ENTL: Embodied Navigation Trajectory Learner

Klemen Kotar [1]
Stanford University

Aaron Walsman
University of Washington

Roozbeh Mottaghi
Meta AI

Abstract

We propose Embodied Navigation Trajectory Learner (ENTL), a method for extracting long sequence representations for embodied navigation. Our approach unifies world modeling, localization and imitation learning into a single sequence prediction task. We train our model using vector-quantized predictions of future states conditioned on current states and actions. ENTL's generic architecture enables the sharing of the the spatio-temporal sequence encoder for multiple challenging embodied tasks. We achieve competitive performance on navigation tasks using significantly less data than strong baselines while performing auxiliary tasks such as localization and future frame prediction (a proxy for world modeling). A key property of our approach is that the model is pre-trained without any explicit reward signal, which makes the resulting model generalizable to multiple tasks and environments. We release the code at https://github.com/klemenkotar/ENTL
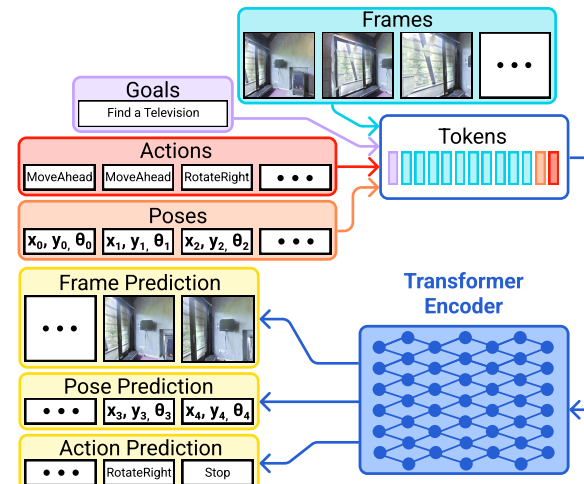
Figure 1. We introduce a method for extracting long sequence representations for embodied navigation. The proposed architecture enables sharing a spatio-temporal transformer-based backbone across multiple tasks: navigation, localization, and future frame prediction.

1. Introduction

Representation learning has shown great success in many AI domains. The approach, common in the NLP field for quite some time, has seen an explosion of popularity through the adaptation of the transformer architecture by work such as BERT [12] and GPT [8]. This initial success has triggered the creation of models with ever more parameters trained on ever larger datasets. Surprisingly, the growing models not only kept improving with the increased parameter counts and datasets, but even started exhibiting emergent behaviors [6]. Furthermore, the general recipe of converting data into sequences of tokens and predicting the next token has proven surprisingly universal. Subsequent work such as ViT [13] and MAE [21] have adapted this approach to the computer vision domain with great success, and transformer models trained on large scale data have become the de facto computer vision backbone.

Many Embodied AI (EAI) tasks have long horizons and sparse reward signals. This means that EAI models must aggregate information across many time steps, and learn how to attribute the successful or unsuccessful outcomes of entire sequences to individual actions. Since the space and time complexity of transformers grows quadratically with sequence length it has been difficult to apply them to these long sequence tasks. Furthermore, it can be difficult to learn which part of a frame will be significant to a decision the model will have to make many steps down the line, making it hard to decide which information to preserve.

Modeling the environment of the embodied agent is a natural self supervised pre-training task. Various abstractions of this problem, such as the prediction of top down maps [48] or graph-based representations [17] have been studied, but none have yielded a strong universal abstraction. More recently, methods such as [36] have explored frame prediction in the RGB space conditioned on camera pose using a buffer of a few frames. These produce visually

---

[1]Work mainly done while author was at Allen Institute for AI

compelling predictions, but fail to capture the movement dynamics of an embodied agent in the scene by conditioning their models with camera pose instead of agent actions. They also rely on a limited bank of previous frames which only allow the model to extrapolate the textures and shapes of the current view of the room.

We use future frame prediction as a universal navigation pre-training task by conditioning it on agent actions. This encodes the navigability of different areas of the environment within the prediction model, and allows us to use agent walkthroughs of the environment as our sole source of supervision, without the requirement of a known camera pose which might not be available in real world settings. Secondly we utilize an architecture that attends over long sequences (up to 50 steps), allowing the agent to synthesize novel views of the environment from many past examples, and obtaining a richer representation of the environment. Finally, we jointly model future frame prediction, agent pose prediction and action prediction, effectively merging imitation learning and representation learning. This approach directly supervises the agent on how to perform a navigation task, and removes the need for a reward signal.

To ensure that crucial information is not forgotten we propose the prediction of future frames in the original RGB space, avoiding any inductive biases that might come from modeling the embeddings of observations. Since predicting the outputs in RGB pixel space can be unstable and difficult, we utilize the image tokens produced by VQ-GAN [15] as our prediction targets. Modern tokenizers such as VQ-GAN can perform near-lossless encodings and reconstructions of RGB images, allowing us to effectively treat them as a reversible transform.

To overcome the long sequence lengths, we utilize a model architecture that attends across space and time in alternating layers similar to [5], allowing the full information of a trajectory to be aggregated without using intractable amounts of memory and computation. We combine this spatio-temporal sequence representation with three separate task heads for localization, future frame prediction and action prediction to maximize parameter sharing and utilize a causal mask to prevent the leakage of information from future states, while still satisfying the input requirements of each head. This design enables us to have a sequence representation that is suitable for all tasks (Figure 1).

We apply the self-supervised offline data training recipe and demonstrate that it can produce a model that performs well on embodied navigation tasks. Our model performs competitively against strong Reinforcement Learning (RL) and Imitation Learning (IL) baselines trained on significantly more data on the PointNav [1] and ObjectNav[4] tasks. Furthermore, our model performs very well at localization, accruing an average error of only 0.43m after a sequence of 50 steps, using RGB frames only and no motion sensors. Finally, our model produces high-quality, realistic future frame predictions, even in scenes with appearance characteristics well outside its training distribution.

In summary, our key contributions are 1) the formulation of a long sequence future frame prediction problem as a self-supervised task for embodied navigation 2) the proposal of a tokenization scheme and model architecture, which allow us to implement this pre-training 3) the contrastive analysis of several adjacent methods and the necessary pieces required to make our approach work.

2. Related Work

Embodied Representations. Various works in the literature have addressed representation learning for/from embodied settings. [31] use physical interactions with objects to learn visual representations. [29] propose a "curiosity"-based approach to learn representations. [14] also propose a curiosity-driven approach that jointly learns the visual representations and the policy. [39] leverage mid-level representations to improve sample efficiency and generalization for embodied tasks. [42] learn representations from agents that play an adversarial game. [32, 46] use egocentric data for visual representation learning, where their goal is to improve the performance of embodied tasks. The focus of these approaches is on *visual* representation learning while our goal is to learn sequences. [3, 45, 22] encode long videos using transformer models. In contrast, we propose an architecture for embodied settings which serves as a backbone for multiple navigation tasks while handling different input constraints of the tasks.

Recent work such as [7, 23, 9] has used transformer-based architectures for interactive settings. We explore applying a vanilla Decision Transformer to our problem and find that predicting a single embedding for future observations is not sufficient. In addition, our approach does not require a reward signal. [35] focus on performing multiple embodied and language-based tasks jointly. In contrast, we propose an architecture to handle long sequences of observation and actions and we also focus on sample efficiency of training, without jointly modeling language. [7] learn representations from robot data in a self-supervised fashion. They do not show results on long-horizon tasks such as navigation. Alternate spatial and temporal layers, vector-quantization of the output space, and handling visually complex environments differentiates our work from [23, 9].

World/Environment Modeling. [37, 25] learn environment models and generate future frames. Their work is conceptually similar to our future frame prediction. [18] learn representations for multi-task RL by predicting the future latent embeddings of the observation. [28] show visual representations can be better than state representations for training control policies. In contrast to these works,

our aim is to learn representations for sequences of actions, observations, and agent poses. World modeling methods such as Dreamer [20] have explored jointly learning a world model and control policy for an interactive agent. We show our method outperforms this approach. Our approach also learns directly from expert trajectories, and does not require a reward signal. Works such as [36] predict future frames in RGB space with high success, but only rely on a few buffer frames and are therefore not capable of modeling an entire environment simultaneously or keeping a memory of previously explored locations, thus they are not well suited as a training objective for embodied sequence learning. [40] learn to predict object detections at future states in embodied environments. In contrast we add no inductive bias to our predictor and model the RGB frames of our environment, eliminating the need for ground truth object labels which might not always be available or reliable. [16] use a transformer to embed observations of a scene formed during exploration and access them when making policy decisions. In contrast we train the model to predict future frames based off past observations, forcing the model to synthesize new views and learn the spatial structure of environments. [33] model embodied navigation trajectories using a transformer. They introduce the inductive bias of splitting their trajectory into zones and predict what zone they are in. Our method simply models action conditioned future frames without any assumptions about the structure of the environment.

Representations for Navigation There are several works addressing representation learning specifically for navigation tasks. For example, [44] learn visual representations from 2.5 billions of interactions in a point-goal navigation setting. [47] improve sample efficiency of navigation models by combining representations learned from auxiliary tasks. [24] use CLIP representations and show large improvements across a set of embodied tasks. In contrast, we have a generic backbone for multiple tasks.

3. Trajectory Learning

Task definition. Our goal in this paper is to learn representations for navigation trajectories (i.e., sequence of images, actions, and poses) of embodied agents. Given a set of trajectories $\mathcal{T}$ of agents navigating in environments, we train a spatio-temporal transformer model to encode these sequences. Each trajectory $\tau \in \mathcal{T}$ is a sequence $(I_i, p_i, a_i)_{i=0}^{N}$, where I, p, and a represent image, agent pose, and action, respectively and are collectively called a state. We represent the pose p as (x, y, θ), the position and orientation of the agent. The objective is to predict the future frame, the action, and the pose of the agent (in a vector-quantized token space described in Section 3.2).

The challenges are: (1) how to encode long sequences of embodied tasks, (2) how to represent observations, actions, and poses so they can be processed by a single transformer

model, (3) and how to train these models without any explicit reward.

Our architecture enables sharing the backbone across four tasks: future frame prediction, agent localization, object-goal navigation (ObjectNav) [4], and point-goal navigation (PointNav) [1]. Our approach relies on treating all of the observations our agent collects and the actions it takes as part of the same long contiguous sequence. Deciding which action our agent should take next is achieved by predicting the next action token in the sequence, and localization is achieved by predicting the pose tokens. Future frame prediction can be achieved by appending the desired action token to our current sequence and predicting the subsequent frame tokens. This approach unifies our training into a simple task of completing a sequence and yields a powerful joint representation that allows us to reason over the timescale of entire task sequences. The model has the potential to generalize to other tasks if their input/output and goals can be specified by the same tokenized representation.

3.1. Data Collection

Training a self-supervised transformer model requires a large amount of trajectory data. In this work, we use off-policy data to bootstrap our model but preliminary tests show that collecting the on-policy data can reduce sample complexity even further and speed up convergence, at the expense of running a simulator.

Collecting suitable data is not trivial, as our agent needs to see demonstrations of complex behavior. For example, the ObjectNav tasks consists of several implicit sub-tasks, namely exploring the environment, identifying an object of the goal category, and walking towards it.

One solution for this is to collect expert human demonstrations, which inherently contain such complex behavior. We utilize this type of data for the Habitat [38] environment provided by [34].

Another solution is to collect demonstrations using a classical planner with access to full scene information. Such omniscient experts do not inherently produce high quality trajectories as they know exactly how to get to the target and thus exhibit no exploratory behavior. We utilize this approach for collecting trajectories in the AI2-THOR [26] environment as human demonstrations are not available.

3.2. Tokenization

To fully exploit the "predict the next token" training recipe we must tokenize our trajectory data into sequences of discrete integers. In NLP this is trivial, as each word can intuitively map to an integer (although in practice more complicated schemes are used). In computer vision, images are often quantized by discrete variational autoencoders, which encode an image as a sequences of several integers, and are capable of reconstructing the image from the se-
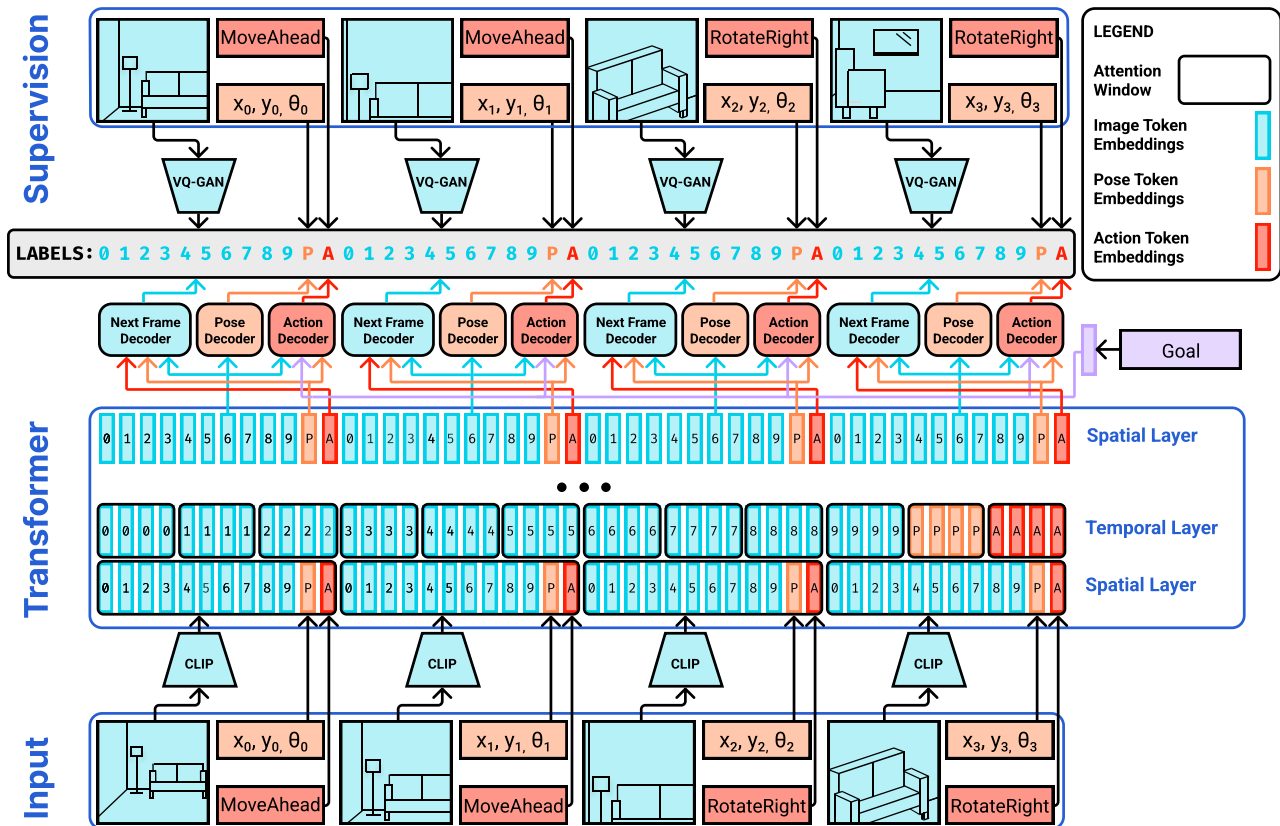
Figure 2. **ENTL Model Architecture.** Sequences of frames, poses and actions are passed through a spatio-temporal transformer encoder, which aggregates information across space and time in alternating layers to model an entire sequence. Poses and actions are discretized and passed through a learned embedding, while frames are converted to CLIP embeddings. Spatial layers pay attention to all the tokens in a frame, while temporal layers pay attention to the n-th token of every frame across an entire sequence. In addition to this, a full diagonal causal mask is applied to the attention matrix. After that the representations are passed through three separate transformer heads, which produce the next frame prediction, the current pose of the agent, and the action the agent will take next. The frame self-supervised labels are provided in the form of vector-quantized tokens, while the labels for the actions and pose are just the discretized ground truth values.

quence. Because of this, they act as slightly lossy reversible transforms, and they allow us to model a proxy for the distribution of action-conditioned future observations, by modeling the distribution over a sequence of just a few future observation tokens. We utilize the following tokenization schemes to convert our embodied trajectories into sequences of tokens:

Image Tokenization is performed by an off-the-shelf discrete variational autoencoder, namely the VQ-GAN model [15]. While tokenizers that quantize images using 32x32 tokens produce more faithful reconstructions, we choose to use a variety that produces only 16x16 tokens for each image, drastically reducing the computation requirements per frame, allowing us to train with longer sequences and larger models. For our experiments we use the encoder with a vocabulary size of 16,384 but in practice we only saw a marginal visual reconstruction benefits over using a vocabulary size of 1024.

Pose Tokenization is performed by discretizing position in

the x-axis, position in the y-axis, and rotation into 3 pose tokens. The absolute x and y positions of the agent are binned to the nearest 0.1m box in the range of -100m to +100m for a total vocabulary size of 2000. The absolute rotation of the agent is binned to the nearest 3.6 degrees in the range of 0 to 356.4. Using a relative positioning scheme teaches the model to compute positional offsets from the initial frame instead of some absolute origin which might be unknown in an evaluation environment.

Action Tokenization is performed by mapping the following actions to integers: *MoveAhead*, *MoveBack*, *RotateLeft*, *RotateRight*, *LookUp*, and *LookDown* and *Stop* for a total vocabulary size of 7. The translation step is 0.25m and the rotation step is 30 degrees. The action set is the same between the AI2-THOR [26] and Habitat [38] environments used for the experiments, while the physical agent dimensions and camera parameters differ.

Goal Tokenization is performed by encoding either the relative offset of a target location (for point-goal navigation

task) using two positional tokens or an object category (for object-goal navigation) using a single target category token.

3.3. Architecture

We propose a model architecture designed to address two crucial challenges of applying our approach to Embodied AI: long sequences and the input constraints of the various prediction heads.

Spatio-Temporal Backbone. Embodied tasks contain sequences of several hundred frames. Each one of our frames contains 256 image tokens, 3 pose tokens, and 1 action token, therefore it is not computationally practical to concatenate them into a single long sequence. Instead we attend across space (by which we mean across tokens within one frame) and across time (by which we mean across tokens that occupy the same position in different frames) in alternating layers as in [5], see Figure 2 for details. This mechanism allows us to keep our sequence lengths low while still aggregating information across all the tokens in a trajectory. We achieve this by simply reshaping the output after every layer. We utilize a GPT-style encoder-only transformer model with 8 spatial and 7 temporal layers and an embedding dimension of 512.

Separate Decoder Heads. The second constraint of our problem space is that the decoders can not share all information. For instance, the action head is predicting the action the agent should take next to achieve some goal, while the frame prediction head needs to take this very same action as an input to be able to predict the next frame. Because of this dependency structure, we must take care to not leak information from the future. To that end we apply a diagonal causal mask over the spatio-temporal encoder and decode the pose, action and future frame predictions using three separate decoders for each frame. The decoders are 4 layer transformers with an embedding dimension of 512 and no masking. They each take a different set of tokens as inputs and produce their corresponding predictions as illustrated in Figure 2. This architecture allow us to share the vast majority of the parameters across all tasks, as all the decoders utilize the embeddings produced by the spatio-temporal encoder and pass their gradients through it during training.

CLIP Embeddings. Our empirical experimentation has shown that the tokens produced by VQ-GAN are not easily separable by semantic concepts such as the categories of objects the image contains. Because of this, image tokens are not ideal inputs for our model, as they do not give us the full representational power of traditional image encoders, despite being trained on large datasets. To overcome this, we feed our model CLIP embeddings instead of image tokens and simply predict the VQ-GAN tokens of the next frame. A similar strategy is employed by BEiT [30] to learn a strong image encoder.

3.4. Loss Function

We optimize our model to jointly predict the pose of the agent, the action it should take towards achieving a specified goal, and the image observation that the agent will obtain if it takes a specified action. We hypothesize that future frame prediction will train our model to synthesize information over a long sequence, since it relies on remembering the visual characteristics of parts the room that our agent has seen a long time ago, as well as understand how specific actions influence the observations that the agent will receive (e.g., moving forward makes objects ahead larger). The pose prediction will help orient our agent and measure the space it moves through. The action prediction produces successful navigation policies for moving across the environment, which is the main goal of our work. Our loss has the following three components, each of which is calculated as the cross entropy between the predicted distribution and the target tokens.

Frame Loss is applied to the future frame prediction head, which takes as input the embedding of the current frame, the current pose, and the action our agent will take next and outputs the distribution over the vector-quantized image tokens of the next frame.

Pose Loss is applied to the pose prediction head, which takes as input the embedding of the current frame and outputs the distribution over the pose tokens of the agent at this time step.

Action Loss is applied to the action prediction head, which takes as input the embeddings of the current frame, the current pose, and a goal and outputs the distribution over actions that the agent should take to achieve the specified goal.

Initial experiments suggested that learning future frame prediction was most important for developing strong trajectory representations, but we chose to model the pose and actions of the agent as well to unify all three tasks into a single sequence prediction problem.

3.5. End Tasks

We describe our end tasks more concretely, below. In addition to being good benchmarks of progress, these tasks can be combined to achieve complex, multi-stage behavior.

Localization: Given a sequence of past frames, agent poses and actions as well as the current frame, determine the position and orientation of the agent.

Future Frame Prediction: Given a sequence of frames, agent poses and actions, predict the next frame conditioned on the next action.

Navigation: Given a target positional offset from the initial location or a target object as well as all past observations and actions, navigate an agent across a building environment to the target. If the target is a position (PointNav), the task is successful if the agent navigates to the target point

and selects the stop action within 0.2m of the specified position. If the target is an object category (ObjectNav), the agent navigates to any instance of the target category and selects the stop action within 1.5m of the object.

3.6. Data Augmentation

Future Frame Masking. One interesting detail we encountered was the importance of masking out the entire future frame during training. We attempted to utilize a fully diagonal masking strategy where each token could attend to all the tokens to the left, and so consequentially most tokens of a frame I_{n+1} could attend to all the tokens of I_n as well as some of the tokens of I_{n+1}. In the limited experiments we attempted to train our model with this masking strategy, it learned to reconstruct I_{n+1} from the partial information about the frame given to it, instead of learning how taking the action a_n transforms frame I_n into frame I_{n+1}.

Hindsight Trajectory Superposition. There is a major imbalance between the frequency of the 'stop' action (when the agent reaches the goal) and all the other actions, as the agent stops only once in each episode. To ensure that our model learns to recognize goal conditions, we modify our action labels such that every state, which is a valid goal condition for any ObjectNav category, is marked as 'stop'. We also select a number of states roughly equivalent to the number of ObjectNav goal states and assign them as PointNav goal states, and their corresponding action labels are set to stop. In addition, we modify the goal tokens of all other states to line up with the goal of the next stop state. We can achieve this type of hindsight trajectory modification, because we only modify the goal tokens and action labels, neither of which affect the state of the spatio-temporal encoder. This allows us to artificially superimpose several tasks over a single trajectory during training so we call it *Hindsight Trajectory Superposition (HTP)*, (inspired by Hindsight Experience Replay [2] which shares conceptual similarities).

4. Experiments

We evaluate ENTL on four tasks: PointNav, ObjectNav, localization, and future frame prediction. We compare our method to some strong, popular baselines. We also provide results of ablation experiments that profile the importance of the future frame prediction loss and show scaling laws. Note that our method only uses RGB images (no depth image or motion sensors).

4.1. Data Collection

Environments. We train and evaluate our approach in two interactive embodied AI simulators: AI2-THOR [26] and Habitat [38]. AI2-THOR simulates houses populated with digital assets that model objects and furniture in a house. Habitat utilizes indoor 3D scans of real world buildings to simulate living and working environments. Each simulator

has different visual and behavioral characteristics, and we show that our method can be applied to both. Across both, our agent can move forward in 0.25m increments, rotate left or right in place by 30 degree increments, tilt its camera up or down by 30 degree increments and signal when it thinks the task has been complete with a 'stop' action.

In the Habitat environment, we train our model on 8M frames collected by human annotators as part of Habitat-Web data [34]. These are high-quality trajectories because they capture the human intuition involved in exploring an environment, identifying an object as an instance of the goal category, and navigating towards it.

In the AI2-THOR environment, we do not have access to a large dataset of human-collected trajectories, but we can obtain expert trajectories from a large collection of procedurally generated scenes by using ProcTHOR [11] and an omniscient expert planner. While these trajectories do not contain the exploratory behavior that humans exhibit when interacting with a new environment, they do contain a greater diversity of floorplans that the agent navigates across. We collected a dataset of 20M ProcTHOR frames, consisting of trajectories of no less than 100 steps (frames) each, in which the agent visits every target object in each of 10k train scenes in random order.

4.2. Training

We train a model on ProcTHOR data, and a separate model on Habitat data for efficiency (combining the data from these environments did not improve the results significantly). The models have the architecture with 8 spatial and 7 temporal layers in the encoder and 4 layers in each of three decoders. Each layer has 8 heads, 512 embedding dimensions and a MLP factor of 4. The entire ENTL model has 96M parameters. We train the model at half precision with a batch size of 8 for 500K steps using the AdamW optimizer [27] and a learning rate of $1e - 4$.

4.3. Navigation

We evaluate the performance of our model on the Point-Nav [1] and ObjectNav [4] tasks.

PointNav. We train a version of ENTL on just 1M frames for the PointNav [1] task. Remarkably, we find that our method is able to achieve performance on par with a PPO baseline method that used 50x more data (Table 1). Additionally, the baseline uses depth as input, which is known to work better for PointNav, while our model uses RGB only. Our model also has the benefit of producing auxiliary outputs which can be used for future frame position and localization, even when trained on such a small dataset. This shows that ENTL is capable of constructing world models and learning to solve simple tasks using a very small amount of samples. Since the encoder of the PPO baseline is not based on CLIP, we also test a version of ENTL with-

out the CLIP encoder using the VQ-GAN tokens as inputs and targets instead (referred to as ENTL RGB) and find that it does not make a large difference. Finally, we evaluate a CLIP+LSTM imitation learning baseline (IL RGB+CLIP) trained on the same data as ENTL and show that it works significantly worse than our method.

Model	Environment	# Frames	Success	SPL
DD-PPO Depth [43]	RoboTHOR	50M	0.93	0.88
IL RGB-CLIP	RoboTHOR	1M	0.61	0.52
ENTL RGB (Ours)	RoboTHOR	1M	0.92	0.87
ENTL RGB+CLIP (Ours)	RoboTHOR	1M	0.95	0.91

Table 1. **Point-goal Navigation Results.** We use AllenAct [43] implementation of DD-PPO [44] as the baseline.

ObjectNav. The model trained on ProcTHOR data is evaluated on RoboTHOR [10] val scenes, showcasing the ability of our model to adapt to visual domains outside of the training data. Our approach achieves competitive task success compared to EmbCLIP [24] trained on ProcTHOR [11] data, which is a state-of-the-art model trained with an order of magnitude more data (typically EmbCLIP success rates has 2-3% variance). Table 2 shows the results.

We also evaluate a version of our model trained on the Habitat MP3D Val split consisting of unseen scenes. The model achieves strong performance when trained on 2.5x less data than the imitation learning baseline using Habitat-Web [34] data (referred to as 'IL w/ Hab.-W'). The reason we chose this baseline is that it uses the same type of data as our model. Note that the architecture is identical for both navigation tasks, but we train the model for each task separately. We chose this subset of data since it allowed us to achieve a similar SPL compared to the baseline, while requiring a much lower sample complexity.

Sequence Learning Baselines. We analyze the performance of two popular sequence learning baselines trained on the same data as ENTL. We implement a vanilla version of Decision Transformer [9] with a CLIP encoder (for fair comparison), where we embed our current observations into a single token, then predict future observations and rewards. We add an additional l2 loss to model future observation tokens based on current observations and actions. This model performs similarly to our "no future frame prediction" baseline, showcasing the importance of our specific type of future frame prediction loss. We also implemented a version of Dreamer [19]. We found that this baseline was mostly successful at nearby targets. See Table 2 for the results. We hypothesize that learning to predict future states in the embedding space is not sufficient to obtain a strong understanding of the environment.

4.4. Localization

An ENTL agent receives a tokenized representation of its current and past pose as input, and predicts the pose in

Model	Environment	# Frames	Success	SPL
EmbCLIP [11]	RoboTHOR	200M	0.52	0.22
ENTL (Ours)	RoboTHOR	20M	0.46	0.16
Dreamer [19]	RoboTHOR	20M	0.31	0.06
Decision Transformer [9]	RoboTHOR	20M	0.40	0.11
IL w/ Hab.-W [34]	Habitat	22M	0.23	0.06
ENTL (Ours)	Habitat	8M	0.17	0.05

Table 2. **Object-goal Navigation Results.** EmbCLIP [24] is trained on trajctories from ProcTHOR [11]. IL w/Hab.-W is the imitation learning model from [34] that, similar to our model, uses only RGB images.

the next state. Because of this it naturally performs the task of localization. We evaluate the ability of our agent to effectively localize over a long trajectory by predicting the pose tokens for a sequence of 50 steps. We observe a final mean localization error of 0.43m and a std of 0.59m on a dataset of 10,000 RoboTHOR trajectories and an error of 1.12m and std of 1.52m on a dataset of 10,000 Habitat trajectories. A naive baseline that simply estimates the agent pose based on the actions achieves errors of 0.93m and 2.34m on RoboTHOR and Habitat, respectively. The high standard deviations suggest that for some trajectories our model performs very well and for a few it has a high error. Note that movements in these environments are stochastic, and our model does not use any depth or motion sensor.
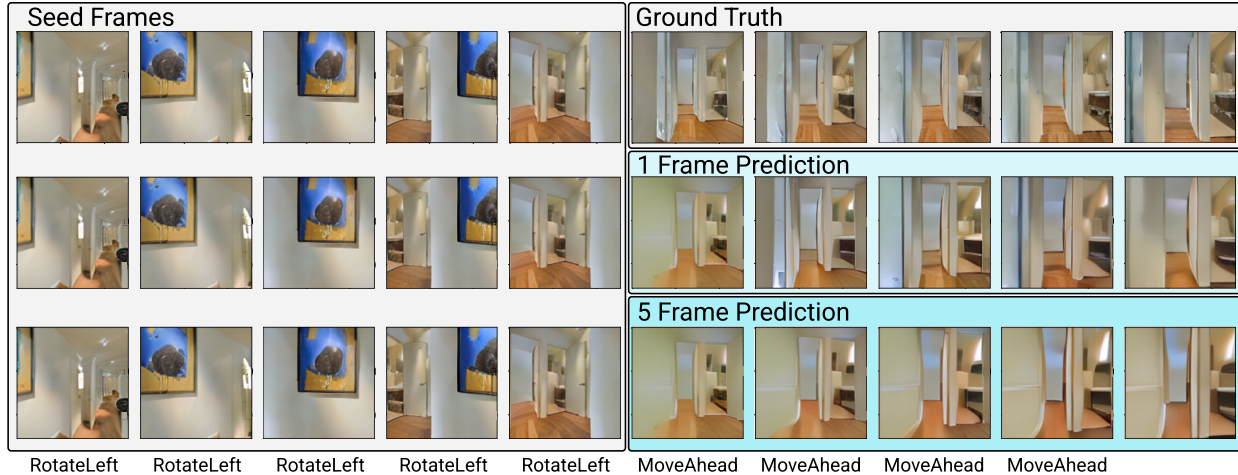
4.5. Future Frame Prediction

While future frame prediction is not the main focus of our work, the nature of our loss function optimizes this objective as well. To test the quality of future frame prediction, we compute the SSIM score [41] of future frame predictions on unseen scenes, and compare it to a naive baseline of simply repeating the last frame. We achieve a SSIM score of 0.361 on 10k Habitat trajectories (compared to the baseline of 0.219) and 0.578 on 10k RoboTHOR trajectories (compared to the baseline of 0.396). Figure 3 shows generation of 1 and 5 frames into the future. For instance, in the Habitat example, the model learns to predict larger objects as the agent moves forward.

4.6. Ablations

Impact of Various Objectives To profile the contribution of the individual objectives during pre-training, we test a version of the ENTL model where the frame loss was set to 0. Without this loss component the model performance drops significantly, highlighting the importance of future frame prediction to the extraction of useful sequence representations. We also profile the effect of omitting the pose prediction loss and find it has negligible impact on performance. Finally we profile the impact of omitting HST label modification and find it has a significant impact. We run these experiments using the smaller 10M model. Refer to
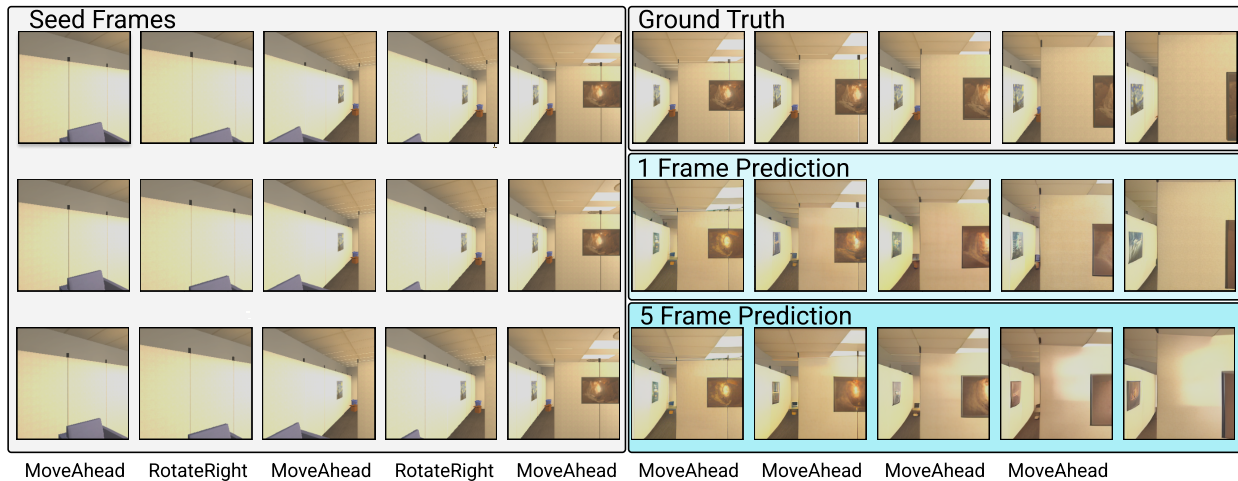
Figure 3. **Predictions of future frames in unseen scenes.** For each rollout 5 seed frames are given to the model, then it predicts the next 5 steps with and without re-seeding after each step, based on the provided actions. Additional examples in appendix 4.

Table 3 for the results.

Model	# Frames	Success	SPL
w/o Frame Pred. Loss	20M	0.21	0.03
w/o HST	20M	0.28	0.04
w/o Pose Pred. Loss	20M	0.34	0.06
w/ Frame Pred. Loss	20M	0.35	0.06

Table 3. **Frame Prediction Loss.**

Scaling Laws. We analyze the effects of model size on final performance. We find that increasing model size monotonically improves performance on the RoboTHOR ObjectNav task. We train a version of our model using 10M and 50M parameters, respectively, using the same dataset and training schedule. Refer to Table 4 for the results.

Model	# Success	SPL
ENTL-10M	0.35	0.06
ENTL-50M	0.42	0.13
ENTL-100M	0.46	0.16

Table 4. **Scaling Laws.**

5. Conclusion

We introduced ENTL: Embodied Navigation Trajectory Learner, a new approach for sequence representation learning for embodied navigation tasks. We presented a strategy for converting embodied trajectories into sequences of discrete tokens, and demonstrated that it is an effective approach in practice. We present a model architecture that effectively synthesized information across both space and time, and trained various instance of that model on canonical embodied navigation tasks. We compare our approach to baseline methods to demonstrate its advantages. Our approach demonstrated significant gains in sample complexity, as well as producing useful auxiliary outputs. For example, our model is able to visualize future frames which can be useful for explainability or planning, and localize the agent. Our architecture does not make assumptions about the specific tasks we studied, so it is generic and can be used for other embodied tasks.

References

[1] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *arXiv*, 2018. 2, 3, 6

[2] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Joshua Tobin, P. Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017. 6

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, 2021. 2

[4] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv*, 2020. 2, 3, 6

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, 2021. 2, 5

[6] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv*, 2021. 1

[7] Rogerio Bonatti, Sai Vemprala, Shuang Ma, Felipe Vieira Frujeri, Shuhang Chen, and Ashish Kapoor. Pact: Perception-action causal transformer for autoregressive robotics pre-training. *arXiv*, 2022. 2

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 1

[9] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021. 2, 7

[10] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. Robothor: An open simulation-to-real embodied ai platform. In *CVPR*, 2020. 7

[11] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. In *NeurIPS*, 2022. 6, 7

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 1

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is

worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1

[14] Yilun Du, Chuang Gan, and Phillip Isola. Curious representation learning for embodied intelligence. In *ICCV*, 2021. 2

[15] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 2, 4

[16] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *CVPR*, 2019. 3

[17] Samir Yitzhak Gadre, Kiana Ehsani, Shuran Song, and Roozbeh Mottaghi. Continuous scene representations for embodied ai. In *CVPR*, 2022. 1

[18] Zhaohan Daniel Guo, Bernardo Ávila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *ICML*, 2020. 2

[19] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. 2020. 7

[20] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *ICLR*, 2021. 3

[21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1

[22] Md Mohaiminul Islam and Gedas Bertasius. Long movie clip classification with state-space video models. In *ECCV*, 2022. 2

[23] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*, 2021. 2

[24] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *CVPR*, 2022. 3, 7, 11

[25] Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Pathdreamer: A world model for indoor navigation. In *ICCV*, 2021. 2

[26] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Kumar Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv*, 2017. 3, 4, 6

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[28] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *ICML*, 2022. 2

[29] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 2

[30] Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. Beit v2: Masked image modeling with vector-quantized visual tokenizers. *arXiv*, 2022. 5

[31] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Kumar Gupta. The curious robot: Learning

visual representations via physical interactions. In *ECCV*, 2016. 2

[32] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *CoRL*, 2022. 2

[33] Santhosh Kumar Ramakrishnan, Tushar Nagarajan, Ziad Al-Halah, and Kristen Grauman. Environment predictive coding for visual navigation. In *ICLR*, 2021. 3

[34] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *CVPR*, 2022. 3, 6, 7

[35] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley D. Edwards, Nicolas Manfred Otto Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *arXiv*, 2022. 2

[36] Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In *CVPR*, 2022. 1, 3

[37] Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In *CVPR*, 2022. 2

[38] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 3, 4, 6

[39] Alexander Sax, Jeffrey O. Zhang, Bradley Emi, Amir Roshan Zamir, Silvio Savarese, Leonidas J. Guibas, and Jitendra Malik. Learning to navigate using mid-level visual priors. In *CoRL*, 2019. 2

[40] Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav S. Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion. *arXiv*, 2021. 3

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004. 7

[42] Luca Weihs, Aniruddha Kembhavi, Winson Han, Alvaro Herrasti, Eric Kolve, Dustin Schwenk, Roozbeh Mottaghi, and Ali Farhadi. Learning generalizable visual representations via interactive gameplay. In *ICLR*, 2021. 2

[43] Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. Allenact: A framework for embodied ai research. *arXiv*, 2020. 7

[44] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2019. 3, 7

[45] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *CVPR*, 2022. 2

[46] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv*, 2022. 2, 11

[47] Joel Ye, Dhruv Batra, Erik Wijmans, and Abhishek Das. Auxiliary tasks speed up learning pointgoal navigation. In *CoRL*, 2020. 3

[48] Minzhao Zhu, Binglei Zhao, and Tao Kong. Navigating to objects in unseen environments by distance prediction. In *IROS*, 2022. 1