This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration

De-An Huang*, Suraj Nair*, Danfei Xu*, Yuke Zhu, Animesh Garg,
Li Fei-Fei, Silvio Savarese, Juan Carlos Niebles
Computer Science Department, Stanford University

Abstract

Our goal is to generate a policy to complete an unseen task given just a single video demonstration of the task in a given domain. We hypothesize that to successfully generalize to unseen complex tasks from a single video demonstration, it is necessary to explicitly incorporate the compositional structure of the tasks into the model. To this end, we propose Neural Task Graph (NTG) Networks, which use conjugate task graph as the intermediate representation to modularize both the video demonstration and the derived policy. We empirically show NTG achieves inter-task generalization on two complex tasks: Block Stacking in BulletPhysics and Object Collection in AI2-THOR. NTG improves data efficiency with visual input as well as achieve strong generalization without the need for dense hierarchical supervision. We further show that similar performance trends hold when applied to real-world data. We show that NTG can effectively predict task structure on the JIGSAWS surgical dataset and generalize to unseen tasks.

1. Introduction

Learning sequential decisions and adapting to new task objectives at test time is a long-standing challenge in AI [5, 9]. In rich real domains, an autonomous agent has to acquire new skills with minimal supervision. Recent works have tackled the problem of one-shot imitation learning [8, 11, 40, 41] that learns from a single demonstration. In this work, we push a step further to address one-shot *visual* imitation learning that operates directly on *videos*. We first train a model on a set of seen in-domain tasks. The model can then be applied on a single video demonstration to obtain an execution policy of the new unseen task.

Learning directly from video is crucial for advancing the existing imitation learning approaches to real-world scenarios as it is infeasible to annotate states, such as object trajectories, in each video. We focus on *long-horizon tasks*, as real-world tasks such as cooking or assembly are inherently long-horizon and hierarchical. Recent works have attempted learning from pixel space [11, 27, 35, 42], but learning long-horizon tasks from video in a one-shot setting remains a challenge, since both the visual learning and task complexity exacerbate the demand for better data efficiency.

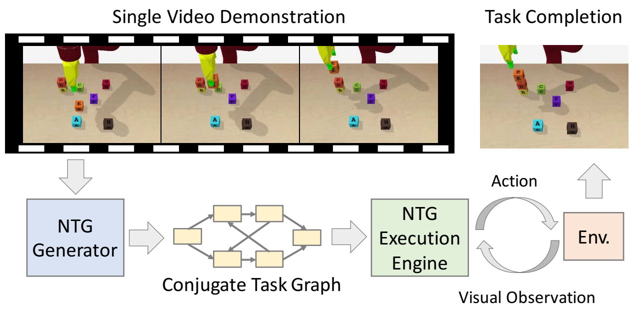


Figure 1. Our goal is to execute an unseen task from a single video demonstration. We propose Neural Task Graph Networks that leverage compositionality by using the task graph as the intermediate representation. This leads to strong inter-task generalization.

Our solution explicitly models the compositionality in the task structure and policy, enabling us to scale one-shot visual imitation to complex tasks. This is in contrast to previous works using unstructured task representations and policies [8, 11]. The use of compositionality has led to better generalization in Visual Question Answering [17, 20, 24] and Policy Learning [3, 7, 38]. We propose Neural Task Graph (NTG) Networks, a novel framework that uses task graph as the intermediate representation to explicitly modularize both the visual demonstration and the derived policy. NTG consists of a generator and an execution engine, where the generator builds a *task graph* from the task demo video to capture the structure of the task, and the execution engine interacts with the environment to perform the task conditioned on the inferred task graph. Figure 1 shows an overview of NTG Networks.

The main technical challenge in using graphical task representations is that the unseen demos can easily introduce states that are never observed during training. For example,

* indicates equal contribution

the goal state of an unseen block stacking task [8, 41] is a block configuration that never appears during training. This challenge is amplified by our goal of learning from visual observation without strong supervision, which obscures the state structure and prevents direct state space decomposition, as done in prior work [8]. Our key observation is that, while there can be countless possible states, the number of possible actions in a certain domain is often limited. We leverage this conjugate relationship between states and actions, and propose to learn NTG on the Conjugate Task Graph (CTG) [16], where the nodes are actions, and the states are captured by the edges. This allows us to modularize the policy and address the challenge of an unknown number of novel states. This is critical when operating in visual space, where states are high dimensional images and modeling a graph over a combinatorial state space is infeasible. Additionally, the CTG intermediate representation can yield alternate action sequences to complete the task, a property that is vital for generalization to unseen scenarios in a world with stochastic dynamics. This sets NTG apart from previous works that directly output the policy over options [41] or actions [8] from a single demonstration.

We evaluate NTG Networks on one-shot visual imitation learning in two domains: Block Stacking in a robot simulator [6] and Object Collection in AI2-THOR [23]. Both domains involve multi-step planning for interaction and are inherently compositional. We show that NTG significantly improves the data efficiency on these complex tasks for direct imitation from video by explicitly incorporating compositionality. We also show that with the data-driven task structure, NTG outperforms methods that learn unstructured task representation [8] and methods that use strong hierarchically structured supervision [41], albeit without requiring detailed supervision. Further, we evaluate NTG on real-world videos. We show that NTG can effectively predict task graph structure on the JIGSAWS [12] surgical dataset and generalize to unseen human demonstrations.

In summary, the main contributions of our work are: (1) Introducing compositionality to both the task and policy representation to enable one-shot visual imitation learning on long-horizon tasks; (2) Proposing Neural Task Graph (NTG) Networks, a novel framework that uses task graph to capture the structure and the goal of a task; (3) Addressing the challenge of novel visual state decomposition using a Conjugate Task Graph (CTG) formulation.

2. Related Work

Imitation Learning. Traditional imitation learning work uses physical guidance [1, 31] or teleoperation [39, 43] as demonstration. While, third-person imitation learning uses date from other agents or viewpoints [27, 35]. Recent methods for one-shot imitation learning [8, 11, 13, 40, 41, 42] can translate a single demonstration to an executable policy. The most similar to ours is NTP [41] that also learns long-horizon tasks. However, NTP (1) uses strong hierarchical frame label supervision and (2) suffers from a noticeable drop in performance with visual state. Our method reduces the need for this strong supervision, requiring only the demonstration action sequence during training, while achieving a performance boost of over 25% in success rates.

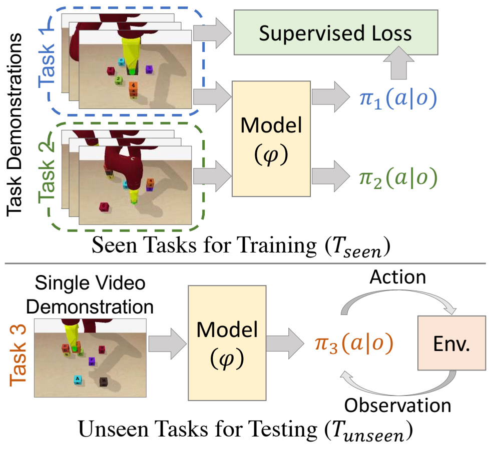


Figure 2. Overview of the setting of one-shot visual imitation learning. The seen tasks (Task 1 and 2) are used to train the model ϕ to instantiate the policy π_i from the demonstration. During testing, ϕ is applied to a single video demonstration from the unseen Task 3 to generate the policy π_3 to interact with the environment.

Task Planning and Representations. Conventionally task planning focuses on high-level plans and low-level state spaces [10, 36]. Recent works integrate perception via deep learning [15, 32, 44]. HTN compounds low-level sub-tasks into higher-level abstraction to reduce the planning complexity [29, 33]. Other representations include: integrating task and motion planning [21] and behavior-based systems [30]. In vision, And-Or Graphs capture the hierarchical structures and have been used to parse video demonstrations [26]. Unlike previous methods, our task graph representation is data-driven and domain-agnostic: we generate nodes and edges directly from task demonstrations.

Structural Video Understanding. Generating task graphs from demonstrations is related to video understanding. Annotation in videos is hard to obtain. One solution is to use the language as supervision. This includes instructional video [2, 18, 34], movie script [37, 45], and caption annotation [14, 25]. We focus on how the structure is helpful for task learning, and assume the annotation for the seen tasks.

Compositional Models in Vision and Robotics. Recent works have utilized compositionality to improve models' generalization, including visual question answering [4, 17, 20] and policy learning [3]. We show the same principle can significantly improve data efficiency in imitation learning to enable visual learning of complex tasks.

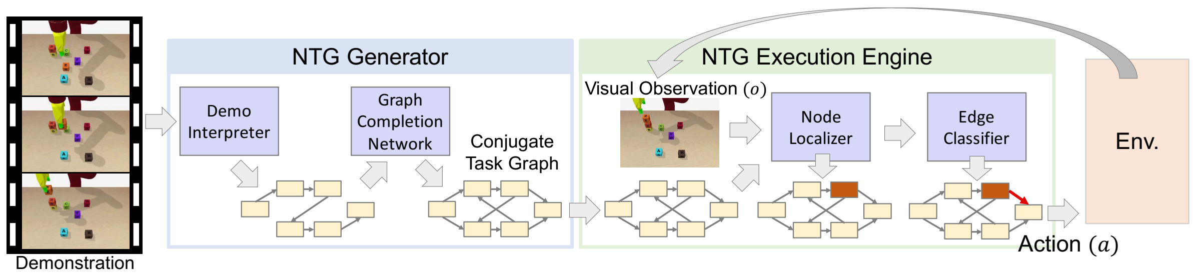


Figure 3. Overview of our Neural Task Graph (NTG) networks. The NTG networks consist of a generator that produces the conjugate task graph as the intermediate representation, and an execution engine that executes the graph by localizing node and deciding the edge transition in the task graph based on the current visual observation.

3. Problem Formulation

Our goal is to learn to execute a previously unseen task from a single video demonstration. We refer to this as one-shot *visual* imitation, where the model directly learns from visual inputs. Let $\mathbb{T}$ be the set of all tasks in the domain of interest, $\mathbb{A}$ be the set of high-level actions, and $\mathbb{O}$ be the space of visual observation. A video demonstration d for a task τ is defined as a video, $d^{\tau} = [o_1, \dots, o_T]$, that complete the task. As shown in Figure 2, $\mathbb{T}$ is split into two sets: $\mathbb{T}_{seen}$ with a large amount of demonstrations and supervision for training, and $\mathbb{T}_{unseen}$ with only task demonstrations for evaluation. The goal is to learn a model $\phi(\cdot)$ from $\mathbb{T}_{seen}$ that can instantiate a policy $\pi_d(a|o)$ from d to perform the tasks in $\mathbb{T}_{unseen}$ using visual observations.

The learning problem is formulated as learning a model $\phi(\cdot)$ that maps demonstration d to the policy $\phi(d) = \pi_d(a|o)$. $\mathbb{T}_{seen}$ is used to train this model with demonstrations and potentially extra supervision. At test time, given a demonstration d from an unseen task, the hope is that $\phi(\cdot)$ trained on $\mathbb{T}_{seen}$ can generalize to novel task instances in $\mathbb{T}_{unseen}$ and produce a policy that can complete the novel task illustrated by the visual demonstration.

4. Neural Task Graph Networks

We have formulated one-shot visual imitation as learning the model $\phi(\cdot)$ that maps a video demonstration to the policy. As shown in Figure 1, our key contribution is explicitly incorporating *compositionality* to improve the data efficiency of generalization. We decompose $\phi(\cdot)$ into two components: a *graph generator* $\phi_{gen}(\cdot)$ that generates the task graph G from the demonstration ($G = \phi_{gen}(d)$), and a *graph execution engine* $\phi_{exe}(\cdot)$ that executes the task graph and acts as the policy ($\pi_d = \phi_{exe}(G)$). The structure of the task graph G modularizes both the demonstration and the policy. This leads to stronger data efficiency of generalizing to unseen tasks. An overview is shown in Figure 3.

4.1. Neural Task Graph Generator

The NTG Generator generates a task graph capturing the structure of an unseen task from a single video demonstration. This is challenging since the video demonstration of an unseen task introduces novel visual states that are not observed in the seen tasks. This challenge is amplified by our goal of learning from *visual* observation, which prevents direct state space decomposition. In this case, generating the traditional task graph is ill-posed due to the exploding number of nodes. We address this by leveraging the conjugate relationship between state and action and work with the conjugate task graph [16], where the nodes are the actions, and the edges implicitly depend on the current state. In the experiments, we show that this scheme significantly simplifies the (conjugate) task graph generation problem.

Conjugate Task Graph (CTG). A *task graph* $\bar{G} = \{\bar{V}, \bar{E}\}$ contains nodes $\bar{V}$ as the states and $\bar{E}$ the directed edges for the transitions or actions between them. A successful execution of the task is equivalent to a path in the graph that reaches the goal node. The task graph captures the structure of the task, and the effect of each action. However, generating this graph for an unseen task is extremely challenging, as each unseen state would be mapped to a new node. This is especially the case in visual tasks, where the state space is high dimensional. We thus work with the *conjugate task graph*(CTG) [16], $G = \{V, E\}$, where the actions are now the nodes V, and the states become edges E, which implicitly encode the preconditions of the actions. This allows us to bypass explicit state modeling, while still being able to perform tasks by traversing the conjugate task graph.

We assume that all actions are observed during training from the seen tasks, which is reasonable for tasks in the same domain. This gives all the nodes in CTG, and the goal is to infer the correct edges. This can be viewed as understanding the preconditions for each action. We propose two steps for generating the edges: (i) *Demo Interpretation*: First we obtain a valid path traversing the conjugate task graph by observing the action order in the demonstration; (ii) *Graph Completion*: The second step is to add the edges that are not observed in the demonstration. There might be actions whose order can be permutated without affecting the final outcome. As we only have a single demonstration, this interchangeability is not captured in the previous step. We learn a Graph Completion Network, which adds more edges that are proper given the edges initialized by step (i).

Demo Interpreter. Given $d = [o_1, \dots, o_T]$, our goal is to output $A = [a_1, \dots, a_K]$, the sequence of the actions executed in the demonstration as the initial edges in the CTG as shown in Figure 4. The visual observations o_t are first encoded by a CNN as $Enc(o_t)$. We then adapt a seq2seq model [28] as our demo interpreter to take $Enc(o_t)$ as inputs and generate A. We do not use a frame-based classifier, as we do not need accurate per-frame action classification. What is critical here is that the sequence of actions A provides reasonable initial action order constraints (edges) to our conjugate task graph. We do assume the training demonstrations in $\mathbb{T}_{seen}$ come with the action sequence A as supervision for our demo interpreter. We only require this "flat" supervision for $\mathbb{T}_{seen}$, as opposed to the strong hierarchical supervision used in the previous work [41].

Graph Completion Network (GCN). Given a valid path (action sequence) from the demo interpreter, the goal is to complete the edges that are not observed in the demo. We formulate this as learning graph state transitions [19, 22]. Our GCN iterates between two steps: (i) edge update and (ii) propagation. Given the node embedding $NE_{gcn}(n_i)$ for each node n_i, the edge strengths are updated as:

$$\mathcal{C}_{ij}^{t+1} = (1 - \mathcal{C}_{ij}^t) \cdot f_{set}(N_i^t, N_j^t) + \mathcal{C}_{ij}^t \cdot f_{reset}(N_i^t, N_j^t), \quad (1)$$

where $\mathcal{C}_{ij}^t$ is the adjacency matrix of the previous iteration, f_{set} and f_{reset} are MLPs for setting and resetting the edge, and $N_i = NE_{gcn}(n_i)$ is the node embedding for node i. Given $\mathcal{C}^t$ and the current node embeddings N^t, the propagation step updates the node embeddings with:

$$N_i^{t+1} = rnn(a_i, N_i^t), a_i = \sum_j \mathcal{C}_{ij}^t f_f(N_j^t) + \mathcal{C}_{ji}^t f_b(N_j^t), \quad (2)$$

where $rnn(a_i, N_i^t)$ takes the message a_i from other nodes as input and updates the hidden state N_i^t to N_i^{t+1}.

4.2. Neural Task Graph Execution

We have discussed how the NTG generates a CTG as the compositional representation of a task demonstration. Next we show how to instantiate a policy from this task graph. We propose the NTG *execution engine* that interacts with the environment by executing the task graph. The execution engine executes a task graph in two steps: (i) Node Localization: The execution engine first localizes the current node in the graph based on the visual observation. (ii) Edge Classification: For a given node, there can be multiple outgoing edges for transitions to different actions. The edge classifier checks the (latent) preconditions of each possible next action and picks the most fitting one. These two steps enable the execution engine to use the generated Conjugate Task Graph as a reactive policy which completes the task given observations. Formally, we decompose this policy as:

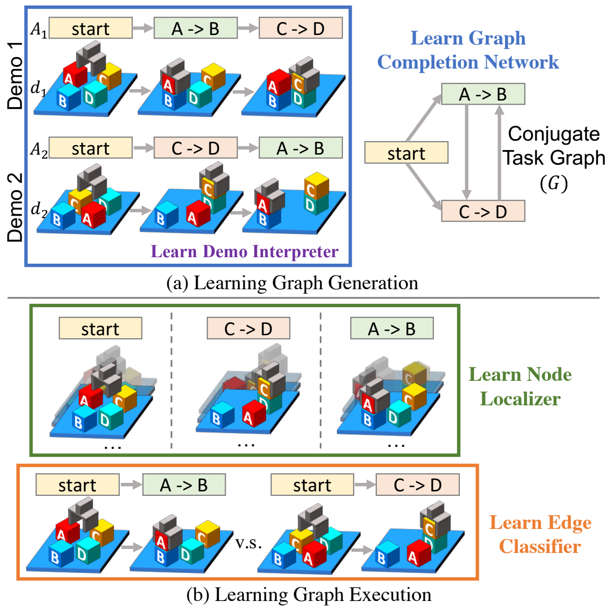


Figure 4. Illustration of our learning setting with a block stacking task. The video demonstrations d_i in the seen tasks only require corresponding action sequence A_i. We aggregate data from all the demonstrations in the same task and use it as the supervision of each component of our model. This approach allows us to bypass the need for strong supervision as in previous works.

$\pi(a|o) \propto \epsilon(a|n, o)\ell(n|o)$, where the localizer $\ell(n|o)$ localizes the current node n based on visual observation o, and the edge classifier $\epsilon(a|n, o)$ classifies which edge transition from n and o. Deciding the edge transition given the node is equivalent to selecting the next action a.

Node Localizer. We define the localizer as: $\ell(n|o) \propto Enc(o)^T NE_{loc}(n)$, where the probability of a node is proportional to the inner product between $Enc(o)$, the encoded visual observation, and $NE_{loc}(n)$, the node embedding of the node. Since our nodes are actions that are already observed in the seen tasks, we can learn the node embeddings effectively. This shows the benefit of modularizing our policy, where sub-modules are more generalizable.

Edge Classifier. The edge classifier is the key for NTG to generalize to unseen tasks. Unlike the localizer, which is approx. invariant across seen and unseen tasks, deciding the correct edge requires the edge classifier to correctly infer the underlying states from the visual observations. Take block stacking as an example. For a task that aims to stack blocks A, B, and C in order, the robot should not pick-and-place C unless B is already *on A*. The edge classifier thus needs to recognize such prerequisites for actions involving block C.

$$\epsilon(a|n, o) \propto (W_\epsilon[Enc(o), NE_{gcn}(n)])^T NE_{loc}(n_a), \quad (3)$$

where n_a is the node for action a, and $NE_{gcn}(\cdot)$ is the final node embedding from our GCN in Section 4.1. As the GCN node embedding is used to generate edges in the conjugate

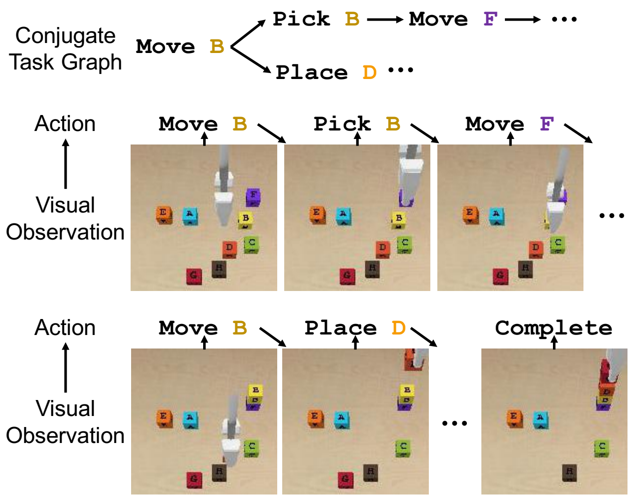


Figure 5. Execution of NTG based on the conjugate task graph. Although the execution engine visited the `(Move B)` node twice, it is able to correctly decide the next action using the edge classifier by understanding the second visit needs to `(Place D)`.

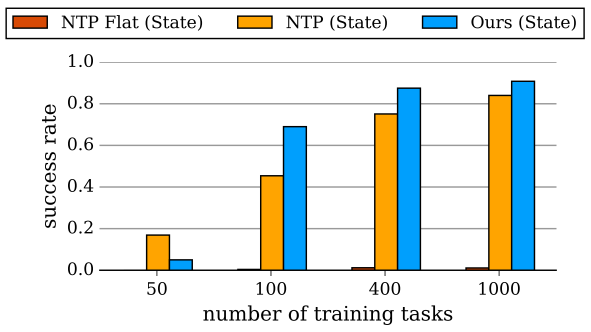
task graph, it captures the task structure. We use NE_{loc} from localization for the destination node.

4.3. Learning NTG Networks

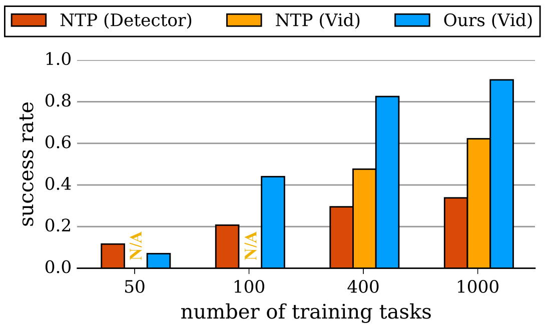
We have described how we decompose $\phi(\cdot)$ into the generator and the execution engine. As discussed in Section 3 we train both on $\mathbb{T}_{seen}$. In contrast to previous works that require strong supervision on $\mathbb{T}_{seen}$ (state-action pairs [8] or hierarchical supervision [41]), NTG only requires the raw visual observation along with the flat action sequence (lowest level program in [41] without a manually defined action hierarchy). An overview of learning different components of NTG is shown in Figure 4.

Learning Graph Generation. For each demonstration d_i^τ of task τ, we have the corresponding $A_i^\tau = [a_1, \ldots, a_K]$, the executed actions. First, we translate A_i to a path $\{P_i^\tau = (\tilde{V}, \tilde{E}_i^\tau)\}$ by using all actions as nodes $\tilde{V}$ and adding edges of the transitions in A_i to $\tilde{E}_i$. For a single task τ, we use the union of all demonstrated paths of τ as the edges $E_t = \bigcup_i \tilde{E}_i^\tau$ of the groud truth conjugate task graph $g_\tau = (V, E_t)$. In this case, the goal of GCN is to transform each P_i^τ to g_τ by completing the missing edges in P_i^τ. We use the binary cross entropy loss following [19] to train the GCN, where the input is P_i^τ and the goal is to generate g_τ.

Learning Graph Execution. Given a task graph from the generator, we learn an execution engine that derives the policy. As discussed in Section 4.2, we decompose the policy into node localizer and edge classifier. For the localizer, we use the video frames as input and the corresponding action labels from the demonstrations as targets. For the edge classifier, we collect all pairs of source-target nodes connected by transitions, and use the action label from the demonstration as the target. Additionally, the edge classifier uses the node embedding from our Graph Completion Network. The idea is that the embedding from the GCN can inform the edge classifier about what kind of visual state it should classify and learn to generalize to the unseen task.



(a) Block Stacking Results with Full State



(b) Block Stacking Results with Visual State

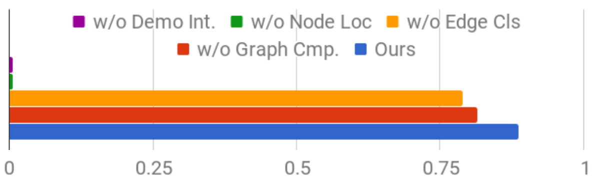
Figure 6. Results for generalizing block stacking to unseen target configuration. (a) Results with the block locations as input, and (b) Results with raw video as input. Our NTG model significantly outperforms the baselines despite using only flat supervision.

5. Experiments

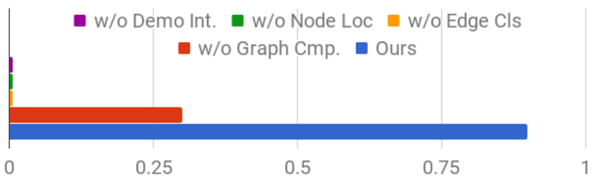
Our experiments aim to answer the following questions: (1) With a single *video* demonstration, how does NTG generalize to unseen tasks and compare to baselines without using compositionality? (2) How do each of the components of NTG contributes to its performance? (3) Is NTG applicable to real-world data? For the first two questions, we evaluate and perform ablation study of NTG in two challenging task domains: the Block Stacking [41] using the BulletPhysics [6] and the Object Collection task in the AI2-THOR [46]. For the last question, we evaluate NTG on real-world surgical data and examine its graph prediction and evaluation of unseen tasks on the JIGSAWS [12] dataset.

5.1. Evaluating Block Stacking in BulletPhysics

We evaluate NTG's generalization to unseen target configurations. The hierarchical structure of block stacking provides a large number of unique tasks and is ideal for ana-



(a) Block Stacking Ablation Study



(b) Object Sorting Ablation Study

Figure 7. Ablation study of NTG. (a) Demo Int. and Node Loc. are almost indispensable. (b) Both GCN and Edge Cls are required to generalize to execution order different from the demonstration.

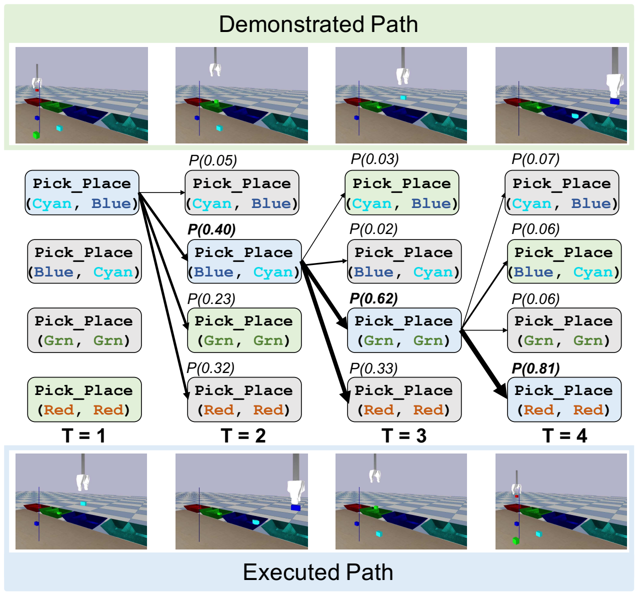


Figure 8. Using GCN, our policy is able to solve an unseen sorting task in a different order than the provided demonstration.

lyzing the effect of explicitly introducing compositionality.

Experimental Setup. The goal of Block Stacking is to stack the blocks into a target configuration. We follow the setup in Xu *et al.* [41]. We use eight 5 cm cubes with different colors and lettered IDs. A task is considered successful if the end configuration matches the task demonstration. We use the 2000 distinct Block Stacking tasks and follow the training/testing split of Xu *et al.* [41].

Baselines. We compare to the following models:

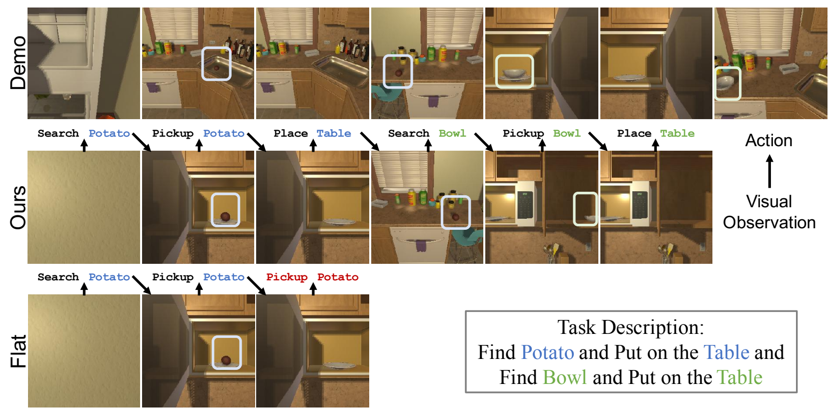
- *Neural Task Programming (NTP)* [41] learns to synthesize policy from demonstration by decomposing a demonstration recursively. In contrast to ours, NTP assumes strong structural supervision: both the program hierarchy and the demonstration decomposition are required at training. We use NTP as an example of methods that encourage compositionality via strong structural supervision.
- *NTP Flat* is an ablation of NTP, which only uses the same supervision as our NTG model (lowest level program).
- *NTP (Detector)* first detects the block and feeds that into the model as the approximated full state. The detector is trained separately with additional supervision.

Results. Results are shown in Figure 6. The x-axis is the number of training seen tasks. We compare models with full state (State) and visual state (Vid) as input. Full state uses the 3D block location, and the visual state uses 64×64 RGB frames. For both input modalities, NTG can capture the structure of the tasks and generalize better to unseen target configuration compared to the baseline. NTG with raw visual input (Ours (Vid)) performs on-par with NTP using full state (NTP (State)). When there is not enough training data (50 tasks), the NTP (State) and NTP (Detector) in are able to outperform NTG because of the extra supervision (hierarchical for NTP (State), and detection for NTP (Detector)). However, once NTG is trained with more than 100 tasks, it is able to quickly interpret novel tasks and significantly outperforms the baselines. Figure 5 shows an NTG execution trace. Although the execution engine visited the `(Move B)` node twice, it is able to correctly decide the next action based on the visual observation by interpreting the underlying state from the visual observation.

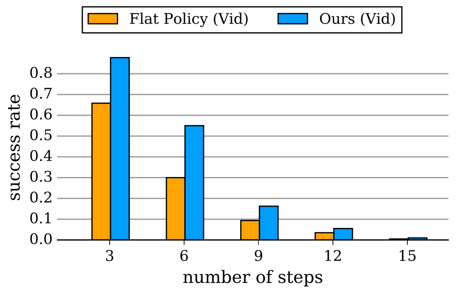
5.2. Ablation Analysis of NTG Model Components

Before evaluating other environments, we analyze the importance of each component of our model. Some subsystems are almost *indispensable*. For example, without the Demo Interpreter, there is no information from the video demonstration, and the policy is no longer task-conditional. We perform the ablation study using 1000 training tasks as follows: For Demo Interpreter, we initialize CTG as a fully connected graph without order constraints from the demonstration. For Node Localizer and Edge Classifier, we replace the corresponding term in the policy $\pi(a|o) \propto \epsilon(a|n,o)\ell(n|o)$ by a constant. For GCN, we skip the graph completion step. As shown in Figure 7(a), the policy cannot complete any of the tasks without Demo Interpreter or Node Localizer. While our full model still performs the best, removing Edge Classifier or GCN does not give as big a performance gap. This is because the Block Stacking tasks from [41] do not all require task structure understanding.

Alternate Solutions for Task. GCN is particularly important for situations requiring alternative execution orders. For example, the task of "putting the red ball into the red bin and the blue ball into the blue bin". It is obvious to us that we can either put the red ball first or the blue ball first.



(a) Example Task Execution Comparing Ours to Flat Policy



(b) Object Collection Results

Figure 9. (a) Object Collection results. The bnding boxes are only for visualization and is not used anywhere in our model. The objects can appear in locations that are different from the demonstration, which leads to challenging and diverse visual state. NTG is able to understand the underlying state (*e.g.* if the object is found) from the visual input and successfully complete the task. (b) Object Collection results on varying numbers of steps. The NTG model is only trained with 6 and 12 steps, and is able to generalize well to other numbers of steps.

This ability to generalize to alternative execution orders is exactly what we aim to capture with GCN. Without GCN, the policy can be easily stuck at unseen execution order (*i.e.,* not understanding object sorting order can be swapped). We thus analyze GCN on the "Object Sorting" task (details in Section VI of [41]), but initialize the scene to require execution order different from the demonstration. These settings will occur often when the policy needs to recover from failure or complete a partially completed tasks. This is challenging because: (i) GCN has to generalize and introduce alternative execution order beyond the demonstration. (ii) Edge Classifier needs to correctly select the action from the newly introduced edges by GCN. As shown in Figure 7(b), the policy cannot complete any of the tasks without Edge Classifier because of the ambiguities in the completed task graph. Figure 8 shows an qualitative example of how our method learns to complete "Object Sorting" with order different from the demonstration using GCN. This shows the importance of both the Edge Classifier and GCN, which are required to complete this challenging task.

5.3. Evaluating Object Collection in AI2-THOR

In this experiment, we evaluate the Object Collection task, in which an agent collects and drop off objects from a wide range of locations with varying visual appearances. We use AI2-THOR [46] as the environment, which allows the agent to navigate and interact with objects via semantic actions (*e.g.* `Open`). This task is more complicated than block stacking because: First, the agent is navigating in the scene and thus can only have partial observations. Second, the photo-realistic simulation enables a variety of visual appearance composition. In order to complete the task, the model needs to understand various appearances of the object and location combinations.

Experimental Setup. An Object Collection task involves visiting M randomly selected searching locations for a set of N target objects out of C categories. Upon picking up a target object, the agent visits and drops off the object at one of K designated drop-off receptacles. A task is considered successful if all of the target objects are placed at their designated receptacles at the end of the task episode. The available semantic actions are `search`, `pickup(object)`, `dropoff(receptacle)`. The `search` action visits each searching locations in a randomized order. `pickup(object)` picks up a selected object and the action would fail if the selected object is not visible to the agent. `dropoff(receptacle)` would teleport the agent to a selected drop-off receptacle (`tabletop`, `cabinet`, etc) and drop off. We use $N = [1, 5]$ objects (3-15 steps) out of $C = 8$ categories, $M = N + 3$ search locations, and $K = 5$ drop-off receptacles.

Baseline. We compare to the "Flat Policy" baseline in [8] to show the importance of incorporating compositionality to the policy. At each step, the Flat Policy uses attention to extract relevant information from the demonstration and combine it with the observation to decide action. For a fair comparison, we implement the Flat Policy using the same architecture as our demo interpreter. Note that the Object Collection domain doesn't have hand-designed hierarchy. Hence NTP [41] is reduced to a similar flat policy model.

Results. The results for Object Collection are shown in Figure 9(b). The models are only trained on 2 and 4 objects and generalize to 1, 3, 5 objects. NTG significantly outperforms the Flat Policy on all numbers of objects. This shows the importance of explicitly incorporating compositionality. Qualitative comparison is shown in Figure 9(a). The bounding boxes are for visualization only and are not used in the model. During evaluation, the objects of interest can appear

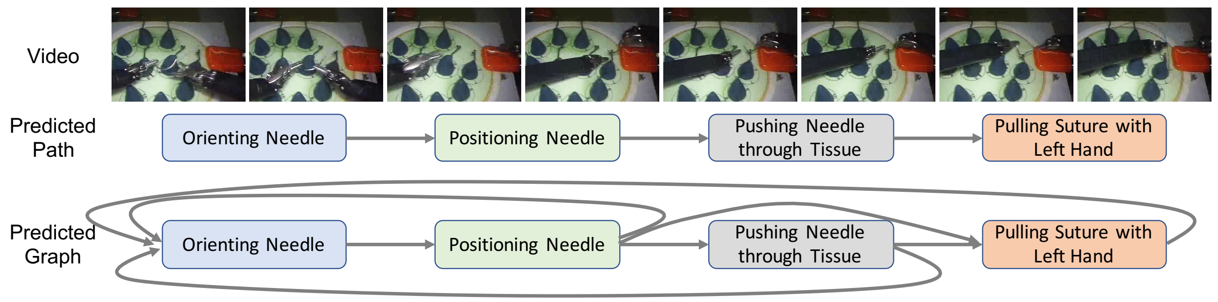

Video
Predicted Path
Orienting Needle
Positioning Needle
Pushing Needle through Tissue
Pulling Suture with Left Hand
Predicted Graph
Orienting Needle
Positioning Needle
Pushing Needle through Tissue
Pulling Suture with Left Hand

Figure 10. Part of a predicted graph from a single demonstration of an unseen task on the JIGSAWS dataset. Our method is able to learn that for the Needle Passing task, after failing any of the step in this subtask, the agent should restart by reorienting the needle.

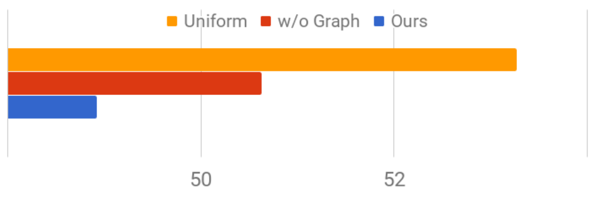


Figure 11. Negative loglikehood (NLL) of expert demonstrations on the JIGSAWS dataset. The policy generated by our full model can best capture the actions performed in human demonstration.

in locations that are different from the demonstration and thus lead to diverse and challenging visual appearances. It is thus important to understand the structure of the demonstration instead of naive appearance matching. Our explicit model of the task structure sets NTG apart from the flat policy and leads to stronger generalization to unseen tasks.

5.4. Evaluating Real-world Surgical Data

We have shown that NTG significantly improves one-shot visual imitation learning by explicitly incorporating compositionality. We now evaluate if this structural approach can be extended to the challenging real-world surgical data from the JIGSAWS dataset [12], which contains videos and states for surgical tasks, and the associated atomic action labeling. In this setting, our goal is to assess NTG's ability to generalize to the task of "Needle Passing", after training on the tasks of "Knot Tying" and "Suturing". This is especially challenging because it requires generalization to a new task with significant structural and visual differences, given only 2 task types for training.

Without a surgical environment, we cannot directly evaluate the policy learned by NTG on the JIGSAWS dataset. Therefore, we evaluate how well the NTG policy is able to predict what a human will do in other demonstrations. This entails generating a policy conditioned on a single demonstration of "Needle passing", and using it to evaluate the negative log-likelihood (NLL) of all the other demonstrations in the "Needle Passing" task. A lower negative log likelihood corresponds to the generated policy better explaining the other demonstrations, and in turn better capturing the task structure.

The results are shown in Figure 11. We compare to the no graph variant of our model and also the lower bound of a uniform policy. Unsurprisingly, the *uniform* policy performs the worst without capturing anything from the demonstration. The *no-graph* variant is able to capture some parts of the expert policy and better capture the expert demonstration. However, the policy generated by full NTG model substantially improves the NLL and is the most consistent with the expert demonstration.

In addition, we show qualitative results of part of our task graph prediction on the JIGSAWS dataset in Figure 10. Again, we train on "Knot Tying" and "Suturing" and evaluate on "Needle Passing". By comparing the predicted path and the final predicted graph, we can see that our model is able to introduce several new edges going back to the action "Orienting Needle". This captures the behavior that when the execution fails in any of step in this subtask of "Needle Passing", the agent should return to "Orienting Needle" and reorient the needle to restart the subtask. This is consistent with our intuition and the ground truth graph.

6. Conclusion

We presented Neural Task Graph (NTG) Network, a one-shot visual imitation learning method that explicitly incorporates task compositionality into both the intermediate task representation and the policy. Our novel Conjugate Task Graph (CTG) formulation effectively handles unseen visual states and serves as a reactive and executable policy. We demonstrate that NTG is able to outperform both methods with unstructured representation [8], and methods with a hand-designed hierarchical structure [41] on a diverse set of tasks, including simulated environment with photo-realistic rendering and a real-world dataset.

Acknowledgements. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. This research was also sponsored in part by the NSF graduate research fellowship.

References

[1] Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L Thomaz. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4(4):343–355, 2012.

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *CVPR*, 2016.

[3] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *ICML*, 2017.

[4] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Deep compositional question answering with neural module networks. In *CVPR*, 2016.

[5] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 1986.

[6] Erwin Coumans and Yunfei Bai. pybullet, a python module for physics simulation, games, robotics and machine learning. `http://pybullet.org/`, 2016–2017.

[7] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning Modular Neural Network Policies for Multi-Task and Multi-Robot Transfer. *arXiv preprint arXiv:1609.07088*, 2017.

[8] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-Shot Imitation Learning. In *NIPS*, 2017.

[9] Richard E Fikes, Peter E Hart, and Nils J Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 1972.

[10] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[11] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *CoRL*, 2017.

[12] Yixin Gao, S. Swaroop Vedula, Carol E. Reiley, Narges Ahmidi, Balakrishnan Varadarajan, Henry C. Lin, Lingling Tao, Luca Zappella, Benjamn Béjar, David D. Yuh, Chi Chiung Grace Chen, René Vidal, Sanjeev Khudanpur, and Gregory D. Hager. Jhu-isi gesture and skill assessment working set (jigsaws) : A surgical activity dataset for human motion modeling. 2014.

[13] Wonjoon Goo and Scott Niekum. One-shot learning of multi-step tasks from observation via activity localization in auxiliary video. *arXiv preprint arXiv:1806.11244*, 2018.

[14] Abhinav Gupta, Praveen Srinivasan, Jianbo Shi, and Larry S Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009.

[15] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017.

[16] Bradley Hayes and Brian Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *ICRA*, 2016.

[17] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *ICCV*, 2017.

[18] De-An Huang, Shyamal Buch, Lucio Dery, Animesh Garg, Li Fei-Fei, and Juan Carlos Niebles. Finding it: Weakly-supervised reference-aware visual grounding in instructional videos. CVPR, 2018.

[19] Daniel D Johnson. Learning graphical state transitions. In *ICLR*, 2017.

[20] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *ICCV*, 2017.

[21] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *ICRA*, 2011.

[22] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[23] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[24] Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. Visual coreference resolution in visual dialog using neural module networks. In *ECCV*, 2018.

[25] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *ICCV*, 2017.

[26] Changsong Liu, Shaohua Yang, Sari Saba-Sadiya, Nishant Shukla, Yunzhong He, Song-Chun Zhu, and Joyce Chai. Jointly learning grounded task structures from language instruction and visual demonstration. In *EMNLP*, 2016.

[27] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *ICRA*, 2018.

[28] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

[29] Dana Nau, Yue Cao, Amnon Lotem, and Hector Munoz-Avila. Shop: Simple hierarchical ordered planner. In *IJCAI*, 1999.

[30] Monica N Nicolescu and Maja J Matarić. A hierarchical architecture for behavior-based robots. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 227–233, 2002.

[31] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *IROS*, 2012.

[32] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016.

[33] Earl D Sacerdoti. A structure for plans and behavior. Technical report, SRI International's Artificial Intelligence Center, 1975.

[34] Ozan Sener, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised semantic parsing of video collections. In *ICCV*, 2015.

[35] Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. *arXiv preprint arXiv:1704.06888*, 2017.

[36] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *ICRA*, 2014.

[37] Makarand Tapaswi, Martin Bauml, and Rainer Stiefelhagen. Book2movie: Aligning video scenes with book chapters. In *CVPR*, 2015.

[38] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *ICLR*, 2018.

[39] David Whitney, Eric Rosen, Elizabeth Phillips, George Konidaris, and Stefanie Tellex. Comparing Robot Grasping Teleoperation across Desktop and Virtual Reality with ROS Reality. In *International Symposium on Robotics Research*, 2017.

[40] Yan Wu and Yiannis Demiris. Towards one shot learning by imitation for humanoid robots. In *ICRA*, 2010.

[41] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *ICRA*, 2018.

[42] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *RSS*, 2018.

[43] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *arXiv preprint arXiv:1710.04615*, 2017.

[44] Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. Visual semantic planning using deep successor representations. In *ICCV*, 2017.

[45] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, 2015.

[46] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017.