Abstract
Language-guided robots performing home and office tasks must navigate in and interact with the world. Grounding language instructions against visual observations and actions to take in an environment is an open challenge. We present Embodied BERT (EmBERT), a transformer-based model which can attend to high-dimensional, multi-modal inputs across long temporal horizons for language-conditioned task completion. Additionally, we bridge the gap between successful object-centric navigation models used for non-interactive agents and the language-guided visual task completion benchmark, ALFRED, by introducing object navigation targets for EmBERT training. EmBERT achieves competitive performance on the ALFRED benchmark, and is the first model to use a full, pretrained BERT stack while handling the long-horizon, dense, multi-modal histories of ALFRED. Model code is available at the following link: https://github.com/amazon-research/embert

1 Introduction
Language is grounded in agent experience based on interactions with the world (Bisk et al., 2020; Bender and Koller, 2020). Task-oriented, instructional language focuses on objects and interactions between objects and actors, as seen in instructional datasets (Damen et al., 2020; Koupaee and Wang, 2018), as a function of the inextricable relationship between language and objects (Quine, 1960). That focus yields language descriptions of object targets for manipulation such as put the strawberries on the cutting board and slice them into pieces (Chai et al., 2018). We demonstrate that predicting navigational object landmarks in addition to manipulation object targets improves the performance of an instruction following agent in a rich, 3D simulated home environment. We posit that object-centric navigation is a key piece of semantic and topological navigation (Kuipers and Byun, 1991) for Embodied AI (EAI) agents generally.

Substantial modeling (Majumdar et al., 2020) and benchmark (Qi et al., 2020b) efforts in EAI navigation focus on identifying object landmarks (Blukis et al., 2018) and destinations (Batra et al., 2020b). However, for agent task completion, where agents must navigate an environment and manipulate objects towards a specified goal (Gordon et al., 2017; Shridhar et al., 2020), most predict movement actions without explicitly identifying navigation object targets (Singh et al., 2020; Pashevich et al., 2021; Nguyen et al., 2021; Abramson et al., 2020). We address this gap, grounding navigation instructions like Head to the sink in the corner by predicting the spatial locations of the goal sink object at each timestep (Figure 1).

Transformer-based models in EAI score the alignment between a language instruction and an already-completed path (Majumdar et al., 2020) or introduce recurrence by propagating part of the hidden state to the next timestep (Hong et al., 2020). The former requires beam search over sequences of environment actions, which is not feasible when actions cannot be undone, such as slicing an apple. The latter introduces a heavy memory requirement, and is feasible only with short trajectories of four to six steps. We overcome both limitations by decoupling the embedding of language and visual features from the prediction of what action to take next in the environment. We first embed language and visual observations at single timesteps using a multi-modal transformer architecture, then train a transformer decoder model to consume sequences of such embeddings to decode actions (Figure 3).

We introduce Embodied BERT (EmBERT), which implements these two key insights:

1. Object-centric Navigation unifies the disjoint navigation and interaction action se-
sequences in ALFRED, giving navigation actions per-step object landmarks.

2. Decoupled Multimodal Transformers enable extending transformer based multimodal embeddings and sequence-to-sequence prediction to the fifty average steps present in ALFRED trajectories.

2. Related Work

Natural language guidance of robots (Tellex et al., 2020) has been explored in contexts from furniture assembly (Tellex et al., 2011) to quadcopter flight control (Blukis et al., 2019).

Embodied AI. For task completion benchmarks, actions like pickup must be coupled with object targets in the visual world, with specification ranging from mask prediction only (Shridhar et al., 2020) to proposals for full low level gripper control (Batra et al., 2020a). Similarly, navigation benchmarks incorporate objects as targets in tasks like object navigation (Qi et al., 2020b; Batra et al., 2020b; Kurenkov et al., 2020), and explicitly modeling those objects assists generally at navigation success (Shrivastava et al., 2021; Qi et al., 2020a, 2021). Many successful modeling approaches for navigation benchmarks incorporate multimodal transformer models that require large memory from recurrence (Hong et al., 2020), beam search over potential action sequences (Majumdar et al., 2020), or shallow layers without large-scale pretraining to encode long histories (Pashevich et al., 2021; Magassouba et al., 2021). In this work, we incorporate navigation object targets into the ALFRED task completion benchmark (Shridhar et al., 2020), and decouple transformer-based multimodal state embedding from transformer-based translation of state embeddings to action and object target predictions. In addition, differently from other approaches that train from scratch their language encoder, we successfully exploit the BERT stack in our multi-modal architecture. In this way, EmBERT can be applied to other language-guided tasks such as VLN and Cooperative Vision-and-DIALOG Navigation (Thomason et al., 2019).

Language-Guided Task Completion. Table 1 summarizes how EmBERT compares to current ALFRED modeling approaches. ALFRED language instructions are given as both a single high level goal and a sequence of step-by-step instructions (Figure 2). At each timestep, we encode the goal instruction and a predicted current step-by-step instruction. We train EmBERT to predict when to advance to the next instruction, a technique introduced by LWIT (Nguyen et al., 2021).

EmBERT uses a panoramic view space to see all around the agent. Rather than processing dense, single vector representations (Shridhar et al., 2020; Singh et al., 2020; Pashevich et al., 2021; Kim et al., 2021; Blukis et al., 2021), EmBERT attends directly over object bounding box predictions embedded with their spatial relations to the agent, inspired by LWIT (Nguyen et al., 2021) and a recurrent VLN BERT model (Hong et al., 2020). We similarly follow prior work (Singh et al., 2020; Pashevich et al., 2021; Nguyen et al., 2021; Kim et al., 2021; Zhang and Chai, 2021) in predicting these bounding boxes as object targets for actions like Pickup, rather than directly predicting a dense object segmentation mask (Shridhar et al., 2020).

Consider the step heat the mug of water in the microwave, where the visual observation before...
The ALFRED benchmark (Shridhar et al., 2020)

Table 1: Model comparison. EmBERT uses a multimodal transformer (TF) to embed language instructions and detected objects in a panoramic view, and a transformer decoder to produce action and object predictions. Ours is the first ALFRED model to add object prediction to navigation steps. Other methods maintain history by taking previous transformer states (TF) (Pashevich et al., 2021), subgoal prediction structures (SG) (Zhang and Chai, 2021), or maintained voxel maps (Blukis et al., 2021), or maintained voxel maps (Blukis et al., 2021) as input.

3 The ALFRED Benchmark

The ALFRED benchmark (Shridhar et al., 2020) pairs household task demonstrations with written English instructions in 3D simulated rooms (Kolve et al., 2017). ALFRED tasks are from seven categories: PICK & PLACE, STACK & PLACE, PICK TWO & PLACE, CLEAN & PLACE, HEAT & PLACE, COOL & PLACE, and EXAMINE LIGHT. Each task involves one or more objects that need to be manipulated, for example an apple, and a final receptacle on which they should come to rest, for example a plate. Many tasks involve intermediate state changes, for example HEAT & PLACE requires cooking the target object in a microwave.

Supervision Data. Each ALFRED episode comprises an initial state for a simulated room, language instructions, planning goals, and an expert demonstration trajectory. The language instructions are given as a high-level goal instruction $I_g$, for example Put a cooked egg in the sink, together with a sequence of step-by-step instructions $\tilde{I}$, for example Turn right and go to the sink, Pick up the egg on the counter to the right of the sink, ... The planning goals $P$ (or subgoals) are tuples of goals and arguments, such as (SliceObject, Apple) that unpack to low-level sequences of actions like picking up a knife, performing a slice action on an apple, and putting the knife down on a countertop. The expert demonstration trajectory $T$ is a sequence of action and object mask pairs, where $T_j = (a_j, M_j)$. Each step-by-step instruction $\tilde{I}_i$ corresponds to a sub-sequence of the expert demonstration, $T_{j:k}$ given by alignment lookup $m_a(i) = (j, k)$ and to a planning goal $P_j$ by alignment lookup $m_p(i) = b$. For example, in Figure 2, instruction $\tilde{I}_0$ corresponds to a GotoLocation navigation goal, as well as a sequence of turning and movement API actions that a model must predict.

Model Observations. At the beginning of each episode in timestep $t = 0$, an ALFRED agent receives the high-level and step-by-step language instructions $I_g, \tilde{I}$. At every timestep $t$, the agent receives a 2D, RGB visual observation representing the front-facing agent camera view, $V^F$. ALFRED models produce an action $a_t$ from among 5 navigation (e.g., Turn Left, Move Forward, Look Up) and 7 manipulation actions (e.g., Pickup, ToggleOn, Slice), as well as an object mask.
M_t. Predicted action $a_t$ and mask $M_t$ are executed in the ALFRED environment to yield the next visual observation. For navigation actions, prediction $M_t$ is ignored, and there is no training supervision for objects associated with navigation actions.

**EmBERT Predictions.** EmBERT gathers additional visual data (Figure 2). After every navigation action, we turn the agent in place to obtain left, backwards, and right visual frames $V^L, V^B, V^R$. Following prior work (Singh et al., 2020), we run a pretrained Mask-RCNN (He et al., 2017) model to extract bounding boxes from our visual observations at each view. We train EmBERT to select the bounding box which has the highest intersection-over-union with $M_t$ (more details in Section 4).

We define a navigation object target for navigation actions. For navigation actions taken during language instruction $I_t$, we examine the frame $V^F_{t:k}$ at time $k$ for $T_k$; $m_a(i) = (j, k)$. We identify the object instance $O$ of the class specified in the planning goal $P_{m_a(i)}$ in $V^F_{t:k}$. We define this object $O$ as the navigation object target for all navigation actions in $T_j$ by pairing those actions with object mask $M^O$ to be predicted during training. We also add a training objective to predict the parent receptacle $P(O)$ of $O$. Parent prediction enables navigating to landmarks such as the table for instructions like Turn around and head to the box on the table, where the box compared to the table on which it rests (Figure 2).

**4 Embodied BERT**

EmBERT uses a transformer encoder for jointly embedding language and visual tokens and a transformer decoder for long-horizon planning and object-centric navigation predictions (Figure 3).

**4.1 Multimodal encoder**

We use OSCAR (Li et al., 2020) as a backbone transformer module to fuse language and visual features at each ALFRED trajectory step. We obtain subword tokens for the goal instruction $I_g = \{g_1, g_2, \ldots, g_n\}$ and the step-by-step instruction $I_j = \{i_1, i_2, \ldots, i_m\}$ using the WordPiece tokenizer (Wu et al., 2016) and process the sequence as: $[CLS] I_g [SEP] I_j [SEP]$, using token type ids to distinguish the goal and step instructions. We derive token embeddings $L \in R^{(m+n+3) \times d_e}$ using the BERT (Devlin et al., 2019) embedding layer, where $d_e$ is the embedding dimensionality.

We provide EmBERT with object-centric representations by using MaskRCNN (He et al., 2017) features to represent detected objects in every frame of the panorama view. We freeze the weights of a MaskRCNN model fine-tuned for AI2-THOR frames (Singh et al., 2020). We fix the number of object detections in the front view $V^F$ to 36, while limiting those in the side views to 18. We represent each object $o \in O$ as an embedding $o \in R^{d_o}$, which is a concatenation of: 1) detection ResNet (He et al., 2016) features; 2) bounding box coordinates; 3) bounding box relative area; and
4) vertical and horizontal heading of the object related to the current agent position, following prior work (Storks et al., 2021). These representations make up the observed object embeddings O. We use a one layer MLP to map object embeddings of dimensionality $d_o$ to size $d_e$.\footnote{In our experiments, in order to reuse the visual embedding available in the OSCAR checkpoint, we use an additional one layer MLP to adapt our visual features to the visual embeddings space learned by OSCAR.} The multi-modal transformer backbone consumes the token and object embeddings to produce multi-modal hidden states $H \in \mathbb{R}^{m+n+|O| \times d_e}$. We obtain these state representations, $h_t$, for each timestep $t$ by computing an element-wise product between $H_0$ and $H_{m+n}$, the hidden state of the [CLS] token and the last [SEP] token placed between language tokens and objects, similar in spirit to the approach described in (Zhou et al., 2020). In this way, we can generate temporally independent agent states for an entire trajectory resulting in a sequence of states $\{h_1, h_2, \ldots, h_{|T|}\}$.

### 4.2 Segment-Level Recurrent Action Decoder

The ALFRED challenge requires models to learn to complete action sequences averaging 50 steps and spanning multiple navigation and manipulation sub-goals. However, due to the quadratic complexity of the self-attention mechanism, feeding long sequences to transformers is computationally expensive (Beltagy et al., 2020). Inspired by the TransformerXL model (Dai et al., 2019), we design the Segment-Level Recurrent Action Decoder architecture that models long trajectories with recurrent segment-level state reuse. At training time we divide trajectories into temporal segments of size $s$. Given two consecutive segments, $s_i$ and $s_{i+1}$, EmBERT caches the representations generated for segment $s_i$. The computed gradient does not flow from $s_{i+1}$ to $s_i$, but cached representations are used as extended context. When predicting the next action, the model can still perform self-attention over the previous segment representations, effectively incorporating additional contextual information that spans an high number of previous timesteps.

The TransformerXL model is intended as an encoder-only architecture which is not able to perform cross-attention with some encoder hidden states. Therefore, we introduce two novel elements to its architecture: 1) encoder hidden states cache; 2) cross-attention over encoder states. First, our extended context is composed of both agent state representations and hidden states from the previous segment $s_i$. In addition, to perform
cross-attention between decoder and encoder hidden states, we modify the TransformerXL self-attention mechanism following common practice in designing transformer decoders (Vaswani et al., 2017). EmBERT encodes the previous actions for the current timestep $a_{t-1}$ and extracts an action embedding $a_t$ from a learnable embedding matrix $A \in \mathbb{R}^{|A| \times d_e}$. In the TransformerXL’s multi-head self-attention layers, we generate keys and values from the agent state representations (encoder) and queries from the action embeddings (decoder). We obtain time-dependent agent state representations $\{\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_{|T|}\}$ as output.

Given time-dependent hidden states, the model predicts action and object mask outputs. We learn a probability distribution over the agent actions $A$ by using a two layer feedforward network (FFN) with dropout and GeLU (Hendrycks and Gimpel, 2016) activation receiving the hidden state $\tilde{h}_t$ for the timestep $t$:

$$\tilde{h}_t = \text{GeLU}(\tilde{h}_t W^1) \quad P(a_t|\tilde{h}_t) = \text{softmax}(\tilde{h}_t W^2),$$

where $W^1 \in \mathbb{R}^{d_e \times d_e}$ and $W^2 \in \mathbb{R}^{d_e \times |A|}$ are two weight matrices. We use sequence-based cross-entropy loss (Sutskever et al., 2014), $L_A$, to supervise the action prediction task. In addition, we derive time-dependent fine-grained representations of token and object embeddings. We use conditional scaling (Dumoulin et al., 2018) to fuse the decoder hidden state $\tilde{h}_t$ with the embedding $H$ to produce the time-dependent embeddings $\tilde{H}$:

$$\tilde{c} = W^e_t \tilde{h}_t \quad \tilde{H}_t = \tilde{c} \cdot H_{i:m=1\ldots(m+n+|O|)},$$

where $W^e_t \in \mathbb{R}^{d_e \times d_e}$ is a weight matrix used to adapt the representation of the original decoder hidden state $\tilde{h}_t$. We predict target objects by selecting one bounding box among the detections in $V^F$ for manipulation actions, or any view for navigation actions. We treat object mask prediction as a classification task where the model first extracts time-dependent object embeddings $\tilde{O} = \tilde{H}_t$, $i = \{(m+n), \ldots, (m+n+|O|)\}$, and then generates logits for each object as follows:

$$\tilde{o}_i = \text{GeLU}(\tilde{O} W^2) \quad P(o_i|\tilde{O}_i) = \text{softmax}(\tilde{o}_i W^2).$$

where $W^1_o \in \mathbb{R}^{d_e \times d_e}$ and $W^2_o \in \mathbb{R}^{d_e \times 1}$ are two weight matrices. At training time, we determine the target object by using the Intersection-Over-Union score between the predicted object masks generated by MaskRCNN for each object and the gold object mask. To supervise this classification task, we use sequence-based cross-entropy loss, $L_O$.

### 4.3 Auxiliary tasks

During the EmBERT training, we jointly optimize $L_A$, $L_O$, and several auxiliary tasks.

#### Next Instruction Prediction. Several existing models for ALFRED encode the sequence of language instructions $I$ together with the goal (Table 1), or concatenate step-by-step instructions. These simplifications can prevent the model from carefully attending to relevant parts of the visual scene. EmBERT takes the first instruction at time $t = 0$, and performs an auxiliary prediction task to advance from instruction $I_j$ to instruction $I_{j+1}$. To supervise the next-instruction decision, we create a binary label for each step of the trajectory that indicates whether that step is the last step for a specific sub-goal, as obtained by $m_a(i)$. We use a similar FNN as Equation 1 to model a Bernoulli variable used to decide when to advance to the next instruction. We denote the binary cross-entropy loss used to supervise this task as $L_{INST}$.

#### Object Target Predictions. EmBERT predicts a target object for navigation actions, together with the receptacle object containing the target, for example a table on which a box sits (Figure 2). For these tasks, we use an equivalent prediction layer to the one used for object prediction. We denote the cross-entropy loss associated with these task by $L_{NAV}$ and $L_{RECP}$.

#### Visual Region Classification. Class-conditioned representations are useful for agent manipulation, especially when combined with hard-crafted procedures for object selections (Singh et al., 2020). Inspired by masked region modeling tasks (Chen et al., 2020b; Shrivastava et al., 2021), we select with $\%15$ probability some objects part of the agent view in a given timestep $t$ and we ask the model to predict their classes. Given the instruction *Turn around and walk to the book on the desk*, at the very first timestep of the trajectory it is likely that none of the mentioned objects are visible. Thus, we assume that at the last step of a sub-goal the agent will have in view the objects associated with the instruction. For the prediction task, we directly use the time-dependent object embeddings $\tilde{O}$ and use an FFN (similar to Equation 1) to estimate a probability distribution over the ALFRED object labels. We use a cross-entropy loss denoted by $L_{VR}$ as supervision for this task.
5 Experiments and Results

EmBERT achieves competitive performance with state of the art models on the ALFRED leaderboard test sets (Table 2), surpassing all but ET (Pashevich et al., 2021) and ABP (Kim et al., 2021) on Seen test fold performance (Table 3) at the time of writing. Notably, EmBERT achieves this performance without augmenting ALFRED data with additional language instructions, as is done in ET (Pashevich et al., 2021), or visual distortion as used in ABP (Kim et al., 2021).

Implementation Details. EmBERT is implemented using AllenNLP (Gardner et al., 2017), PyTorch-Lightning,2 and Huggingface-Transformers (Wolf et al., 2019). We train using the Adam optimizer with weight fix (Loshchilov and Hutter, 2017), learning rate $2e^{-5}$, and linear rate scheduler without warmup steps. We use dropout of 0.1 for the hidden layers of the FFN modules and gradient clipping of 1.0 for the overall model weights. Our TransformerXL-based decoder is composed of 2 layers, 8 attention heads, and uses a memory cache of 200 slots. At training time, we segment the trajectory into 10 timesteps. In order to optimize memory consumption, we use bucketing based on the trajectory length. We use teacher forcing (Williams and Zipser, 1989) to supervise EmBERT during the training process. To decide when to stop training, we monitor the average between action and object selection accuracy for every timestep based on gold trajectories. The best epoch according to that metric computed on the validation seen set is used for evaluation. The total time for each epoch is about 1 hour for a total of 20 hours for each model configuration using EC2 instances p3.8xlarge using 1 GPU.

Action Recovery Module. For obstacle avoidance, if a navigation action fails, for example the agent choosing MoveAhead when facing a wall, we take the next most confident navigation action at the following timestep, as in MOCA (Singh et al., 2020). We introduce an analogous object interaction recovery procedure. When the agent chooses an interaction action such as Slice, we first select the bounding box of highest confidence to retrieve an object interaction mask. If the resulting API action fails, for example if the agent attempts to Slice a Kettle object, we choose the next highest confidence bounding box at the following timestep. The ALFRED challenge ends an episode when an agent causes 10 such API action failures.

Comparison to Other Models. Table 2 gives EmBERT performance against top and baseline models on the ALFRED leaderboard at the time of writing. Seen and Unseen sets refer to tasks in rooms that were or were not seen by the agent at training time. We report Task success rate and Goal Conditioned (GC) success rate. Task success rate is the average number of episodes completed successfully. Goal conditioned success rate is more forgiving; each episode is scored in $[0, 1]$ based on the number of subgoals satisfied, for example, in a stack & place task if one of two mugs are put on a table, the GC score is 0.5 (Shridhar et al., 2020). Path weighted success penalizes taking more than the number of expert actions necessary for the task.

EmBERT outperforms MOCA (Singh et al., 2020) on Unseen scenes, and several models on Seen scenes. The primary leaderboard metric is Unseen success rate, measuring models’ generalization abilities. Among competitive models, EmBERT outperforms only MOCA at Unseen generalization success. Notably, EmBERT remains competitive on Unseen path-weighted metrics, because it does not perform any kind of exploration or mapping as in HLSM (Blukis et al., 2021) and

---

2https://www.pytorchlightning.ai/
### Validation Fold Performance

<table>
<thead>
<tr>
<th>Init Weights #SB</th>
<th>Mem Nav $O$ $P(O)$</th>
<th>VRC</th>
<th>Task</th>
<th>GC</th>
<th>Task</th>
<th>Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSCAR 18 200</td>
<td>✓</td>
<td>✓</td>
<td>28.54 (22.88)</td>
<td>38.69 (31.28)</td>
<td>1.46 (.72)</td>
<td>10.19 (6.25)</td>
</tr>
<tr>
<td>OSCAR 18 200</td>
<td>✓</td>
<td>✓</td>
<td>34.76 (28.46)</td>
<td>41.30 (35.50)</td>
<td>3.66 (1.55)</td>
<td>12.61 (7.49)</td>
</tr>
<tr>
<td>OSCAR 18 200</td>
<td>✓</td>
<td>✓</td>
<td>36.22 (27.05)</td>
<td>44.57 (35.23)</td>
<td>4.39 (2.21)</td>
<td>13.03 (7.54)</td>
</tr>
<tr>
<td>OSCAR 18 200</td>
<td>✓</td>
<td>✓</td>
<td>37.44 (28.81)</td>
<td><strong>44.62 (36.41)</strong></td>
<td><strong>5.73 (3.09)</strong></td>
<td><strong>15.91 (9.33)</strong></td>
</tr>
<tr>
<td>OSCAR 18 200</td>
<td>✓</td>
<td>✓</td>
<td>23.66 (17.62)</td>
<td>29.97 (24.16)</td>
<td>2.31 (1.24)</td>
<td>12.08 (7.62)</td>
</tr>
<tr>
<td>BERT 18 200</td>
<td>✓</td>
<td>✓</td>
<td>26.46 (19.41)</td>
<td>35.70 (27.04)</td>
<td>3.53 (1.77)</td>
<td>13.02 (7.57)</td>
</tr>
<tr>
<td>OSCAR 9 200</td>
<td>✓</td>
<td>✓</td>
<td>29.30 (20.14)</td>
<td>36.28 (27.21)</td>
<td>3.06 (1.13)</td>
<td>12.17 (6.69)</td>
</tr>
<tr>
<td>OSCAR 9 200</td>
<td>✓</td>
<td>✓</td>
<td>31.75 (23.52)</td>
<td>38.80 (32.21)</td>
<td>2.56 (1.28)</td>
<td>12.97 (8.24)</td>
</tr>
<tr>
<td>OSCAR 9 200</td>
<td>✓</td>
<td>✓</td>
<td>20.37 (16.30)</td>
<td>28.64 (23.11)</td>
<td>1.46 (0.75)</td>
<td>10.47 (6.26)</td>
</tr>
<tr>
<td>OSCAR 9 200</td>
<td>✓</td>
<td>✓</td>
<td>28.33 (20.77)</td>
<td>36.83 (28.03)</td>
<td>2.68 (1.18)</td>
<td>11.60 (6.78)</td>
</tr>
<tr>
<td>OSCAR 9 200</td>
<td>✓</td>
<td>✓</td>
<td>27.84 (20.66)</td>
<td>36.59 (27.97)</td>
<td>2.44 (1.06)</td>
<td>11.46 (6.76)</td>
</tr>
<tr>
<td>OSCAR 18 1</td>
<td>✓</td>
<td>✓</td>
<td>25.31 (18.79)</td>
<td>34.27 (26.09)</td>
<td>3.42 (1.49)</td>
<td>12.25 (7.34)</td>
</tr>
<tr>
<td>OSCAR 9 1</td>
<td>✓</td>
<td>✓</td>
<td>20.98 (13.98)</td>
<td>33.33 (22.74)</td>
<td>1.10 (0.60)</td>
<td>10.33 (4.69)</td>
</tr>
<tr>
<td>OSCAR 18 1</td>
<td>✓</td>
<td>✓</td>
<td>21.95 (12.99)</td>
<td>35.04 (22.31)</td>
<td>1.58 (.54)</td>
<td>11.08 (6.18)</td>
</tr>
<tr>
<td>OSCAR 0 200</td>
<td>✓</td>
<td>✓</td>
<td>28.33 (20.77)</td>
<td>36.83 (28.03)</td>
<td>2.68 (1.18)</td>
<td>11.60 (6.78)</td>
</tr>
<tr>
<td>MOCA (Singh et al., 2020)</td>
<td></td>
<td></td>
<td>18.90 (13.20)</td>
<td>28.02 (21.81)</td>
<td>3.65 (1.94)</td>
<td>13.63 (8.30)</td>
</tr>
</tbody>
</table>

Table 3: Validation Fold Performance. We present ablations adjusting the number of side-view bounding boxes, attended memory length, with and without predicting navigation target $O$, target parent object $P(O)$, and visual region classification (VRC) loss. We also explore initializing our multi-modal encoder with BERT versus OSCAR initialization. The highest values per fold and metric are shown in blue. Path weighted metrics are given in parenthesis.

We do not utilize the MOCA Instance Association in Time module (Singh et al., 2020) that is mimicked by ET (Pashevich et al., 2021). That module is conditioned based on the object class of the target object selected across timesteps. Because we directly predict object instances without conditioning on a predicted object class, our model must learn instance associations temporally in an implicit manner, rather than using such an inference time “fix”.

**EmBERT Ablations.** Removing the object-centric navigation prediction unique to EmBERT decreases performance on all metrics (Table 3). We show that limiting memory for the action decoder to a single previous timestep, initializing with BERT rather than OSCAR weights, and limiting vision to the front view all decrease performance in both Seen and Unseen folds.

We find that our parent prediction and visual region classification losses, however, do not improve performance. To investigate whether a smaller model would benefit more from these two auxiliary losses, we ran EmBERT with only 9 bounding boxes per side view, which enables fitting longer training segments in memory (we use 14 timesteps, rather than 10). We found that those losses improved EmBERT performance on the Unseen environment via both success rate and goal conditions metrics, and improved success rate alone in Seen environments when the non-frontal views were limited to 9, rather than 18, bounding boxes. Given the similar performance of EmBERT with all three auxiliary losses at 18 and 9 side views, we believe EmBERT is over-parameterized with the additional losses and 18 side view bounding boxes. It is possible that data augmentation efforts to increase the volume of ALFRED training data, such as those in ET (Pashevich et al., 2021), would enable us to take advantage of the larger EmBERT configuration.

### 6 Conclusions

We apply the insight that object-centric navigation is helpful for language-guided Embodied AI to a benchmark of tasks in home environments. Our proposed Embodied BERT (EmBERT) model adapts the pretrained language model transformer OSCAR (Li et al., 2020), and we introduce a decoupled transformer embedding and decoder step to enable attending over many features per timestep as well as a history of previous embedded states (Figure 1). EmBERT is the first to bring object-centric navigation to bear on language-guided, manipulation and navigation-based task completion. We find that EmBERT’s object-centric navigation and ability to attend across a long time horizon both contribute to its competitive performance with state-of-the-art ALFRED models (Table 3).

Moving forward, we will apply EmBERT to other benchmarks involving multimodal input
through time, such as vision and audio data (Chen et al., 2020a), as well as wider arrays of tasks to accomplish (Puig et al., 2018). To further improve performance on the ALFRED benchmark, we could conceivably continue training the Mask RCNN model from MOCA (Singh et al., 2020) forever by randomizing scenes in AI2THOR (Kolve et al., 2017) and having the agent view the scene from randomized vantage points with gold-standard segmentation masks available from the simulator. For language supervision, we could train and apply a speaker model for ALFRED to generate additional training data for new expert demonstrations, providing an initial multimodal alignment for EmBERT, a strategy shown effective in VLN tasks (Fried et al., 2018).

7 Implications and Impact

We evaluated EmBERT only on ALFRED, whose language directives are provided as a one-sided “recipe” accomplishing a task. The EmBERT architecture is applicable to single-instruction tasks like VLN, as long as auxiliary navigation object targets can be derived from the data as we have done here for ALFRED, by treating the “recipe” of step-by-step instructions as empty. In future work, we would like to incorporate our model on navigation tasks involving dialogue (Thomason et al., 2019; de Vries et al., 2018) and real robot platforms (Banerjee et al., 2020) where lifelong learning is possible (Thomason et al., 2015; Johansen et al., 2020). Low-level physical robot control is more difficult than the abstract locomotion used in ALFRED, and poses a separate set of challenges (Blukis et al., 2019; Anderson et al., 2020). By operating only in simulation, our model also misses the full range of experience that can ground language in the world (Bisk et al., 2020), such as haptic feedback during object manipulation (Thomason et al., 2020, 2016; Sinapov et al., 2014), and audio (Chen et al., 2020a) and speech (Harwath et al., 2019; Ku et al., 2020) features of the environment. Further, in ALFRED an agent never encounters novel object classes at inference time, which represent an additional challenge for successful task completion (Suglia et al., 2020).

The ALFRED benchmark, and consequently the EmBERT model, only evaluates and considers written English. EmBERT inherently excludes people who cannot use typed communication. By training and evaluating only on English, we can only speculate whether the object-centric navigation methods introduced for EmBERT will generalize to other languages. We are cautiously optimistic that, with the success of massively multi-lingual language models (Pires et al., 2019), EmBERT would be able to train with non-English language data. At the same time, we acknowledge the possibility of pernicious, inscrutable priors and behavior (Bender et al., 2021) and the possibility for targeted, language prompt-based attacks (Song et al., 2021) in such large-scale networks.

References


Byeonghwi Kim, Suvaansh Bhambri, Kunal Pratap Singh, Roozbeh Mottaghi, and Jonghyun Choi. 2021. Agent with the big picture: Perceiving surroundings for interactive instruction following. In Embodied AI Workshop CVPR.


Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond J. Mooney. 2016. Learning multi-modal grounded linguistic semantics by playing “I spy”. In International Joint Conference on Artificial Intelligence (IJCAI).


A Appendix

A.1 Additional Auxiliary Losses

In this section we describe alternative auxiliary losses that we designed for EmBERT training using ALFRED data. After validation, these configurations did not produce results comparable with the best performing model. This calls for a more detailed analysis of how to adequately design and combine such losses in the complex training regime of the ALFRED benchmark.

Masked Language Modeling  The task-oriented language in ALFRED differs from the web crawl text used to train large-scale Transformers. We tune our initial model weights using a masked language modeling objective (Devlin et al., 2019). We mask with a $\%15$ probability a token among the ones in $I_g$ and $I_t$ at the very last step of a sub-goal. Differently from captions data or Wikipedia, when which such supervision should be provided is crucial. Given the instruction Turn around and walk to the book on the desk, at the very first timestep of the trajectory it is likely that none of the mentioned objects are visible. Thus, we assume that at the last step of a subgoal the agent will have in view the objects associated with the instruction. We apply the same conditional scaling approach to generate time-dependent language representations $\tilde{L}$ as the one used in Equation 2. We denote the masked language modeling loss used for this task by $L_{MLM}$.

Masked Region Modeling  This is analogous to the Visual Region Classification (VCR) loss that we integrated in the model. The main difference is that $15\%$ of the visual features are entirely masked (i.e., replaced with zero values) and we ask the model to predict them given the time-dependent representations generated by EmBERT for them.

Image-text Matching  The masked region and language modeling losses encourage the model to learn fine-grained object and language token representations, respectively. However, we are also interested in global representations that are expressive enough to encode salient information of the visual frames. For this reason, we design an additional loss $L_{IM}$. Given the state representation for the current timestep $t$, EmBERT predicts whether the current visual features can be associated with the corresponding language features or not. We maximize the cosine similarity between the visual features of the current timestep $t$ and the corresponding language features while, at the same time, minimizing the cosine similarity between the current visual features and other language instructions in the same batch. In this task, just like when modeling the robot state, we use $\tilde{L}_0$ as the language features and $\tilde{L}_{m+n}$ as the visual features. We define $L_{IM}$ the same way as the contrastive loss in CLIP (Radford et al., 2021). However, we expect the model to use the time-dependent representation of the agent state in order to truly understand the meaning of a language instruction. In this case the meaning of an instruction can be appreciated only after several timesteps when the corresponding sequence of actions has been executed.

A.2 EmBERT Asset Licenses

AI2THOR (Kolve et al., 2017) is released under the Apache-2.0 License, while the ALFRED benchmark (Shridhar et al., 2020) is released under the MIT License.