Transformers for One-Shot Visual Imitation

Sudeep Dasari
Robotics Institute
Carnegie Mellon University, USA
sdasari@andrew.cmu.edu

Abhinav Gupta
Robotics Institute
Carnegie Mellon University, USA
gabhinav@andrew.cmu.edu

Abstract: Humans are able to seamlessly visually imitate others, by inferring their intentions and using past experience to achieve the same end goal. In other words, we can parse complex semantic knowledge from raw video and efficiently translate that into concrete motor control. Is it possible to give a robot this same capability? Prior research in robot imitation learning has created agents which can acquire diverse skills from expert human operators. However, expanding these techniques to work with a single positive example during test time is still an open challenge. Apart from control, the difficulty stems from mismatches between the demonstrator and robot domains. For example, objects may be placed in different locations (e.g. kitchen layouts are different in every house). Additionally, the demonstration may come from an agent with different morphology and physical appearance (e.g. human), so one-to-one action correspondences are not available. This paper investigates techniques which allow robots to partially bridge these domain gaps, using their past experience. A neural network is trained to mimic ground truth robot actions given context video from another agent, and must generalize to unseen task instances when prompted with new videos during test time. We hypothesize that our policy representations must be both context driven and dynamics aware in order to perform these tasks. These assumptions are baked into the neural network using the Transformers attention mechanism and a self-supervised inverse dynamics loss. Finally, we experimentally determine that our method accomplishes a $\sim$ 2x improvement in terms of task success rate over prior baselines in a suite of one-shot manipulation tasks.[1]

Keywords: Representation Learning, One-Shot Visual Imitation

1 Introduction

Imitation is one of the most important cornerstones of intelligence. Watching other humans act, inferring their intentions, and attempting the same actions in our own home environments allows us to expand our skill set and enhance our representations of the world [1]. On the other hand, robots - while capable of imitating skills like table tennis [2] and driving [3] – are much less flexible when it comes to visual imitation. Most prior work in robotic imitation assumes that the agent is trying to acquire a single skill from demonstration(s) collected kinesthetically [4] (i.e. a human manually guides a robot) or via tele-operation [5]. These approaches can work so long as the target test-time task and environment are do not significantly differ from those seen during training. Is it possible to develop a robotic agent which can learn to imitate without these restrictions?

Visual imitation requires extracting a higher level goal from the visual demonstration and using the inferred goal to predict actions from pixels. But how does one represent goal/intention and how can this contextual information be incorporated into the policy function itself? There are three primary approaches in prior work: the first approach is to represent goals/intentions as pixels by generating goal images, and then inferring actions given current observations and inferred goals [6, 7]. While this approach is intuitive and interpretable, it is difficult to generate pixels, in a way that respects structural differences in the image. Figure 1 shows an example with well defined task semantics, but where a change in object positions makes it difficult to visually map the human state to the

[1]For code and project video please check our website: https://oneshotfeatures.github.io/

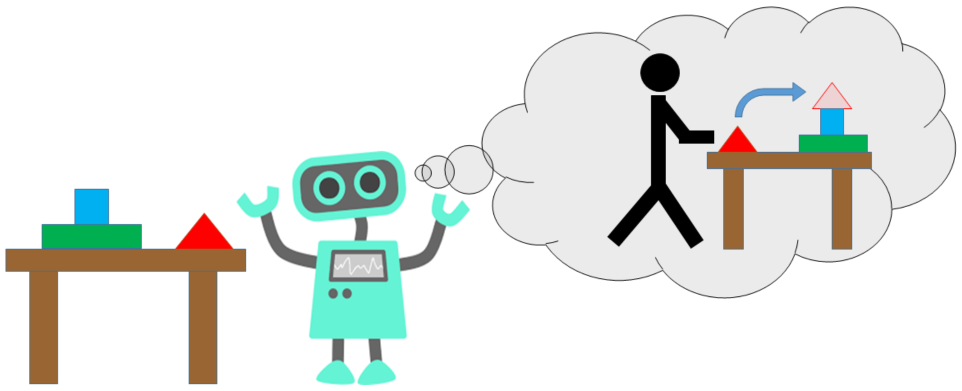
Figure 1: What should the robot do given video from another demonstration agent? A human would immediately know to place the red triangle on the blue square, and can use their past experience to execute the task. Is it possible to teach a robot to do the same?

robot environment. The second approach has been to model visual imitation as a one-shot learning problem [8], which can be solved with meta-learning algorithms [9]. Here, a robot is given a single example, in the form of a video or demonstration (e.g. video + control telemetry), and must use that information to perform new instances of the same task. The demonstration is used to update the parameters of a policy function and the updated policy is executed on the robot. Domain gaps can be addressed with a learned adaptive los function [10]. While the one-shot formalism is very useful, estimating policy parameters from a single example can be an extremely difficult problem and prone to over-fitting.

In this paper, we explore a third alternative: task-driven features for one-shot learning. We process both observations from the target agent and demonstrations frames from a "teacher" agent in order to extract context-conditioned state representations. What neural network architectures can create task-driven features? While in the past, approaches such as LSTMs have been used, in this work, we focus on self-attention architectures. In particular, the Transformers architecture - while simple - has seen broad success in NLP [11] and Vision [12] tasks. Furthermore, using attention for control tasks has has basis in biology and psychology. Indeed, humans use attention mechanisms to create context driven representations [13], and directly supervising policies with human attention can dramatically increase task performance [14].

In this paper, we propose using transformers [11] (or non-local self-attention modules [12]) to extract relational features which act as input state vectors for the policy function. Our transformers take as input both the spatial ResNet Features from teacher demonstration and the target agent. This allows the policy to automatically adapt its features to the task at hand, by using context frames to focus only on important task-specific details. For example, in Figure 1 the robot could use human context frames to focus only on relevant details like the red block's location, and entirely ignore distracting elements like the table's leg. However, transformer features could easily end up improperly weighting important details during test time. We propose to solve this issue by further supervising the state representation learning with an unsupervised inverse dynamics loss. This loss constrains the learning problem and ensures the final representations can model the underlying dynamics, as well as task specific details. Ultimately, our method achieves significant improvements over one-shot imitation learning baselines on a suite of pick and place tasks: our final policies demonstrate a 2x performance gain and can match baseline performance with 3x fewer data-points.

2 Related Work

Learning from Demonstration (LfD) is a rich and diverse field of study which focuses on enabling robots to learn skills from human or other expert demonstrations. A thorough review is out of scope for this paper, so we gladly refer the reader to survey articles [15, 16, 17]. Of prior work, Behavior Cloning (BC) [18, 19], a common formulation of LfD, is most related to our project. BC involves imitating an expert agent given a set of trajectories (a.k.a time series of observations and actions), by fitting a function which approximates the expert's action in a given state. This simple formulae has proven successful in imitating a wide range of behaviors from visual inputs, including robotic manipulation tasks [20] and driving [21]. These methods have been extended to situations where expert observations are present without action labels [22], including prior work which linked this problem to inverse dynamics minimization [23]. However, both of these approaches require the demonstration agent match the imitator.

BC algorithms often assume that they are approximating a single state conditioned policy. In an environment with multiple tasks or multiple variations of the same task, this constraint can be limiting. Work on goal conditioned imitation learning seeks to relax these assumptions by allowing for policies which condition on a goal variable alongside the current state, and adjust their behavior accordingly. There are myriad ways to introduce goal conditioning, including with the robot's state [24], ”goal” images of the final state [25, 26, 27], natural language [28], and video or images of humans [29, 30]. In our project, we assume the robot has a single video of another agent (be it another robot or a human) doing a task, and must complete that same task itself using past experience. This is a specific instance of the one-shot learning problem [8], and has been investigated before previously using meta-learning with an adaptive loss [10]. Instead of using meta-learning, we propose to attack this problem with an attention mechanism over image frames.

A challenge in this line of work is learning visual representations which can enable the robot to deduce the task from video of another agent *and* perform the task itself. Work in computer vision demonstrated that deep neural networks are capable of learning such flexible representations for action recognition [31] and state estimation [32], but often require large image datasets to fully train. Unfortunately, collecting ImageNet [33] scale datasets on robotics platforms is prohibitively expensive, due to the cost of continuous robot operation and hardware fragility. Work in self-supervised learning [34, 35, 36] offers a glimmer of hope, by showing how large and (relatively) cheap sets of unlabelled images can be used to learn expressive and useful representations for other downstream tasks. These representations could be used directly as reward functions [37, 38], but it can be very difficult to define rewards for a suite of tasks. Instead, unsupervised learning techniques alongside simple data augmentation can be used to increase data efficiency when directly acquiring policies with reinforcement learning [39, 40, 41]. Even simpler self-supervised losses - like inverse modelling (i.e. predicting action between two sequential states) - can be used to learn robust policies which adapt to new environments [42]. Our goal in this project is to apply these insights in representation learning to the one-shot imitation learning problem.

3 Our Method

3.1 Problem Definition

Our method follows prior work [9, 10], and formalizes the one-shot imitation learning problem as supervised behavior cloning on a data-set of tasks. For each task $\mathcal{T}$ (e.g. place blue bottle in bin), we have several demonstration videos and target trajectories. Note that the demonstration videos and target trajectories are semantically similar tasks but could have different starting/end states. We represent each demonstration video as v_i and each target trajectory, t_i, as a temporal sequence of observations (o) and actions (a). Hence, $t_i = \{(o_i^{(1)}, a_i^{(1)}), \ldots, (o_i^{(k)}, a_i^{(k)})\}$.

Models are trained on a dataset of tasks $\mathcal{D} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$. During test time, new test tasks - $\mathcal{T}_{test}$ - are sampled which the model must successfully control the imitator agent to perform. Thus, all methods are evaluated on task success rates in held out environments. Our setup is challenging because: (a) morphological differences between demonstration and target agent (e.g. one is human and other is robot arm); (b) missing correspondence between demonstration videos and target trajectories.

3.2 Feature Learning with Transformers

Given video context from a demonstrator agent and image frames from the test environment, our representation module must deduce relevant features and efficiently pass them on to later stages of the pipeline for action prediction. For example, when given a video of a green bottle being dropped in a bin, the vision module should detect and represent the green bottle in its own environment while ignoring other distracting objects. We propose to learn this mechanism end-to-end using self-attention Transformer modules [11], in the hope that this powerful inductive bias helps the policy perform tasks successfully.

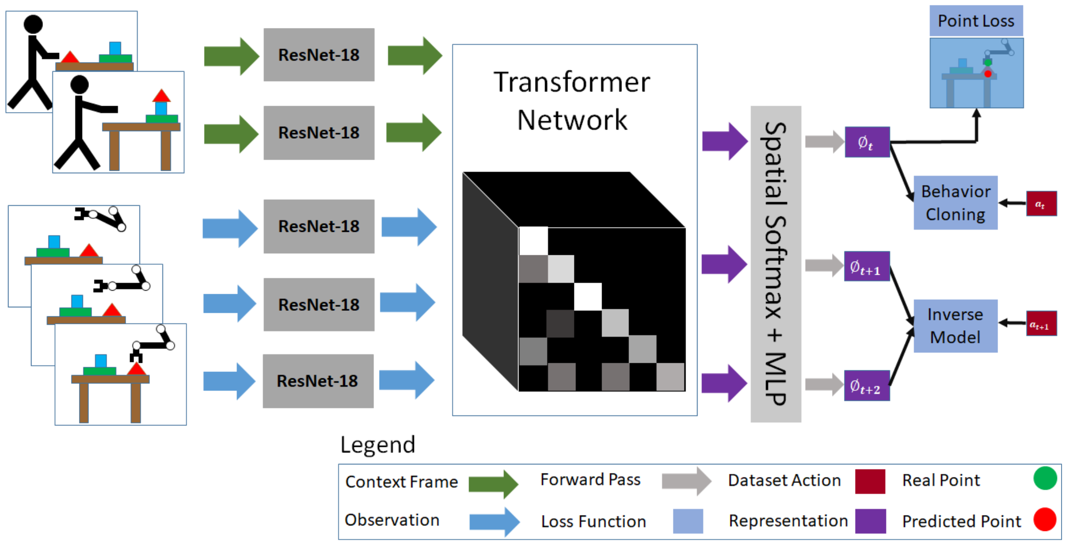

Figure 2: Our method uses a Transformer neural network to create task-specific representations, given context and observation features computed with ResNet-18 (w/ added positional encoding). The attention network is trained end-to-end with a behavior cloning loss, an inverse modelling loss, and an optional point loss supervising the robot's future pixel location in the image.

Before the attention module, individual images from both the context video and current state are passed through a ResNet-18 architecture [43], and spatial features (size $[512, T, H, W]$) are collected before the average pooling step. At this stage, the features are flattened (size $[512, T * H * W]$) and sinusoidal positional encodings [11] are added to the tensor (i.e. time and space treated as single dimension). These embeddings can allow neural networks to represent higher frequency functions [44], and we empirically found that they were crucial to preserving spatial and temporal information in the attention module. After adding positional encodings, the features are reshaped to their original size.

Next, the non-local multi-head attention operator is applied to the input tensor. We adopt a specific implementation of the Transformers self-attention module presented in Wang et al. [12], which we augment with multi-headed self-attention [11]. First, the module generates **K**ey, **Q**uery, and **V**alue tensors by applying three separate 3D spatio-temporal convolutions (we use kernel size $k = 1$) with ReLU activation to the input tensor. To be clear, each convolution layer's input and output are $[d, T, H, W]$ tensors, where d is the Transformer's embedding size. These generated key, query, and value tensors are then flattened and projected down n separate times - once for each attention "head" - before attention is applied (final shape per head $[d, T * H * W]$). The self-attention operator is applied to each head individually. Considering attention head j, temperature parameter τ, and projected tensors K_j, Q_j, V_j, this amounts to:

$$A_j = \textbf{softmax}(K_j^T Q_j / \tau) \qquad V_j^{(out)} = V_j A_j$$

The individual attention heads are then concatenated together channel-wise, and then projected back to the original 512 dimension size with another 3D convolution ($O = \textbf{Conv3D}(\textbf{concat}[V_1^{(out)}, \ldots, V_n^{(out)}])$). Note that this multi-head attention operator can be implemented with little overhead using batched matrix multiplication. Dropout [45], then a residual connection, and finally batch normalization [46] are applied to get the final output $f(x) = \textbf{batchnorm}(x + \textbf{dropout}(O))$, with final size $[512, T, H, W]$. In order to appropriately apply this to behavior cloning (where o_{t+1} is not known during test time), we make this operation causal by appropriately padding the 3D convolution operators and masking the attention.

3.3 Goal Conditioned Behavior Cloning

As discussed previously, our objective is to learn a policy $\pi(a_t | o_{1:t}, v)$ which ingests the current (or optionally all previous) state observations alongside a context video, and predicts a distribution over possible actions the expert policy would select. We process the input video stream with stacked attention modules to yield fixed size spatial features, with one feature map per time-step. The features are projected down to a fixed size representation vector using a spatial softmax operator [47], followed by a multi-layer perceptron with ReLU activations, and finally L2 normalization to unit length. This representation $\phi_t = F(o_{1:T}, v)$ is used for action prediction.

Multi-Modal Action Prediction: One of the most naive ways to predict $\pi(a_t|o_{1:t}, v)$ from ϕ_t is to simply parameterize the policy as a normal distribution $\pi(a_t|o_{1:t}, v) = \mathcal{N}(\mu(\phi_t), \sigma(\phi_t))$, and to sample actions from that. However, this approach can run into severe limitations when the real expert distribution is multi-modal. Consider a robot attempting to top-down lift a cup by its handle. Rotating the gripper by 90°or -90°, but not rotating at all (i.e. the mean action) would result in task failure since the gripper would close on top of the handle. Prior work [20, 48, 26] showed this limitation matters in practice, and rectifies the situation by predicting a mixture of uni-modal distributions. We adopt the same solution used by Lynch et al. [26]. First, we discretize the action space (discussed in detail in Section 4.1) and then parameterize the policy as a discretized logistic mixture distribution [49]. For each timestep, we predict k logistic distributions with separate mean and scale, and form a mixture by convexly weighting them with vector α. The behavior cloning training loss is simply negative log-likelihood for this distribution:

$$\mathcal{L}_{BC}(\mathcal{D}, \theta) = -\ln(\Sigma_{i=0}^{k} \alpha_k(\phi_t)\ P(a_t, \mu_i(\phi_t), \sigma_i(\phi_t))$$

Where, $P(a_t, \mu_i(\phi_t), \sigma_i(\phi_t)) = F(\frac{a_t+0.5-\mu_i(\phi_t)}{\sigma_i(\phi_t)}) - F(\frac{a_t-0.5-\mu_i(\phi_t)}{\sigma_i(\phi_t)})$ and $F(\cdot)$ is the logistic CDF. During test time, actions are simply sampled from the distribution and executed on the robot without rounding. For most of our experiments, the model performed best when using two mixture components and learned constant variance parameters per action dimension.

3.4 Inverse Model Regularizer

Our method also adds a self-supervised inverse modeling objective to act as a regularizer to the behavior cloning loss during training. Context and trajectory snippets are sampled from the dataset, and images in them are randomized with sampled translations, color shifts, and crops. This randomization is applied consistently to frames from the context video, whereas images from the agent's observation stream (a.k.a trajectory images) are randomized *individually*. This randomized image stream is passed through the attention and representation modules to generate $\tilde{\phi}_t$. The representations $\tilde{\phi}_t$ and $\tilde{\phi}_{t+1}$ are used to predict a discretized logistic mixture distribution over intermediate actions. Thus, the inverse loss is:

$$\mathcal{L}_{INV}(\mathcal{D}, \theta) = -\ln(\Sigma_{i=0}^{k} \alpha_k(\tilde{\phi}_t, \tilde{\phi}_{t+1})\ \text{logistic}(\mu_i(\tilde{\phi}_t, \tilde{\phi}_{t+1}), \sigma_i(\tilde{\phi}_t, \tilde{\phi}_{t+1})))$$

We share parameters between the behavior cloning and inverse modeling objectives for the attention module, representation module, and distribution prediction heads (i.e. after first layer). In practice, we use the randomized image stream for both tasks as well, in order to minimize memory consumption.

3.5 Point Prediction Auxiliary Loss

Finally, our model uses ϕ_t to predict a 2D keypoint location corresponding to the location of the gripper in the image H timesteps in the future. Ground truth for this auxiliary loss is easy to acquire given either a calibrated camera matrix or object detector trained on the robot gripper. One could instead predict the 3D gripper position in world coordinates if neither is available. While not strictly needed for control, this loss is very valuable during debugging, since it lets us visually check during training if the model understand where the robot ought to be H timesteps in the future. The point prediction is parameterized with a simple multi-variate 2D normal distribution $\hat{p}_{t+H} \sim \mathcal{N}(\mu(\phi_t), \Sigma(\phi_t))$ with loss $\mathcal{L}_{pnt}(\mathcal{D}, \theta) = -\ln(\text{likelihood}(p_{t+H}, \hat{p}_{t+H}))$. Thus, the overall loss for our method is:

$$\mathcal{L}(\mathcal{D}, \theta) = \lambda_{BC}\ \mathcal{L}_{BC}(\mathcal{D}, \theta) + \lambda_{INV}\ \mathcal{L}_{INV}(\mathcal{D}, \theta) + \lambda_{pnt}\ \mathcal{L}_{pnt}(\mathcal{D}, \theta)$$

4 Experimental Results

Our model is evaluated on robotic manipulation tasks - namely pick and place tasks - in simulation using multi-agent MuJoCo [50] environments. Our evaluations investigate the following questions:

Model	Reaching Success	Picking Success	Placing/Overall Success
Our Method	**99.4%** $\pm 1.2\%$	**92.5%** $\pm 4.1\%$	**88.8%** $\pm 5.0\%$
Contextual-LSTM	$38.8\% \pm 7.6\%$	$26.3\% \pm 6.9\%$	$23.8\% \pm 6.7\%$
DAML [10]	$36.9\% \pm 7.6\%$	$10.6\% \pm 4.8\%$	$6.9\% \pm 4.0\%$
DAML Auxiliary	$47.8\% \pm 7.4\%$	$17.8\% \pm 5.6\%$	$13.3\% \pm 5.0\%$

Table 1: Comparison between our method and baselines in 16 pick and place tasks. Values indicate success rates and 95% confidence intervals for "stages"[2] in the overall pick and place task.

(1) can our model perform new task instances (defined in 4.1) previously unseen during training? And (2) what components (e.g. inverse loss, etc.) are most crucial for successful control?

4.1 Simulation Environment and Tasks

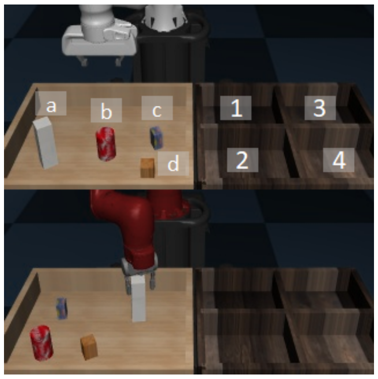

Figure 3: Our base environment is adopted from RoboTurk [51]. The 16 tasks consist of taking an object (a-b) to a bin (1-4). Top robot is agent and bottom is demonstrator.

Environment Description: The environments we use are modified variants of those originally presented in RoboTurk [51]. Visually, the *base environment* - shown in Figure 3 - is the exact same as the original from RoboTurk, except the object meshes are replaced with primitive geometric types (e.g. boxes and cylinders) in order to improve simulation contact stability and run-time. This modification results in only minor visual differences. In order to investigate visual imitation across agent morphology, we use duplicate versions of the environment with two visually distinct robots. The Sawyer robot (red robot in Figure 3) provides demonstration videos and the Panda robot (white robot in Figure 3) acts as the agent which our model must control. Both environment's action spaces are modified to support end-effector control. Given a target x, y, z position, rotation in axis-angle form, and gripper joint angle the environment solves for desired robot joint angles with inverse kinematics and sends joint velocities to the robot using a simple PD controller. Thus, the final action space consists of a target pose discretized into 256 independent bins per dimension in order to support our behavior cloning loss. It's important to note that the demonstrations we train on do not cover the whole state space, so the robot is mostly constrained to 3-DOF movement.

Task Definition: A "task instance" consists of picking an object from a specific start location - uniformly distributed on the table in Fig. 3 - and placing the object in one of the four bins on the right. Task instances are grouped into "tasks" based on shared properties. For example, picking a milk carton (from Fig. 3) and placing it into bin 1 is a task, and different task instances are constructed by changing the carton's start position. This precise definition allows us to collect a suite of train task instances, train models on that data, and test generalization to new task instances.

Data Collection Methodology: Training data is collected using an expert pick-place policy (built using privileged information from the simulator) in the target environment(s). For each task ($\mathcal{T}$) we repeatedly, sample a demonstration video (v_i) by executing the expert policy on the Sawyer robot, then shuffle the objects, and sample an expert trajectory (t_i) by executing the expert policy on the Panda robot. This way a dataset of tasks is formed from individual task instances.

4.2 Baseline Comparisons

Our investigation begins by evaluating our method's performance in 16 tasks in the base environment (Figure 3). We seek to determine the robot's physical competency at manipulating all four objects, as well as its ability to deduce which task it should perform from context video. A natural way to quantify this is by breaking down the 16 pick and place tasks into "reach," "pick," and "place" stages[2], and reporting success rates on each stage individually. Failure modes can be successfully deduced from these rates. For example, since reaching is a physically easy task, if the robot does not reach the object then it is likely unable to deduce the target object from the context frames. Furthermore, if the robot reaches the object but is unable to pick it up, its physical dexterity (or lack thereof) is likely to blame.

We collect 100 train task instances using the methodology described previously for each of the 16 tasks. That amounts to 1600 total demonstration videos alongside 1600 expert robot trajectories. We train our method on the dataset and compare against the following baselines:

- **Contextual-LSTM:** This baseline utilizes a standard Encoder-Decoder LSTM [52, 53] (augmented with self-attention [54, 55]), to first consume the context video, and then predict actions from encoded observations. It uses the same mixture distribution our model uses. Before LSTM processing, images frames are embedded using a pre-trained ResNet-18 [43] neural net combined with spatial-softmax [47] and fully-connected layers. The whole network is trained end-to-end with a behavior cloning loss.
- **Domain Adaptive Meta-Learning:** DAML [10] uses a learned loss function to adapt a neural network's parameters to perform the desired task. We used a wider version of the network used in the original paper, since we found that using deeper models (like ResNet-18) resulted in overfitting on this task. To increase performance, the same discrete logistic action distribution is used. DAML is trained end-to-end with the MAML meta-learning algorithm [56] using a behavior cloning loss, along with explicit supervision of the pick and drop locations.
- **DAML-Auxiliary:** This method uses the same meta-learning model described above, except only the predicted pick and place locations are used during test time. Given this prediction, a grasp motion is executed in the environment using a hard coded grasp policy.

For each of the 16 tasks, the models are prompted to perform new task instances (unseen during training) using freshly generated context videos. Success rates for our method and baselines (averaged across tasks) are shown in Table 1. As you can see, our method is the only one which can reliably perform new task instances. Its overall success rate is double the competing models' reaching success rate, including the DAML-auxiliary model which makes strong task assumptions, and the LSTM model which uses embedding level attention. The LSTM baseline's (which uses standard attention) relative failure supports our hypothesis that the Transformer architecture uniquely enables difficult visual processing. For additional experiments testing generalization to new objects (i.e. new tasks instead of new task instances) refer to Appendix A.1.

4.3 Architecture Ablation

While the our model clearly outperforms the other baselines, it is unclear if the Transformers architecture or additional losses deserve more credit. To test this thoroughly, the Transformers model is tested against an ablated version of itself without the attention mechanism (i.e. just temporal-spatial convolutions) using the same base environment comparison described before. Furthermore, models are trained with various versions of the baseline neural network architectures, alongside the additional loss terms. Specifically, 4 baseline architectures are considered: 2 of them adopt the small convolutional network used in prior work [10, 57] either with or without an additional LSTM [52] on top, and the other 2 use ResNet features [43] (again with or without LSTM). Note all architectures were tuned to maximize *their own* test performance rather than to match some other metric (e.g. number of parameters), since doing so often led to worse results for the baseline (e.g. larger LSTMs overfit more than Transformers). Results are presented in Figure 4. The key takeaways are encouraging. First, the Transformers architecture (w/ attention) outperforms a library of other architectures for this task by large margins, even using the same losses. Furthermore, the baselines perform better when trained with the additional losses compared to being trained purely with a behavior cloning loss as done before (contextual-LSTM's success rate improves $20\% \rightarrow 40\%$).

[2]Reaching is defined as placing the gripper within < 0.03 units from the target object, picking requires stably lifting the object > 0.05 units off ground, and placing requires putting the object in the correct bin (a.k.a task completion)

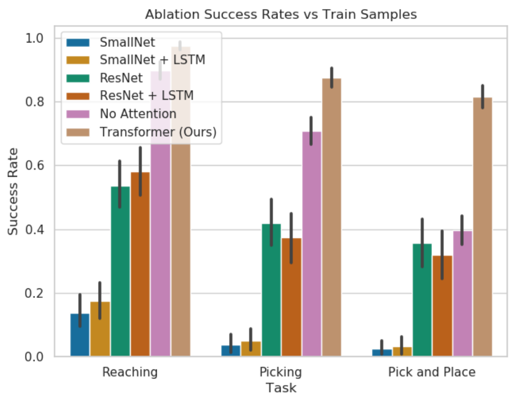

Figure 4: Our Transformer model is compared against other neural networks (all trained w/ our losses and code) to determine how useful the attention mechanism really is. The Transformer architecture outperforms all others, including a version of itself w/out attention.

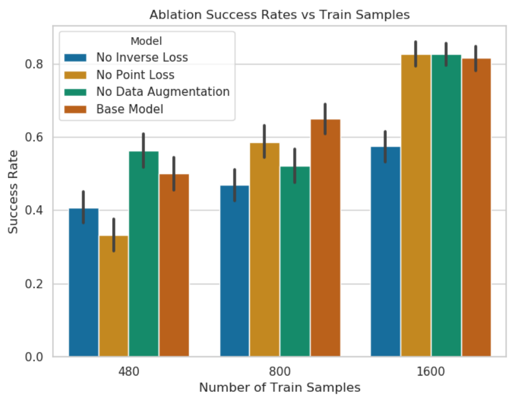

Figure 5: We compute success rate v.s number of train samples for our method and versions with one loss excluded (all w/ Transformer). Note the model without inverse loss is usually outperformed when compared to its peers trained on the same data.

4.4 Loss Function and Method Ablations

Given that our training losses/code boosted baseline architecture performance compared to using just behavior cloning, we now seek to test exactly which component was most useful. It's entirely possible that some of the additional parts offer more utility in the "low-data" regime where overfitting is more likely, and thus are less useful when more data is present. Thus, we collect two more versions of the base environment dataset with fewer samples (480 and 800 samples pairs), and train three ablations - one model without the inverse loss, one without the point loss, and one without data augmentation - alongside our base model on all three datasets (two new sets + original). That results in a total of 12 models, all of which we evaluate in the same manner as before. Overall success rates for all models are in Figure 5. Note that the model without the inverse loss is outperformed by its counterparts in two out of three datasets, whereas the point loss only makes a significant difference in the smallest dataset. Indeed as the number of datapoints increases, so does the importance of the inverse loss: the model without inverse loss is more than 25% worse than its counterparts in the $N = 1600$ case! While the inverse loss clearly makes a difference, this cannot be observed as "positive transfer" in the behavior cloning train/test loss (see Appendix A.2). This suggests inverse loss regularization helps test time performance in ways not captured in the training objective. Finally, conditioning our policy on context video proved to be more effective than just feeding it the last frame, which indicates the demonstration helps our model determine which task to perform compared to using a "goal image" frame. For more check Appendix A.3.

5 Discussion

In this project we explore the one-shot visual imitation learning problem. Our experiments highlight two technical contributions - applying the Transformers architecture to one-shot imitation tasks and a self-supervised inverse modelling objective - which both result in large performance gains over baseline one-shot imitation learning approaches. More specifically, our ablations show that our model trained without the self-supervised inverse loss performs significantly worse when compared to other versions with the inverse loss, and all of our Tansformers models (even without inverse loss) outperform a Seq2Seq LSTM trained with traditional "embedding level" attention mechanisms by roughly 2x.

The main takeaway here is that injecting the right biases - both in terms of network design and the loss function - can help policies perform better during test-time. We believe that the Transformer's attention mechanism provides such a bias by allowing for task conditioned representations, whereas the inverse model forces the policy to preserve information which is needed for robust control during test time. We hope that these findings prove useful to others working on one-shot imitation learning and goal conditioned reinforcement learning in general.

Acknowledgments

We'd like to begin by acknowledging the students and collaborators at CMU who gave valuable feedback which made the final paper much stronger. In particular, we'd like to recognize Sam Powers, Devendra Chaplot, Kenneth Marino, Adithya Murali, and Shubham Tulsiani. Finally, this research was funded by ONR MURI, the ONR Young Investigator Award to Abhinav Gupta and the DAPRA MCS award.

References

[1] S. Vogt and R. Thomaschke. From visuo-motor interactions to imitation learning: behavioural and brain imaging studies. *Journal of Sports Sciences*, 25(5):497–517, 2007.

[2] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[3] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[4] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 365–371. IEEE, 2011.

[5] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[6] P. Sharma, D. Pathak, and A. Gupta. Third-person visual imitation learning via decoupled hierarchical controller. In *Advances in Neural Information Processing Systems*, pages 2597–2607, 2019.

[7] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv:1912.04443*, 2019.

[8] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[9] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368, 2017.

[10] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[12] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[13] C. A. Rothkopf, D. H. Ballard, and M. M. Hayhoe. Task and context determine where you look. *Journal of vision*, 7(14):16–16, 2007.

[14] R. Zhang, Z. Liu, L. Zhang, J. A. Whritner, K. S. Muller, M. M. Hayhoe, and D. H. Ballard. Agil: Learning attention from human for visuomotor tasks. In *Proceedings of the european conference on computer vision (eccv)*, pages 663–679, 2018.

[15] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[16] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 59(BOOK_CHAP), 2008.

[17] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[18] S. Ross, G. Gordon, and D. Bagnell. 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*, pages 627–635, 2011.

[19] M. Bain. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[20] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[21] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[22] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[23] C. Yang, X. Ma, W. Huang, F. Sun, H. Liu, J. Huang, and C. Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In *Advances in Neural Information Processing Systems*, pages 239–249, 2019.

[24] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp. Goal-conditioned imitation learning. In *Advances in Neural Information Processing Systems*, pages 15324–15335, 2019.

[25] A. Mandlekar, F. Ramos, B. Boots, L. Fei-Fei, A. Garg, and D. Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. *arXiv preprint arXiv:1911.05321*, 2019.

[26] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132, 2020.

[27] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Advances in Neural Information Processing Systems*, pages 8538–8547, 2018.

[28] C. Lynch and P. Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[29] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2020.

[30] A. Xie, F. Ebert, S. Levine, and C. Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *arXiv preprint arXiv:1904.05538*, 2019.

[31] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[32] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield. Camera-to-robot pose estimation from a single image. *arXiv preprint arXiv:1911.09231*, 2019.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[35] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[36] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[37] P. Sermanet, K. Xu, and S. Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

[38] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[39] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[40] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[41] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[42] N. Hansen, Y. Sun, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[46] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[47] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[48] R. Rahmatizadeh, P. Abolghasemi, A. Behal, and L. Bölöni. From virtual demonstration to real-world manipulation using lstm and mdn. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[49] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[50] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[51] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. *arXiv preprint arXiv:1811.02790*, 2018.

[52] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
[53] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
[54] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
[55] J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.
[56] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.
[57] A. Zhou, E. Jang, D. Kappler, A. Herzog, M. Khansari, P. Wohlhart, Y. Bai, M. Kalakrishnan, S. Levine, and C. Finn. Watch, try, learn: Meta-learning from demonstrations and reward. *arXiv preprint arXiv:1906.03352*, 2019.

A Appendix

A.1 Baseline Comparisons: Multi-Object Environments

While the prior experiments showed our model could successfully generalize to new task instances, can it also generalize to new tasks including unseen objects? To answer this question the baseline comparisons (described in Section 4.2) are repeated in environments with multiple objects. Importantly, the objects used during test time are unseen during training.

Environment Description: The *multi-object environment* is cloned from the base environment (presented in Section 4.1) and modified to include more objects with different shapes and textures. Note that while object appearance and shape is randomized, dynamical properties - like friction - are kept constant since they cannot be visually judged. The simulator has 30 unique objects, 26 of which are seen during training and 4 are only used during test time.

Data Collection Process: To collect train tasks, 4 objects are sampled from the 26 train objects, which results in an environment with 16 tasks. For each task, multiple task instances composed of expert demonstration videos (v_i) and imitator trajectories (t_i) are collected using the same methodology as before (refer to Section 4.2 and Section 4.1). In total, the train dataset is composed of 1200 tasks (2400 task instances total). Test tasks are also sampled in the same fashion as before, except using the 4 new objects. Our method is able to succeed at the object picking stage of the tasks $50 \pm 9.9\%$ of the time which is $\sim$ 2x better than the best baseline (contextual-LSTM) which only picks $23 \pm 8.4\%$ of the time. Unfortunately, all methods (including ours) often place objects in the wrong bin resulting in final success rates of $23 \pm 8.4\%$ for our method and $22 \pm 8.3\%$ for the best baseline. In practice, this failure mode is easy to rectify since a hard coded policy will always place the object in the right bin. Encouragingly, our policy is best at grasping and picking unseen objects which is the *hardest* part of this task. Nonetheless, this failure mode shows more improvements are needed for this method to work in broader settings.

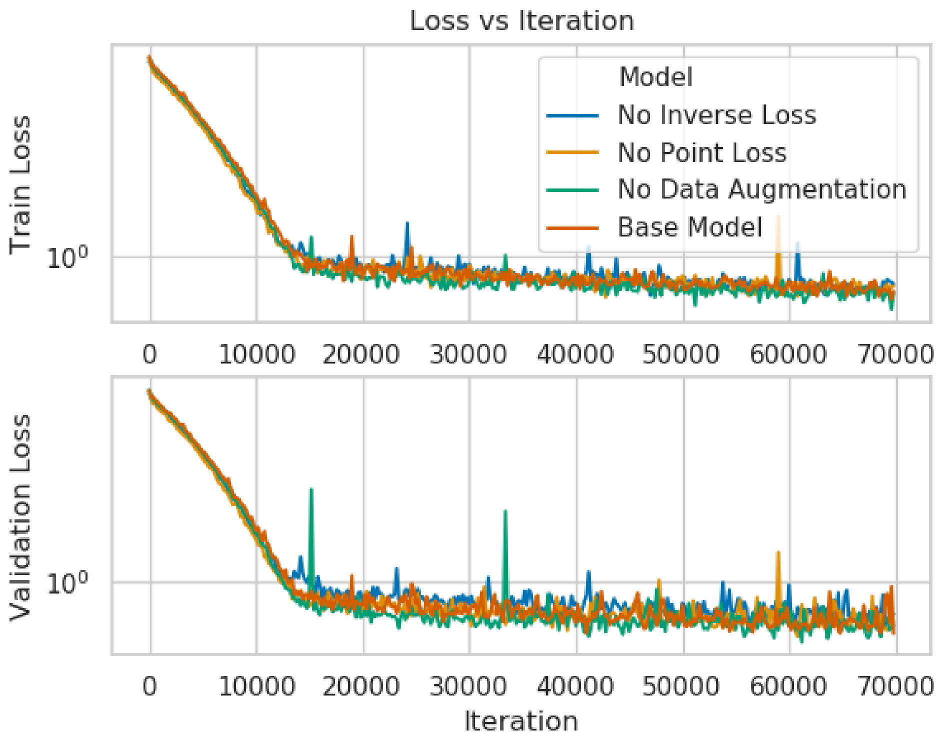

Figure 6: One hypothesis is that the ablated models fail at test time because they cannot optimize the behavior cloning loss. Comparing train and val loss for models trained on the same data (N=1600) eliminates this possibility.

A.2 Regularization Effect on Behavior Cloning Loss

While the inverse model regularization term clearly changed test time performance for the better (shown in Section 4.4), can this be explained by positive transfer to the behavior cloning task? In other words, it is possible the inverse modelling loss merely prevents over-fitting in the behavior cloning loss, and thus some other regularization term could achieve the same effect.

To test this theory, we plot behavior cloning loss (both training and validation) vs train iteration for both the base model, and ablation models from Section 4.4. Note that behavior cloning train performance is nearly identical, whereas final success rates are dramatically different. We believe these facts in tandem confirm that self-supervised inverse modeling forces our representation to capture information which is useful for robust test performance, but not necessary to minimize the cloning loss.

A.3 Time-Step Ablation

Instead of using a context video from the demonstrator agent to infer the task, our model could just use the last frame from the demonstration video. After all, the last frame should uniquely specify which object should go in which bin, and prior work [27] has successfully used goal image conditioning. To test this, we train a version of our model which conditions just on the final frame from the context video, and compare its performance on the benchmarks from Section 4.2. This modified model achieves a final success rate of $61 \pm 9.7\%$ which is significantly less than the $88 \pm 5.0\%$ our model (which ingests more frames from context) can achieve. This effect holds even if the base model uses just one extra context frames (i.e. both beginning and end frame). We hypothesize that these frames, while not strictly necessary, help the infer which task it needs to perform, thus resulting in a performance boost.