
+Semantics-aware Memory Retrieval	80.0	41.5	0.87s	12.51s

Table 5: Ablation study on different device memory budgets for KV cache with LLaVA-34b.

Memory Budget	SR	SPL	RtL	E2EL
16GB	74.7	37.3	0.59s	14.24s
24GB	79.0	39.1	0.71s	14.29s
32GB	80.0	41.5	0.87s	12.51s
40GB	80.3	41.9	0.93s	11.72s

5 Conclusion

This work proposes EfficientNav, the first work enabling on-device zero-shot VLN. We propose discrete memory caching to enable re-using KV cache and avoid re-computation in LLM planning. We also propose attention-based memory clustering to reduce the impact of ignoring cross-attention. To improve planning performance of smaller LLMs, we propose semantics-aware memory retrieval to remove the redundancy in navigation map. EfficientNav overcomes all existing baselines and first enables to run zero-shot VLN on local devices.

6 Limitations

In this section, we discuss the current limitations and potential avenues for future research.

First of all, in our method, we propose the first on-device VLN system that enables efficient zero-shot in-door navigation. In fact, our memory caching and memory retrieval method also benefits the LLM inference on the cloud server by avoiding re-computation and removing redundant information. However, EfficientNav can not solve the problems of high communication latency or privacy concerns that cloud serving faces.

Also, in our experiment, EfficientNav achieves 6.7× real-time latency reduction compared with GPT-4 planner. However, as the inference speed of LLM can not reach that of small models, so EfficientNav can not be used in applications that need extremely low real-time latency.

7 Ethics Statement

In our paper, our method first enables efficient on-device zero-shot in-door VLN, overcoming the tight memory constraints by designing novel memory caching and retrieval mechanisms. Our method democratized VLN to local devices and can enable AI applications such as rescue robots to areas with poor connections, protecting privacy and saving energy at the same time. However, our LLM-based VLN works in a zero-shot manner. This can avoid the heavy training cost, but the pre-trained LLM may not master some highly specialized knowledge. Therefore, in high-risk areas such as chemical laboratories with hazardous materials, EfficientNav should be carefully used.

References

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. 2024. Etpnav: 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.

Szot Andrew, Yadav Karmesh, Clegg Alex, Berges Vincent-Pierre, Gokaslan Aaron, Chang Angel, Savva Manolis, Kira Zsolt, and Batra Dhruv. 2022. Habitat rearrangement challenge 2022. https://aihabitat.org/challenge/2022_rearrange.

Aram Bahrini, Mohammadsadra Khamoshifar, Hossein Abbasimehr, Robert J Riggs, Maryam Esmaeili, Rastin Mastali Majdabadkohne, and Morteza Pasehvar. 2023. Chatgpt: Applications, opportunities, and threats. In *2023 Systems and Information Engineering Design Symposium (SIEDS)*, pages 274–279. IEEE.

Wenzhe Cai, Siyuan Huang, Guangran Cheng, Yuxing Long, Peng Gao, Changyin Sun, and Hao Dong. 2024. Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5228–5234. IEEE.

Yihan Cao, Jiazhao Zhang, Zhinan Yu, Shuzhen Liu, Zheng Qin, Qin Zou, Bo Du, and Kai Xu. 2024. Cognav: Cognitive process modeling for object goal navigation with llms. *arXiv preprint arXiv:2412.10439*.

Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. 2020. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258.

Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee Wong. 2024. Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9796–9810.

Vishnu Sashank Dorbala, Gunnar Sigurdsson, Robinson Piramuthu, Jesse Thomason, and Gaurav S Sukhatme. 2022. Clip-nav: Using clip for zero-shot vision-and-language navigation. *arXiv preprint arXiv:2211.16649*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. 2024. {Cost-Efficient} large language model serving for multi-turn conversations with {CachedAttention}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 111–126.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. 2024. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338.

Junhao Hu, Wenrui Huang, Haoyi Wang, Weidong Wang, Tiancheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. 2024.

Epic: Efficient position-independent context caching for serving large language models. *arXiv preprint arXiv:2410.15332*.

Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Xin Liu, Xuanzhe Liu, and Xin Jin. 2024. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *arXiv preprint arXiv:2404.12457*.

Zhao Kaichen, Song Yaoxian, Zhao Haiquan, Liu Haoyu, Li Tiefeng, and Li Zhixu. 2024. Towards coarse-grained visual language navigation task planning enhanced by event knowledge graph. *arXiv preprint arXiv:2408.02535*.

Leela S Karumbunathan. 2022. Nvidia jetson agx orin series.

Oussama Khatib. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98.

Taewoong Kim, Byeonghwi Kim, and Jonghyun Choi. 2024. Multi-modal grounded planning and efficient replanning for learning embodied agents with a few examples. *arXiv preprint arXiv:2412.17288*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Chenqi Lin, Tianshi Xu, Zebin Yang, Runsheng Wang, Ru Huang, and Meng Li. 2024. Fastquery: Communication-efficient embedding table query for private llm inference. *arXiv preprint arXiv:2405.16241*.

Shu Liu, Asim Biswal, Audrey Cheng, Xiangxi Mo, Shiyi Cao, Joseph E Gonzalez, Ion Stoica, and Matei Zaharia. 2024. Optimizing llm queries in relational workloads. *arXiv preprint arXiv:2403.05821*.

Yaotian Liu and Jeff Zhang. 2024. Skip-scar: A modular approach to objectgoal navigation with sparsity and adaptive skips. *arXiv preprint arXiv:2405.14154*.

Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. 2024. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. 2022. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *Advances in Neural Information Processing Systems*, 35:32340–32352.

Yongguo Mei, Yung-Hsiang Lu, Yu Charlie Hu, and CS George Lee. 2006. Deployment of mobile robots with energy and timing constraints. *IEEE Transactions on robotics*, 22(3):507–522.

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.

Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. 2022. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5173–5183.

Xuanle Ren, Zhaohui Chen, Zhen Gu, Yanheng Lu, Ruiguang Zhong, Wen-Jie Lu, Jiansong Zhang, Yichi Zhang, Hanghang Wu, Xiaofu Zheng, et al. 2023. Cham: A customized homomorphic encryption accelerator for fast matrix-vector product. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE.

Deval Shah, Ningfeng Yang, and Tor M Aamodt. 2023a. Energy-efficient realtime motion planning. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–17.

Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. 2023b. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*, pages 2683–2699. PMLR.

Dhruv Shah and Sergey Levine. 2022. Viking: Vision-based kilometer-scale navigation with geographic hints. *arXiv preprint arXiv:2202.11271*.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. 2023c. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. 2024. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. 2021. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266.

Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. 2023. Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration. *arXiv preprint arXiv:2311.12015*.

Zixuan Wang, Bo Yu, Junzhe Zhao, Wenhao Sun, Sai Hou, Shuai Liang, Xing Hu, Yinhe Han, and Yiming Gan. 2024. Karma: Augmenting embodied ai agents with long-and-short term memory systems. *arXiv preprint arXiv:2409.14908*.

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. 2019. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*.

Pengying Wu, Yao Mu, Bingxian Wu, Yi Hou, Ji Ma, Shanghang Zhang, and Chang Liu. 2024a. Voronav: Voronoi-based zero-shot object navigation with large language model. *arXiv preprint arXiv:2401.02695*.

Xiaodong Wu, Ran Duan, and Jianbing Ni. 2024b. Unveiling security, privacy, and ethical concerns of chatgpt. *Journal of Information and Intelligence*, 2(2):102–115.

Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. 2023. Offline visual representation learning for embodied navigation. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. 2024. Cacheblend: Fast large language model serving with cached knowledge fusion. *arXiv preprint arXiv:2405.16444*.

Bangguo Yu, Hamidreza Kasaei, and Ming Cao. 2023. L3mvn: Leveraging large language models for visual target navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3554–3560. IEEE.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and Wang He. 2024. Navid: Video-based vlm plans the next step for vision-and-language navigation. *arXiv preprint arXiv:2402.15852*.

Junbo Zhang and Kaisheng Ma. 2024. Mg-vln: Benchmarking multi-goal and long-horizon vision-language navigation with language enhanced memory map. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7750–7757. IEEE.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2023. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104*.

Gengze Zhou, Yicong Hong, and Qi Wu. 2024. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7641–7649.

Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. 2023. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR.

A LLM-based Navigation System

Currently, LLM-based navigation methods can be divided into two categories. The first is end-to-end models, such as NaVid (Zhang et al., 2024), which directly convert inputs into control policies for the robot. However, this approach comes with a significant training cost. The second, more mainstream and simpler approach, involves using LLMs as a high-level planner, and a lower-level controller handled by the robot itself. (Long et al., 2024; An et al., 2024; Shah et al., 2023b; Cai et al., 2024; Kaichen et al., 2024). The LLM planner conducts sub-goal planning based on the environment and the instruction to reach the final goal. The controller (Shah et al., 2023c; Sridhar et al., 2024), usually using a small neural network, finds a trajectory from the current place to the sub-goal and controls the robot moving in a real-time manner. As the final goal is usually unseen at the beginning, one navigation task needs multiple navigation steps. To achieve high success rate, existing works (Chen et al., 2024; Long et al., 2024; Shah et al., 2023b; Wang et al., 2024; Kaichen et al., 2024) all use giant models, e.g., GPT-4, as the planner, preventing the navigation system deployed on local devices. While directly switching to smaller LLMs will cause significant success rate drops (Cao et al., 2024; Long et al., 2024; Kim et al., 2024).

As shown in Table 1, EfficientNav is the first work to enable accurate on-device zero-shot VLN. EfficientNav designs a memory selection mechanism to remove redundant information in the navigation map. Hence, help the smaller LLM focus on relevant information and improve success rate.