# Break and Make: Interactive Structural Understanding Using LEGO Bricks

Aaron Walsman,[1] Muru Zhang[1], Klemen Kotar[2], Karthik Desingh[1], Ali Farhadi[1], and Dieter Fox[1,3]

[1] University of Washington awalsman@cs.washington.edu
[2] Allen Institute for Artificial Intelligence
[3] NVIDIA

**Abstract.** Visual understanding of geometric structures with complex spatial relationships is a fundamental component of human intelligence. As children, we learn how to reason about structure not only from observation, but also by interacting with the world around us – by taking things apart and putting them back together again. The ability to reason about structure and compositionality allows us to not only build things, but also understand and reverse-engineer complex systems. In order to advance research in interactive reasoning for part-based geometric understanding, we propose a challenging new assembly problem using LEGO bricks that we call **Break and Make**. In this problem an agent is given a LEGO model and attempts to understand its structure by interactively inspecting and disassembling it. After this inspection period, the agent must then prove its understanding by rebuilding the model from scratch using low-level action primitives. In order to facilitate research on this problem we have built **LTRON**, a fully interactive 3D simulator that allows learning agents to assemble, disassemble and manipulate LEGO models. We pair this simulator with a new dataset of fan-made LEGO creations that have been uploaded to the internet in order to provide complex scenes containing over a thousand unique brick shapes. We take a first step towards solving this problem using sequence-to-sequence models that provide guidance for how to make progress on this challenging problem. Our simulator and data are available at github.com/aaronwalsman/ltron. Additional training code and PyTorch examples are available at github.com/aaronwalsman/ltron-torch-eccv22.

## 1 Introduction

The physical world is made out of objects and parts. Buildings are made out of roofs, rooms and walls, chairs are made out of seats, backs and legs, and cars have doors, wheels and windshields. The ability to reason about these parts and the structural relationships between them are a key component of our ability to build tools and shelters, solve complex organizational problems and manipulate the world around us. Building part-based reasoning capability into intelligent agents has been a long-standing goal of the computer vision, robotics and broader AI communities.

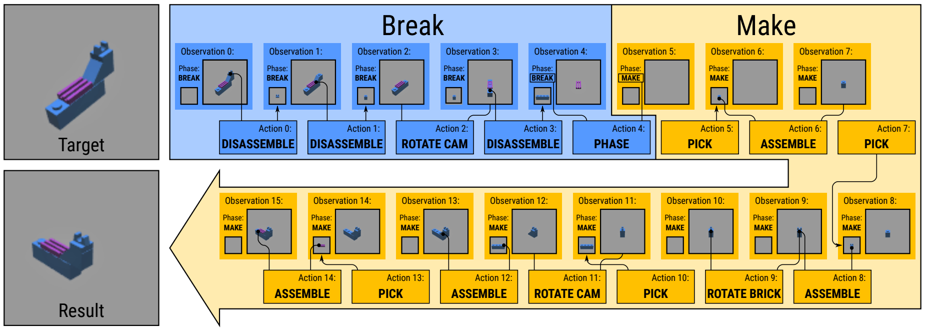
Target
Result
Break
Make
Observation 0: Phase: BREAK
Observation 1: Phase: BREAK
Observation 2: Phase: BREAK
Observation 3: Phase: BREAK
Observation 4: Phase: BREAK
Observation 5: Phase: MAKE
Observation 6: Phase: MAKE
Observation 7: Phase: MAKE
Action 0: DISASSEMBLE
Action 1: DISASSEMBLE
Action 2: ROTATE CAM
Action 3: DISASSEMBLE
Action 4: PHASE
Action 5: PICK
Action 6: ASSEMBLE
Action 7: PICK
Observation 15: Phase: MAKE
Observation 14: Phase: MAKE
Observation 13: Phase: MAKE
Observation 12: Phase: MAKE
Observation 11: Phase: MAKE
Observation 10: Phase: MAKE
Observation 9: Phase: MAKE
Observation 8: Phase: MAKE
Action 14: ASSEMBLE
Action 13: PICK
Action 12: ASSEMBLE
Action 11: ROTATE CAM
Action 10: PICK
Action 9: ROTATE BRICK
Action 8: ASSEMBLE

**Fig. 1.** A training example of the **Break and Make** task on a four-brick model in our dataset. During **Break** phase, the agent must learn to disassemble removable parts based on RGB images to understand the underlying structure. During the second **Make** phase, the agent must learn to pick bricks and reassemble the scene based on all past observations.

In this paper we propose **Break and Make**, a challenging new problem designed to investigate interactive structural reasoning using LEGO bricks. This problem is designed to simulate the process of reverse engineering: taking apart a complex object to learn more about its structure, and then using this newfound knowledge to put it back together again. This task is naturally divided into two phases. In the first **Break** phase, a learning agent is presented with a previously unseen LEGO model and has the opportunity to disassemble and inspect it in order to observe its internal structural and hidden components. After this, in the second **Make** phase, the agent is presented with an empty scene and must use the information gathered during the **Break** phase to rebuild the model from scratch. Both phases must be completed using visual action primitives designed to simulate the LEGO construction process. These actions require an agent to reason not only about individual bricks, but also the connection points between them.

In order to facilitate research on this challenging problem, we provide a dataset of 1727 ethically-sourced fan-made LEGO models with generous public licensing. These models range in size from 5 to 7302 individual bricks and use a library of 1790 distinct brick shapes. We also include a set of augmentations and a random model generator in order to provide more examples for large-scale training. Finally, we provide a 3D simulator and interactive learning environment with an OpenAI gym interface designed to train agents on this problem. Our simulator is compatible with a file format commonly used in the LEGO fan community, and is therefore capable of displaying and manipulating a wide range of models found online.

The **Break and Make** problem presents a difficult challenge for a number of reasons. First, the interchangeable nature of LEGO bricks and the large number of distinct brick shapes results in a very large state and action space. Second, this problem requires precise memory in order to bridge the long-term temporal distance between observations in the **Break** phase and reconstruction actions that must be taken in the **Make** phase. Third, this problem also requires precise spatial reasoning in order to carefully place bricks in the correct location using a visual observation and action space. Finally, it is difficult to provide direct supervision for this problem, even with accurate information from the simulator. This stems from the fact that it is not possible to directly compute which observations are necessary to capture the structural details of a model. We are however able to provide noisy supervision using a custom planner that reasons over visual observations. This planner generates a series of actions and observations that will feasibly disassemble and reassemble the model, but it comes with no guarantee that an agent equipped with only the visual observations from the sequence would have enough information to make the necessary decisions.

Despite these challenges, we show that progress can be made on LEGO assemblies containing up to 8 bricks using sequence-to-sequence models based on Transformers and LSTMs. We detail a suite of experiments that show the current capability of these models and demonstrate how performance deteriorates as the problem becomes more complex in terms of both model size and variety of brick shapes.

Our primary contributions are:

1. We introduce a challenging new interactive problem **Break and Make** requiring complex scene understanding and construction. Section 3 describes this problem in detail.
2. We present **LTRON**, a new simulator and dataset that allow interactive learning agents to build and manipulate LEGO models. Sections 3.2 and 3.4 provide details.
3. We also present a transformer-based network architecture **StudNet**, designed to make progress on this challenging problem. We compare this with with other sequence-to-sequence models and demonstrate the difficulty of attacking this problem using current techniques. Section 4 provides details on these approaches and Section 5.1 discusses their results.

## 2 Related Work

### 2.1 Understanding Compositional Structures

Interactive scene understanding and reasoning about compositional structure has origins in the early days of AI. An early example is Winograd's SHRDLU system [55] that used language instructions to interactively stack virtual blocks and answer questions posed by a human operator.

More recently researchers have introduced a number of interactive environments such as RoboThor[5], iGibson[43], Habitat[48] and MultiON[53] designed to simulate indoor environments for embodied learning agents. Many tasks have been proposed for these environments, such as goal-directed navigation [60], interactive question answering [11,4] and instruction following [44]. While many of these tasks and environments offer some degree of object manipulation, most of these interactions involve only a small number of object classes, and do not require the agent to reason about complex compositional structures. In contrast the **Break and Make** task requires an agent to reason in detail about these structures and how to build them from a library of 1790 unique parts.

In the non-interactive domain, researchers have released a number of simulated tasks and datasets [2,34,29] designed to provide access to a diverse set of objects with increasing detail, part structure and complexity. Others such as CLEVR [20], CLEVERER [59], and CATER [10] are designed around answering questions about object relationships in images and videos. In these settings, it is easy to procedurally generate a large dataset using randomization, but it has been difficult to generate datasets with large object and relationship vocabularies. Researchers have also taken great effort to annotate natural images and videos with detailed attributes [7], parts [52] and relationships [25].

Scene understanding via active or interactive perception is a classic way for robots and embodied agents to explore and model their environment. Researchers have investigated varying levels of detail and semantics in this space [51,32,41,39,47,46]. Previously it has been difficult to explore objects with fine-grained part structure in these settings due to the difficulty in collecting and annotating this data. **LTRON** provides complex models in an interactive environment, allowing agents to collect large amounts of data for researching complex cluttered environments with compositional structures. Another recent line of work explores learning physical properties of the world either from observations of rigid body interactions [58,56,57] or unsupervised physical interaction with a robot [8]. While we do not provide explicit rigid body dynamics in **LTRON**, we allow agents to explore extremely detailed physical structures with complex part-interactions at a scope that has not been practical in the past.

### 2.2 Building 3D Structures

In robotics, there has been a long-standing interest in enabling robots to build or assemble structured objects. Several authors have explored assembling IKEA furniture [30,45,27]. Others [16,42] have used Deep Reinforcement Learning and Learning from Demonstration methods to teach robots high precision assembly tasks using a real robot. While LTRON does not offer the realistic dynamics necessary to support traditional robotic manipulation, it does offer a high degree of scene complexity and compositionality which allows researchers to explore fine-grained spatial reasoning.

Recently construction and object-centric reasoning have become important topics in the reinforcement learning and AI community. Multiple datasets [54,21] have been developed to train agents to build and reason about geometric forms using CAD software. While they support a small number of primitive-based modelling tools, our building environment supports constructing models from over one thousand discrete brick types. Other recent works [1,9] have used reinforcement learning for block-stacking problems, and to create structures designed to achieve goals such as connecting or covering other blocks.

Researchers have also investigated the task of generating programs to describe and/or assemble shapes out of low level primitives [35,22] and reason about the relationships between them [15].

LEGO bricks are popular construction toys that are often an early entry point for children to learn about building. They are also an excellent abstraction for real-world construction problems, which has led other researchers to explore using LEGO for various construction problems. Several approaches have been proposed to automatically construct LEGO assemblies from a reference 3D body [23]. For example, multiple authors [36,26] have suggested methods for automated reconstruction based on genetic and evolutionary algorithms. Duplo bricks have also been used for tracking human demonstrations and assembly [12].

In contrast to these approaches, recent works have suggested data-driven deep learning approaches for LEGO problems based on generative models of graphs [49], and image to voxel reconstruction [28]. Similar to **Break and Make**, Chung et al. [3] propose a method for assembling LEGO structures from a reference image using interactive learning. Unlike **LTRON** these approaches use a use only a limited number of bricks, and do not support the large variety of bricks in the LEGO universe.

## 3 Task and Data

The **Break and Make** task requires an agent to learn how to inspect a LEGO assembly using rendered images, and then use the information gathered in this way to rebuild the assembly from scratch. Both the inspection phase and the construction phase are inherently interactive problems that require multi-step reasoning due to the ambiguities resulting from occlusions and the iterative nature of the building process. Many LEGO bricks have groups of similar neighbors which may appear identical under partial occlusion. Furthermore, complex structures often contain interior bricks that are not visible at all unless outer bricks are removed. These two factors mean that for many assemblies, there is no single viewpoint that completely captures an entire structure. Therefore in order to solve this problem an agent must often consider multiple viewpoints and take apart the assembly in order to fully understand it.

### 3.1 LEGO Bricks

A LEGO **brick** describes the shape and connection-point structure of a single LEGO part. While most LEGO bricks are a single rigid shape, some such as ropes and connector hoses are flexible. **LTRON** currently does not support these flexible components, so they are removed from all models before training. Some other bricks have moving parts, but in this case we break each of these into a separate brick shape for each moving component. We use polygon meshes extracted from the LDraw [19] package to represent all bricks. The **color** of a brick is represented as a single integer that refers to a specific RGB color value in a lookup table, which is consistent with LDraw conventions.

Each brick also contains a number of **connection points**. These describe how bricks may be connected to each other. The prototypical connection point is the short cylindrical stud that covers the top of many bricks in a rectangular grid, and the corresponding holes that cover the bottom. However, there are a large number of additional connection point types that exist in the LEGO universe, including technic pins, axles, clips, poles and ball/socket joints. In developing **LTRON** we have tried to faithfully represent as many of these as possible in order to provide a rich action space for interactive learning. Each of these connection points has a number of attributes related to its physical dimensions and compatibility with other bricks. One important attribute of all connection points is **polarity**, which describes whether the connection point is an extrusion (positive polarity) or cavity (negative polarity). We use part metadata from the LDCAD[33] software package in order to detect these connection points on bricks and provide manipulation actions for them.

We refer to a collection of multiple bricks and their 3D locations as an **assembly**. Mathematically, this can be modelled as a set tuples $a = \{b_1, b_2, \dots b_n\}$ where each tuple $b_i = (s_i, c_i, R_i, t_i)$ represents an **instance** of a single brick. Each of these instances $b_i$ contains a brick shape index $s_i \in N_{shapes}$, a color index $c_i \in N_{colors}$, a 3D rotation $R_i \in SO(3)$ and a 3D translation $t_i \in \mathbb{R}^3$. The relative placement of the instances, combined with their shapes and the connection points associated with those shapes allow us to construct a set of **connections** describing a pair of connection points that are in very close proximity to each other and are mutually compatible.

### 3.2 Environment

In order to manipulate an assembly, our environment provides two virtual work spaces. The first, which we refer to as the **table** work space contains the agent's work in progress towards inspecting or assembling a model. The second work space, which we refer to as the **hand** contains only a single brick that the agent is about to place, or has just removed from the table workspace. Each workspace provides a 2D image rendered from a camera viewpoint that can be controlled by the agent. The table is rendered at $256 \times 256$ pixels and the hand is rendered at $96 \times 96$ pixels. Many of the actions below require the agent to select one or more connection points on bricks in the hand or table workspace. To do this the agent must specify a 2D location in screen space, and the polarity of the connection point it wishes to select. This is similar to the Alfred dataset[44] and AI2 THOR 2.0[24] which allow interaction with objects using pixel-based selection. To reduce the size of this action space, the resolution of this selection space is downsampled by 4 to $64 \times 64$ for the table and $24 \times 24$ for the hand. **LTRON** uses these workspaces to provide the following manipulation actions as shown in Fig.2.

**Disassemble**: The agent must select a valid connection point in the table workspace. If the brick can be removed without causing collision, the associated brick instance is removed from the table work space and replaces any brick instance currently in the hand workspace.

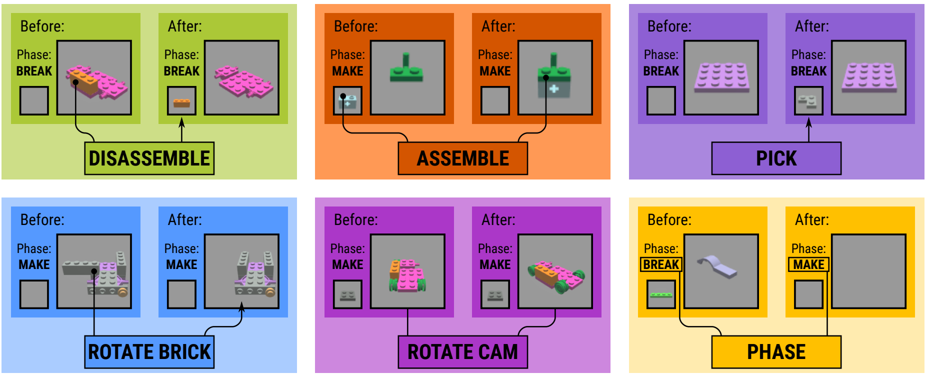

**Fig. 2.** We define in total six different actions. **Disassemble** removes a brick from the table workspace by selecting a connection point to detach. **Assemble** moves a brick from the hand workspace to the scene by attaching a pair of specified connection points. **Pick** selects a new brick shape and color and adds it to the hand workspace. **Rotate Brick** rotates the assembled brick at the selected connection point. **Rotate Camera** rotates the camera to reduce the ambiguity caused by occlusions. **Switch Phase** switches between break phase and make phase.

**Assemble**: The agent must specify valid and compatible connection points on one brick in the hand workspace and another in the table workspace. If the brick may be placed without collision, the brick in the hand is removed and placed into the table workspace attached to the specified connection point. If the table workspace is empty and there is no destination connection point to select, the agent may select a valid connection point in the hand workspace alone. This will remove the brick from the hand workspace and add it to the table workspace by placing the specified connection point at the origin.

**Pick**: The agent specifies a shape id and color id. A new brick with the specified shape and color replaces any brick instance currently in the hand work space.

**Rotate 90/180/270**: The agent must select a valid connection point on a brick in the table workspace. If rotating the brick will not cause collision, the brick is rotated by the specified angle about the primary axis of the connection point.

**Rotate Camera Left/Right/Up/Down/Frame**: In some cases it may be necessary to view an assembly from different viewpoints in order to effectively manipulate it, so we provide five actions for each workspace that the agent can use to manipulate the camera. The first four rotate the camera up, down, left or right about a fixed center point. Rotating left and right rotates by 45 degrees about the scene's up-axis, while rotating up and down alternate between a downward viewing angle 30 degrees above the center point and an upward viewing angle 30 degrees below the center point. The fifth **Frame** camera action moves the camera's fixed center point to the centroid of the current brick assembly.

**Switch Phase**: Finally there are two additional actions that switch from the Break phase to the Make phase, and that end the episode when the agent is finished building. Switching from the Break phase to the Make phase clears both workspaces.

### 3.3 Evaluation

The **Break and Make** task requires a learning agent to visually inspect a LEGO assembly in order to gather enough information to then build it again from scratch. In order to assess the capability of a learned model, it is necessary to compare the generated assembly that it builds with the target assembly it is trying to copy. We provide four different metrics that attempt to estimate various aspects of the agent's success.

**$\mathbf{F1_b}$ score:** The first metric is an F1 score over bricks in the two assemblies which we refer to as $F1_b$. This metric ignores pose and simply measures whether the agent was able to add the correct bricks to its estimated assembly regardless of how they are connected together. For this metric, we first remove pose information from the generated assembly $\hat{a}$ and the target assembly $a^*$ to produce a multi-set of brick shape and colors $m^* = \{(s_0^*, c_0^*) \dots (s_n^*, c_n^*)\}$ for the target assembly and another $\hat{m} = \{(\hat{s}_0, \hat{c}_0) \dots (\hat{s}_n, \hat{c}_n)\}$ for the assembly the agent generated. We can the compute true positives, false positives and false negatives as:

$$TP_b = m^* \cap \hat{m}, \quad FP_b = \hat{m} - m^*, \quad FN_b = m^* - \hat{m}$$

We then use these three quantities to compute an F1 score. Getting a score of 1.0 on this problem is necessary to rebuilding the assembly correctly, but it is not sufficient. This metric is still useful though because it allows us to categorize errors. If the agent was not able to rebuild the structure, but was able to identify the necessary bricks for that structure, then it may give us guidance for which aspect of the system needs the most improvement.

**$\mathbf{F1_a}$ score:** Unlike $F1_b$, $F1_a$ includes pose and is designed to measure the accuracy if the entire assembly. In this metric, we first define a rotation threshold $\theta_\epsilon$ and a distance threshold $d_\epsilon$ and say that two bricks $i$ and $j$ are *aligned* iff they have the same shape $s_i = s_j$ and color $c_i = c_j$ and their centers are close $||t_i - t_j|| < d_\epsilon$ and the geodesic distance between their orientations is close $G(R_i, R_j) < \theta_\epsilon$.

Given that we care more about the *relative* position of bricks to each other, than their *absolute* position in the scene, we first compute a single rotation $R_0$ and translation $t_0$ that bring as many bricks in $a^*$ into alignment with $\hat{a}$ as possible. We then consider each brick in $\hat{a}$ to be a true positive if it is aligned with another brick in $a^*$ and consider it to be a false negative otherwise. Any brick in $a^*$ that is not aligned to a brick in $\hat{a}$ is a false negative. We then use these quantities to compute $F1_a$.

**Assembly Edit Distance (AED):** While this $F1_a$ metric gives us a useful measure of similarity between two assemblies, it is possible that it may over-penalize some small mistakes. Consider the case where a long chain of bricks has been reconstructed correctly except for a single mistake in the middle. Because of the single rigid transform $R_0$ and $t_0$, we can only align either the top half or the bottom half of the reconstruction $\hat{a}$ with the target assembly $a^*$, and will incur a massive penalty for this single mistake. To mitigate this, we introduce Assembly Edit Distance (AED): we compute $R_0$ and $t_0$ as before, but once this is done, we mark all bricks that are aligned under this transformation and remove them from their respective assemblies. We then repeat this process with the remaining bricks and count how many rigid alignments must be computed until either the scene is empty, or the remaining bricks cannot be aligned because their shapes or colors do not match. We then add an additional edit penalty of 1, representing a single edit to remove the brick, for each brick in $\hat{a}$ left at the end of this process, and a penalty of 2, representing an edit to add the brick to the assembly and an edit to move it into place, for each brick in $a^*$ left at the end of the sequence.

**$\mathbf{F1_e}$ score:** An added bonus of the **AED** metric is that it can be used to compute a matching between each brick in the generated assembly $\hat{a}$ and the target assembly $a^*$. This matching allows us to compute one final metric: an F1 score over edges ($F1_e$), or connections between two bricks. We consider every pair of bricks that are connected to each other in the generated assembly $\hat{a}$ to be a true positive edge if both of those bricks have been matched to a brick in the target assembly $a^*$ and the matching bricks in the target assembly are also connected to each other. Otherwise the connected pair is a false positive. Any connected pair in the target assembly $a^*$ that is not matched in this way is a false negative. Like $F1_b$ this metric can be considered necessary but insufficient, but again it is useful because it lets us characterize the errors made during the build process. If the agent was not able to determine the correct spatial alignment of the bricks, but is able to connect the right bricks together, then it may tell us the agent is struggling with the precise placement necessary to align bricks correctly. This is similar to a metric used in Visual Genome [25], but uses our iterative matching edit distance to compute assignment and has no action/attribute labels on individual edges.

### 3.4 Dataset

We provide two sources of scene files to train and evaluate agents on these tasks. The first is a set of fan-made reproductions of official LEGO sets that have been uploaded to the Open Model Repository (OMR) [18], while the second is a set of randomly constructed models that we have generated with the **LTRON** simulator.

The OMR contains 1727 files that are incredibly diverse, ranging in size from 5 to 7302 bricks. The sets come from over fifty distinct product categories such as “City,” “Castle,” and “Star Wars” that have been released over a span of several decades and use 1790 distinct brick shapes. These files have many properties in common with other naturally occurring data sources such as a long tail of increasingly rare bricks, and edge-cases that are difficult to model. This is a blessing to researchers who are interested in building models that can handle complex data distributions, and a curse to those looking for quick progress. In general these models are much larger than we are presently able to train on. Both the mean and the median number of bricks in a scene is more than one hundred, while our experiments below show that current methods struggle with scenes containing only eight bricks. In order to generate a large amount of training data with smaller scenes, we have sliced these models into compact connected components using the connection points to find groups of connected bricks. In all cases we have used a master train/test split on the original files to inform the train/test on all slices of those files. Table 1 shows the train test splits for these slices. See the supplementary material for more details on the statistics, slicing procedure and cleaning process of this data.

In contrast to the OMR data above, our randomly generated models are constructed by iteratively selecting brick shapes and colors at random and attempting to connect them to other bricks using randomly selected compatible connection points. This provides a much larger source of data that is in many ways easier to use for training, but unfortunately has many qualitative differences from the more natural OMR data. For example OMR scenes with a similar number of bricks tend to be much more compact than our randomly generated files as a byproduct of the human designers' preferences for tightly fitting configurations. Similarly, the OMR scenes exhibit more symmetry, and more high-level structure such as clearly identifiable walls and branching structures. Despite these issues, this randomly constructed data is still very useful as a way to explore how the problem becomes easier as we reduce the number of brick shapes.

| Open Model Repository | Train Scenes | Test Scenes | Total Scenes |
|---|---|---|---|
| Original Scenes | 1360 | 367 | 1727 |
| 2 Brick Slices | 136072 | 2000 | 138072 |
| 4 Brick Slices | 61514 | 2000 | 63514 |
| 8 Brick Slices | 28094 | 2000 | 30094 |
| Random Construction | Train Scenes | Test Scenes | Total Scenes |
| 2 Bricks | 50000 | 2000 | 52000 |
| 4 Bricks | 50000 | 2000 | 52000 |
| 8 Bricks | 50000 | 2000 | 52000 |

**Table 1.** Train/test split sizes for the Open Model Repository and our Randomly Generated Data.

## 4 Methods

### 4.1 Model

Our StudNet models are based on the popular Transformer[50] architecture. In this model, the input images are first broken into 16 × 16 pixel tiles similar to the VIT architecture[6]. The model then extracts features from each tile using a learned linear layer and two positional embeddings, one that encodes the tile's XY coordinates in the image and another that encodes the tile's frame id in the temporal sequence. We unroll the XY coordinates of the image into a single one-dimensional coordinate space and concatenate the coordinates of the table image and the hand image so that a single index can be used to determine which image the tile belongs to and its 2D location. These tile features are then fed into a transformer that uses GPT-style[37] causal masking to prevent tokens that occur early in the sequence from paying attention to later tokens.

Transformer models notoriously require very large memory due to the $N^2$ attention mechanism that allows for long-range connectivity between tokens in the sequence. In order to make this architecture tractable on the long sequences of tokens produced by **LTRON**, we employ a simple but effective data compression technique: at each step we only include image tiles which have changed since the previous frame. In the first frame, we also remove all tiles that contain only the solid background color. Given that manipulating a single brick usually only changes a small portion of the image, this results in substantial savings. In addition to the image tiles, we provide a token that specifies the current phase (**Break** or **Make**).

The **Break and Make** task requires an agent to take both discrete high-level actions as well as select low-level pixel locations to assemble and disassemble bricks. We model this using five separate heads: a mode head that selects one of the primary action types (see Figure 2) to take at each step, a shape selector head and color selector head that are used when picking up a new brick, and a table location and hand location heatmap that is used to select pixel locations for brick interaction. The shape and color heads are linear layers that project from the transformer hidden dimension to the number of shapes and colors used in a particular experiment. Unfortunately we cannot decode a dense heatmap for the pixel locations directly from the tokens coming out of the transformer encoder because our compression strategy throws many of these tokens away. We experiment with two different decoder styles to address this issue.

The first, which we refer to as StudNet-A uses a separate transformer decoder layer. This layer receives a dense positional encoding as the query tokens, and the output of the encoder as the key and value tokens resulting in a dense output. Although some details differ, this is similar to the Perceiver IO[17] and MAE[13] models that do primary computation at a lower resolution and use cross-attention to expand to dense output when necessary. We decode at $16 \times 16$ resolution and upsample to $64 \times 64$.

The second decoder, which we refer to as StudNet-B, feeds the input images through a small convolutional network to produce a $64 \times 64$ feature map for the table and a $24 \times 24$ feature map for the hand. In our experiments we use the first layer of a Resnet-18[14] for this. Two additional heads, one for the table workspace and another for the hand workspace, compute a single feature from a per-frame readout token, and use dot-product attention with the convolutional feature map to produce a heatmap of click locations.

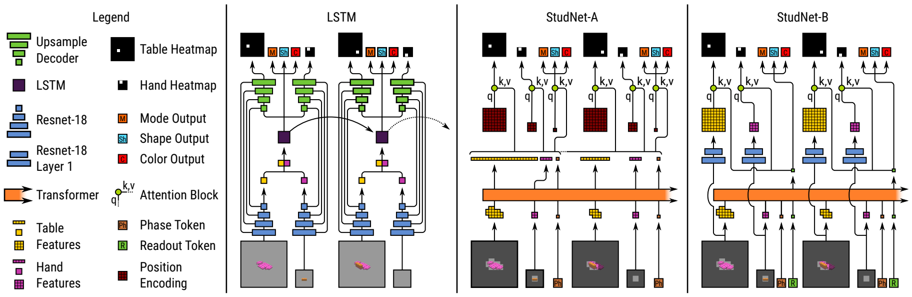

**Fig. 3.** Network architectures used in our experiments.

We compare these models against a convolution and LSTM baseline. This model takes guidance from the ALFRED Dataset [44] which similarly requires an agent to reason about high level actions as well as pixel-based selection. In this network, the images from both the table and hand workspaces are fed through a Resnet-18 backbone [14], and are then concatenated and passed to an LSTM. The output of this LSTM is then decoded using five heads. The first three produce the mode, shape and color actions. The second and third heads tile the LSTM feature to match the shape of the table and hand resnet features, then upsample these with UNET-style [38]/FPN [31]-style lateral connections from the image encoder to produce a dense feature that is used to select cursor locations. In experiments we use two versions of this model, one trained from scratch, and another where the Resnet-18 backbone has been pretrained on a pixel-labeling task designed to densely predict brick shapes and colors.

### 4.2 Training

We train the models above using behavior cloning on offline sequences. In order to generate these sequences, we have developed a visual planner that interfaces directly with **LTRON**. This planner uses hidden state information combined with rendered occlusion maps to reason about which bricks are currently visible in the scene and plan assembly and disassembly sequences accordingly. While this information allows the planner to determine which bricks can be manipulated, it does not strictly guarantee that the visual information acquired during the planning process is enough to unambiguously resolve the full 3D structure of the scene, or correctly identify the shapes of every brick. This is due to the fact that many brick shapes look identical to others when viewed from certain angles or under partial occlusion, and so it may be important to change the camera viewpoint or disassembly order to resolve these ambiguities. Due to the large number of brick shapes, we have not attempted to exhaustively catalogue when and how these ambiguities arise for every combination of brick shapes. Therefore the planner currently has no way of knowing when these conditions occur.

### 4.3 Limitations

The visual planner can be quite slow and uses a two-stage process that requires reasoning over groups of individual actions. Both of these issues make it difficult to use the planner as an expert for methods such as DAgger[40] that require the expert to produce labels for sequences generated by the model. We therefore do not attempt to solve **Break and Make** using these approaches at the present time, and limit ourselves to methods that can train on a static dataset. Building improved planners with the ability to quickly provide high-quality actions would be beneficial for this problem.

## 5 Experiments

### 5.1 Break and Make

We evaluate the models above on the Random Construction and Sliced OMR datasets at three fixed scene sizes: two bricks, four bricks and eight bricks. While these scenes are quite small compared to the complete models in the Open Model Repository, they often require dozens of interaction steps to complete and present a challenging problem.

On the random construction data with six brick types and six colors, all models make substantial progress on small scenes. Table 2 shows the models' performance on each of these tasks under the four metrics described in Section 3.3. Note that performance drops substantially as the scenes get larger.

**Random Construction**

| | 2 Bricks | | | | 4 Bricks | | | | 8 Bricks | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | $F1_b$ | $F1_e$ | $F1_a$ | AED | $F1_b$ | $F1_e$ | $F1_a$ | AED | $F1_b$ | $F1_e$ | $F1_a$ | AED |
| LSTM | 0.61 | 0.38 | 0.43 | 2.16 | 0.41 | 0.09 | 0.13 | 7.25 | 0.02 | 0.00 | 0.02 | 16.05 |
| Pretr. LSTM | 0.70 | 0.51 | 0.45 | 1.89 | 0.25 | 0.01 | 0.08 | 8.46 | 0.03 | 0.00 | 0.02 | 16.09 |
| StudNet-A | 0.90 | 0.86 | 0.58 | 1.11 | 0.56 | 0.29 | 0.24 | 5.80 | 0.02 | 0.01 | 0.01 | 15.87 |
| StudNet-B | 0.87 | 0.77 | 0.57 | 1.30 | 0.64 | 0.34 | 0.25 | 5.48 | 0.38 | 0.14 | 0.12 | 13.90 |

**Table 2.** Test results of our four models on randomly constructed assemblies across three scene sizes. See Section 3.3 for details on metrics.

The Sliced OMR dataset contains 1790 brick shapes and 98 colors making it structurally and visually significantly more challenging than the random construction dataset. Table 3 illustrates that all of the models we tested score significantly lower on this dataset. In particular our StudNet-A transformer architecture fails to correctly learn to switch from disassembling to rebuilding the LEGO models and thus scores very poorly across all of our metrics. Our StudNet-B architecture shows the best overall performance, demonstrating that progress can be made even on the most challenging 8 brick dataset. This illustrates not only that **Break and Make** is a fundamentally hard problem, but also that its difficulty can be regulated by the dataset selection while maintaining the same action space and problem structure. This allows future work to make meaningful progress on simple datasets like Random Construction and then progress to ever more difficult datasets.

**Open Model Repository**

| | 2 Bricks | | | | 4 Bricks | | | | 8 Bricks | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | $F1_b$ | $F1_e$ | $F1_a$ | AED | $F1_b$ | $F1_e$ | $F1_a$ | AED | $F1_b$ | $F1_e$ | $F1_a$ | AED |
| LSTM | 0.43 | 0.33 | 0.31 | 2.76 | 0.10 | 0.03 | 0.07 | 7.67 | 0.01 | 0.00 | 0.01 | 16.01 |
| Pretr. LSTM | 0.45 | 0.34 | 0.33 | 2.86 | 0.04 | 0.01 | 0.03 | 8.16 | 0.00 | 0.00 | 0.00 | 15.97 |
| StudNet-A | 0.00 | 0.00 | 0.00 | 3.99 | 0.00 | 0.00 | 0.00 | 8.08 | 0.00 | 0.00 | 0.00 | 16.01 |
| StudNet-B | 0.36 | 0.18 | 0.29 | 3.74 | 0.14 | 0.02 | 0.12 | 8.30 | 0.05 | 0.00 | 0.04 | 16.05 |

**Table 3.** Test results of our four models on OMR assemblies across three scene sizes. See Section 3.3 for details on metrics.

### 5.2 Ablations and Failure Analysis

Given the relatively low performance of the models presented here on the break and make task, we also conducted several experiments designed to discover which part of this problem is most difficult for future research. Appendix D.1 contains the details of these experiments. We also attempt to pretrain a model on the randomly generated assemblies and fine-tune on the OMR data. Appendix D.2 contains details. Finally we also provide a human baseline to verify the tractability of this problem in Appendix D.3.

## 6 Conclusion

**LTRON** and the **Break and Make** challenge offer an ideal environment to study a number of important technical problems in Machine Learning and Artificial Intelligence. First, the **LTRON** simulator offers an environment to explore interactive building and construction problems at a level of detail and granularity that has not previously been possible. Second, while we have only been able to make progress on very small LEGO models in this paper, **LTRON** has the ability to represent very large assemblies with hundreds and even thousands of bricks. Our hope is that the existence of these very difficult large-scale tasks that are currently beyond the scope of modern temporal-spatial visual modelling techniques will inspire researchers to explore new ways to scale algorithms and hardware to accomplish the goals. Finally **Break and Make** provides an ideal setting to explore interactive learning algorithms designed for long-term credit assignment, as agents must connect low-level actions taken during disassembly and inspection with reward signals collected in the distant future during reassembly.

## References

1. Bapst, V., Sanchez-Gonzalez, A., Doersch, C., Stachenfeld, K., Kohli, P., Battaglia, P., Hamrick, J.: Structured agents for physical construction. In: International Conference on Machine Learning. pp. 464–474. PMLR (2019)
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
3. Chung, H., Kim, J., Knyazev, B., Lee, J., Taylor, G.W., Park, J., Cho, M.: Brick-by-brick: Combinatorial construction with deep reinforcement learning. Advances in Neural Information Processing Systems **34**, 5745–5757 (2021)
4. Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., Batra, D.: Embodied question answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–10 (2018)
5. Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., Salvador, J., Schwenk, D., VanderBilt, E., Wallingford, M., et al.: Robothor: An open simulation-to-real embodied ai platform. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3164–3174 (2020)
6. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
7. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1778–1785. IEEE (2009)
8. Finn, C., Goodfellow, I., Levine, S.: Unsupervised learning for physical interaction through video prediction. arXiv preprint arXiv:1605.07157 (2016)
9. Ghasemipour, S.K.S., Kataoka, S., David, B., Freeman, D., Gu, S.S., Mordatch, I.: Blocks assemble! learning to assemble with large-scale structured reinforcement learning. In: International Conference on Machine Learning. pp. 7435–7469. PMLR (2022)
10. Girdhar, R., Ramanan, D.: Cater: A diagnostic dataset for compositional actions and temporal reasoning. ArXiv **abs/1910.04744** (2020)
11. Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., Farhadi, A.: Iqa: Visual question answering in interactive environments. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4089–4098 (2018)
12. Gupta, A., Fox, D., Curless, B., Cohen, M.: Duplotrack: A reatime system for authoring and guiding duplo model assembly. In: Proceedings of the 25th annual ACM symposium adjunct on User interface software and technology. ACM, New York, NY, USA (2012)
13. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Huang, J., Smith, C., Bastani, O., Singh, R., Albarghouthi, A., Naik, M.: Generating programmatic referring expressions via program synthesis. In: International Conference on Machine Learning. pp. 4495–4506. PMLR (2020)

16. Inoue, T., De Magistris, G., Munawar, A., Yokoya, T., Tachibana, R.: Deep reinforcement learning for high precision assembly tasks. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 819–825. IEEE (2017)
17. Jaegle, A., Borgeaud, S., Alayrac, J.B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al.: Perceiver io: A general architecture for structured inputs & outputs. arXiv preprint arXiv:2107.14795 (2021)
18. Jessiman, J., et al.: Open Model Repository. `https://omr.ldraw.org`
19. Jessiman, J., et al.: LDraw. `http://www.ldraw.org` (2022)
20. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C.L., Girshick, R.: Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In: CVPR (2017)
21. Jones, B., Hildreth, D., Chen, D., Baran, I., Kim, V.G., Schulz, A.: Automate: A dataset and learning approach for automatic mating of cad assemblies. ACM Transactions on Graphics (TOG) **40**(6), 1–18 (2021)
22. Jones, R.K., Barton, T., Xu, X., Wang, K., Jiang, E., Guerrero, P., Mitra, N.J., Ritchie, D.: Shapeassembly: Learning to generate programs for 3d shape structure synthesis. ACM Transactions on Graphics (TOG) **39**(6), 1–20 (2020)
23. Kim, J.: Survey on automated lego assembly construction (2015)
24. Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., Farhadi, A.: Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474 (2017)
25. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., Bernstein, M., Fei-Fei, L.: Visual genome: Connecting language and vision using crowdsourced dense image annotations (2016), `https://arxiv.org/abs/1602.07332`
26. Lee, S., Kim, J., Kim, J.W., Moon, B.R.: Finding an optimal lego brick layout of voxelized 3d object using a genetic algorithm. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 1215–1222 (2015)
27. Lee, Y., Hu, E.S., Lim, J.J.: Ikea furniture assembly environment for long-horizon complex manipulation tasks. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 6343–6349. IEEE (2021)
28. Lennon, K., Fransen, K., O'Brien, A., Cao, Y., Beveridge, M., Arefeen, Y., Singh, N., Drori, I.: Image2lego: Customized lego set generation from images. arXiv preprint arXiv:2108.08477 (2021)
29. Li, Y., Mo, K., Shao, L., Sung, M., Guibas, L.: Learning 3d part assembly from a single image. In: European Conference on Computer Vision. pp. 664–682. Springer (2020)
30. Lim, J.J., Pirsiavash, H., Torralba, A.: Parsing ikea objects: Fine pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2992–2999 (2013)
31. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
32. McCormac, J., Clark, R., Bloesch, M., Davison, A., Leutenegger, S.: Fusion++: Volumetric object-level slam. In: 2018 international conference on 3D vision (3DV). pp. 32–41. IEEE (2018)
33. Melkert, R.: LDCad. `http://www.melkert.net/LDCad` (2017)
34. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object un-

derstanding. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 909–918 (2019)
35. Nandi, C., Willsey, M., Anderson, A., Wilcox, J.R., Darulova, E., Grossman, D., Tatlock, Z.: Synthesizing structured cad models with equality saturation and inverse transformations. In: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 31–44 (2020)
36. Peysakhov, M., Regli, W.: Using assembly representations to enable evolutionary design of lego structures. Artificial Intelligence for Engineering Design, Analysis and Manufacturing **17**, 155 – 168 (2003)
37. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
38. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
39. Rosinol, A., Gupta, A., Abate, M., Shi, J., Carlone, L.: 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans (2020)
40. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635. JMLR Workshop and Conference Proceedings (2011)
41. Salas-Moreno, R.F., Newcombe, R.A., Strasdat, H., Kelly, P.H., Davison, A.J.: Slam++: Simultaneous localisation and mapping at the level of objects. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1352–1359 (2013)
42. Savarimuthu, T.R., Buch, A.G., Schlette, C., Wantia, N., Roßmann, J., Martínez, D., Alenyà, G., Torras, C., Ude, A., Nemec, B., et al.: Teaching a robot the semantics of assembly tasks. IEEE Transactions on Systems, Man, and Cybernetics: Systems **48**(5), 670–692 (2017)
43. Shen, B., Xia, F., Li, C., Martín-Martín, R., Fan, L., Wang, G., Buch, S., D'Arpino, C., Srivastava, S., Tchapmi, L.P., et al.: igibson, a simulation environment for interactive tasks in large realisticscenes. arXiv preprint arXiv:2012.02924 (2020)
44. Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., Fox, D.: Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10740–10749 (2020)
45. Suárez-Ruiz, F., Zhou, X., Pham, Q.C.: Can robots assemble an ikea chair? Science Robotics **3**(17), eaat6385 (2018)
46. Sucar, E., Wada, K., Davison, A.: Nodeslam: Neural object descriptors for multiview shape reconstruction. In: 2020 International Conference on 3D Vision (3DV). pp. 949–958. IEEE (2020)
47. Sui, Z., Chang, H., Xu, N., Jenkins, O.C.: Geofusion: geometric consistency informed scene estimation in dense clutter. IEEE Robotics and Automation Letters **5**(4), 5913–5920 (2020)
48. Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D.S., Maksymets, O., et al.: Habitat 2.0: Training home assistants to rearrange their habitat. Advances in Neural Information Processing Systems **34**, 251–266 (2021)
49. Thompson, R., Ghalebi, E., DeVries, T., Taylor, G.W.: Building lego using deep generative models of graphs. arXiv preprint arXiv:2012.11543 (2020)
50. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. arXiv preprint arXiv:1706.03762 (2017)

51. Vineet, V., Miksik, O., Lidegaard, M., Nießner, M., Golodetz, S., Prisacariu, V.A., Kähler, O., Murray, D.W., Izadi, S., Pérez, P., et al.: Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 75–82. IEEE (2015)
52. Wah, C., Branson, S., Perona, P., Belongie, S.: Multiclass recognition and part localization with humans in the loop. In: 2011 International Conference on Computer Vision. pp. 2524–2531. IEEE (2011)
53. Wani, S., Patel, S., Jain, U., Chang, A.X., Savva, M.: Multi-on: Benchmarking semantic map memory using multi-object navigation. In: Neural Information Processing Systems (NeurIPS) (2020)
54. Willis, K.D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J.G., Solar-Lezama, A., Matusik, W.: Fusion 360 gallery: A dataset and environment for programmatic cad reconstruction. arXiv preprint arXiv:2010.02392 (2020)
55. Winograd, T.: Shrdlu: A system for dialog (1972)
56. Wu, J., Lim, J.J., Zhang, H., Tenenbaum, J.B., Freeman, W.T.: Physics 101: Learning physical object properties from unlabeled videos. In: BMVC. vol. 2, p. 7 (2016)
57. Wu, J., Lu, E., Kohli, P., Freeman, B., Tenenbaum, J.: Learning to see physics via visual de-animation. In: NIPS. pp. 153–164 (2017)
58. Wu, J., Yildirim, I., Lim, J.J., Freeman, B., Tenenbaum, J.: Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. Advances in neural information processing systems **28**, 127–135 (2015)
59. Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., Tenenbaum, J.B.: Clevrer: Collision events for video representation and reasoning (2020)
60. Zhao, X., Agrawal, H., Batra, D., Schwing, A.: The Surprising Effectiveness of Visual Odometry Techniques for Embodied PointGoal Navigation. In: Proc. ICCV (2021)