Time-Contrastive Networks: 
Self-Supervised Learning from Pixels

Pierre Sermanet∗† Corey Lynch∗†R Yevgen Chebotar∗‡
Jasmine Hsu† Eric Jang† Stefan Schaal‡ Sergey Levine†

†Google Brain, Mountain View, CA, USA
‡University of Southern California, Los Angeles, CA, USA
{sermanet, coreylynch, hellojas, ejang, slevine}@google.com
{ychebota, sschaal}@usc.edu

Abstract

We propose a self-supervised approach for learning representations and robotic behaviors entirely from unlabeled videos recorded from multiple viewpoints, and study how this representation can be used in two robotic imitation settings: imitating object interactions from videos of humans, and imitating human poses. Imitation of human behavior requires a viewpoint-invariant representation that captures the relationships between end-effectors (hands or robot grippers) and the environment, object attributes, and body pose. We train our representations using a triplet loss, where multiple simultaneous viewpoints of the same observation are attracted in the embedding space, while being repelled from temporal neighbors which are often visually similar but functionally different. This signal causes our model to discover attributes that do not change across viewpoint, but do change across time, while ignoring nuisance variables such as occlusions, motion blur, lighting and background. We demonstrate that this representation can be used by a robot to directly mimic human poses without an explicit correspondence, and that it can be used as a reward function within a reinforcement learning algorithm. While representations are learned from an unlabeled collection of task-related videos, robot behaviors such as poring are learned by watching a single 3rd-person demonstration by a human. Reward functions obtained by following the human demonstrations under the learned representation enable efficient reinforcement learning that is practical for real-world robotic systems. Video results, open-source code and dataset are available at sermanet.github.io/imitate

1 Introduction

While supervised learning has been successful on a range of tasks where labels can be easily specified by humans, such as object classification, many problems that arise in interactive applications like robotics are exceptionally difficult to supervise. For example, it would be impractical to label every aspect of a pouring task in enough detail to allow a robot to understand all the task-relevant properties. Pouring demonstrations could vary in terms of background, containers, and viewpoint, and there could be many salient attributes in each frame, e.g. whether or not a hand is contacting a container, the tilt of the container, or the amount of liquid currently in the target vessel or its viscosity. Ideally, robots in the real world would be capable of two things: learning the relevant attributes of an object interaction task purely from observation, and understanding how human poses and object interactions can be mapped onto the robot, in order to imitate directly from third-person video observations.

In this work, we take a step toward addressing these challenges simultaneously through the use of self-supervision and multi-viewpoint representation learning. We obtain the learning signal from

∗Equal contribution
RGoogle Brain Residency program (g.co/brainresidency)

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
unlabeled multi-viewpoint videos of interaction scenarios, as illustrated in Figure 1 (left). By learning from multiview-video, the learned representations effectively disentangle functional attributes like pose from viewpoint and agent. We then show how the robot can learn to link this visual representation to a corresponding motor command using either reinforcement learning or self-regression, effectively learning new tasks by observing humans.

2 Related Work

Imitation and reinforcement learning: Imitation learning [1] has been widely used for learning robotic skills from expert demonstrations [2–5] and can be split into two areas: behavioral cloning [6, 7] and inverse reinforcement learning [8]. Both types of imitation learning typically require the expert to provide demonstrations in the same context as the learner. In robotics, this might be accomplished by means of kinesthetic demonstrations [9] or teleoperation [10], but both methods require considerable operator expertise. If we aim to endow robots with wide repertoires of behavioral skills, being able to acquire those skills directly from third-person videos of humans would be dramatically more scalable. Recently, a range of works have studied the problem of imitating a demonstration observed in a different context, e.g. from a different viewpoint or an agent with a different embodiment, such as a human [11–13]. Liu et al. [14] proposed to translate demonstrations between the expert and the learner contexts to learn an imitation policy by minimizing the distance to the translated demonstrations. In contrast to Liu et al., our framework can handle the domain shift between human and robot embodiment without any explicit joint-level alignment.

Label-free training signals: Label-free learning of visual representations promises to enable visual understanding from unsupervised data. Our method makes use of simultaneously recorded signals from multiple viewpoints to construct an image embedding. A number of prior works have used multiple modalities and temporal or spatial coherence to extract embeddings and features. For example, [15, 16] used co-occurrence of sounds and visual cues in videos to learn meaningful visual features. [17] also propose a multi-modal approach for self-supervision by training a network for cross-channel input reconstruction. [18, 19] use the spatial coherence in images as a self-supervision signal and [20] use motion cues to self-supervise a segmentation task. These methods are more focused on spatial relationships, and the unsupervised signal they provide is complementary to the one explored in this work. A number of prior works use temporal coherence [21–24]. Others also train for viewpoint invariance using metric learning [25, 27]. The novelty of our work is to combine both aspects in opposition, as explained in Sec. 3.1. [28] uses a triplet loss that encourages first and last frames of a tracked sequence to be closer together in the embedding, while random negative frames from other videos are far apart. Our method differs in that we use temporal neighbors as negatives to push against a positive that is anchored by a simultaneous viewpoint. This causes our method to discover meaningful dimensions such as attributes or pose, while [28] focuses on learning intra-class invariance.
3 Imitation with Time-Contrastive Networks

Our approach to imitation learning is to only rely on sensory inputs from the world. We achieve this in two steps. First, we learn abstract representations from passive observation of the world. Second, we use these representations to guide robotic imitations of human behaviors and train the robot to perform new tasks through reinforcement learning.

3.1 Training Time-Contrastive Networks

We illustrate our time-contrastive (TC) approach in Fig. 1 (left). The method uses multi-view metric learning via a triplet loss [29]. The core idea is that two frames (anchor and positive) coming from the same time but different viewpoints (or modalities) are pulled together, while a visually similar frame from a temporal neighbor is pushed apart. This signal serves multiple purposes. First, the cross-view correspondence encourages learning invariance to viewpoint, scale, occlusion, motion-blur, lighting and background, since the positive and anchor frames show the same subject with variations along these factors. For example, Fig. 1 (left) exhibits all these transformations between the top and bottom sequences except for occlusion. Learning a rich set of invariances is useful for vision in general, but we are particularly interested in viewpoint invariance for 3rd-person to 1st-person correspondence for imitation. Second, the time-contrastive signal introduces competition between temporal neighbors to explain away visual changes over time. For example, in Fig. 1 (left), since neighbors are visually similar, the only way to tell them apart is to model the amount of liquid present in the cup, or to model the pose of hands or objects and their interactions.

Multi-view data collection is simple and can be captured with as little as two operators equipped with smartphones. One operator keeps a fixed point of view of the region of interest while performing the task, while the other moves the camera freely to introduce the rich set of variations discussed above. While more cumbersome than single-view capture, we argue that multi-view capture is cheap, simple, and practical, when compared to alternatives such as human labeling.

We can also consider time-contrastive models trained on single-view video as shown in Fig. 1 (right). In this case, the positive frame is randomly selected within a certain range of the anchor. A margin range is then computed given the positive range. Negatives are randomly chosen outside of the margin range and the model is trained as before. While we show in Sec. 4 that multi-view TCN performs best, the single-view version can still be useful when no multi-view data is available.

3.2 Learning Robotic Behaviors with Reinforcement Learning

In this work, we consider an imitation learning scenario where the demonstrations come from a 3rd-person video observation of an agent with an embodiment that differs from the learning agent, e.g., robotic imitation of a human. Due to the differences of the contexts, direct tracking of the demonstrated pixel values does not provide a sensible way for learning the imitation behavior. As described in the previous section, the TCN embedding provides a way to extract image features that are invariant to the camera angle, manipulated objects and can explain physical interactions in the world. We use this insight to construct a reward function that is based on the distance between TCN embedding of a human video demonstration and camera images recorded with a robot camera. As shown in Sec. 4.2 by optimizing this reward function through trial and error we are able to mimic demonstrated behaviors with a robot by only utilizing its visual input and the video demonstrations for learning. Although we use multiple multi-view videos to train the TCN, the video demonstration consists only of a single video of a human performing a task from a random viewpoint.

Let $V = (v_1, \ldots, v_T)$ be the TCN embeddings of each frame in a video demonstration sequence. For each camera image observed during a robot task execution, we compute TCN embeddings $W = (w_1, \ldots, w_T)$. We define a reward function $R(v_t, w_t)$ based on the squared Euclidean distance and a Huber-style loss:

$$R(v_t, w_t) = -\alpha \| w_t - v_t \|^2 - \beta \sqrt{\gamma + \| w_t - v_t \|^2}$$

where $\alpha$ and $\beta$ are weighting parameters, and $\gamma$ is a small constant. The squared Euclidean distance (weighted by $\alpha$) gives us stronger gradients when the embeddings are further apart, which leads to larger policy updates at the beginning of the learning. The Huber-style loss (weighted by $\beta$) starts prevailing when the embedding vectors are getting very close ensuring high precision of the task execution and fine-tuning of the motion towards the end of the training.

In order to learn robotic imitation policies, we optimize the reward function described above using reinforcement learning. In particular, for optimizing robot trajectories, we employ the PILQR
algorithm [30]. This algorithm combines approximate model-based updates via LQR with fitted time-varying linear dynamics, and model-free corrections. We notice that in our tasks, the TCN embedding provides a well-behaved low-dimensional (32-dimensional in our experiments) representation of the state of the visual world in front of the robot. By including the TCN features in the system state, we can leverage the linear approximation of the dynamics during the model-based LQR update and significantly speed up the training. The details of the reinforcement learning setup can be found in Appendix A.1.

3.3 Direct Human Pose Imitation

In the previous section, we discussed how reinforcement learning can be used with TCNs to enable learning of object interaction skills directly from video demonstrations of humans. In this section, we describe another approach for using TCNs: direct imitation of human pose. While object interaction skills primarily require matching the functional aspects of the demonstration, direct pose imitation requires learning an implicit mapping between human and robot poses, and therefore involves a much more fine-grained association between frames. Once learned, a human-robot mapping could be used to speed up the exploration phase of RL by initializing a policy close to the solution.

We learn a direct pose imitation through self-regression. It is illustrated in Fig. 2 (left) and Fig. 2 (right) in the context of self-supervised human pose imitation. The idea is to directly predict the internal state of the robot given an image of itself. Akin to looking at itself in the mirror, the robot can regress its prediction of its own image to its internal states. We first train a shared TCN embedding by observing human and robots performing random motions. Then the robot trains itself with self-regression. Because it uses a TCN embedding that is invariant between humans and robots, the robot can then naturally imitate humans after training on itself. Hence we obtain a system that can perform end-to-end imitation of human motion, even though it was never given any human pose labels nor human-to-robot correspondences. We demonstrate a way to collect human supervision for end-to-end imitation in Fig. 2 (left). However contrary to time-contrastive and self-regression signals, the human supervision is very noisy and expensive to collect. We use it to benchmark our approach in Sec. 4.3 and show that large quantities of cheap supervision can effectively be mixed with small amounts of expensive supervision.

4 Experiments

Our experiments aim to study three questions. First, we examine whether the TCN can learn visual representations that are more indicative of object interaction attributes, such as the stages in a pouring


**Table 1:** Pouring alignment and classification errors: all models are selected at their lowest validation loss.

<table>
<thead>
<tr>
<th>Method</th>
<th>alignment error</th>
<th>classification error</th>
<th>training iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>28.1%</td>
<td>54.2%</td>
<td>-</td>
</tr>
<tr>
<td>Inception-ImageNet</td>
<td>29.8%</td>
<td>51.9%</td>
<td>-</td>
</tr>
<tr>
<td>Shuffle &amp; Learn[24]</td>
<td>22.8%</td>
<td>27.0%</td>
<td>575k</td>
</tr>
<tr>
<td>single-view TCN (triplet)</td>
<td>25.8%</td>
<td>24.3%</td>
<td>266k</td>
</tr>
<tr>
<td>multi-view TCN (npairs)</td>
<td>18.1%</td>
<td>22.2%</td>
<td>938k</td>
</tr>
<tr>
<td>multi-view TCN (triplet)</td>
<td>18.8%</td>
<td>21.4%</td>
<td>397k</td>
</tr>
<tr>
<td>multi-view TCN (lifted)</td>
<td>18.0%</td>
<td>19.6%</td>
<td>119k</td>
</tr>
</tbody>
</table>

4.1 Discovering Attributes from General Representations

4.1.1 Liquid Pouring

In this experiment, we study what the TCN captures simply by observing a human subject pouring liquids from different containers into different cups. The videos were captured using two standard smartphones, one from a subjective point of view by the human performing the pouring, and the other from a freely moving third-party viewpoint. Capture is synchronized across the two phones using an off-the-shelf app and each sequence is approximately 5 seconds long. We divide the collected multi-view sequences into 3 sets: 133 sequences for training (about 11 minutes total), 17 for validation and 30 for testing. The training videos contain clear and opaque cups, but we restrict the testing videos to clear cups only in order to evaluate if the model has an understanding of how full the cups are.

In all subsequent experiments, we use a custom architecture derived from the Inception architecture [31]. It consists of the Inception model up until the layer “Mixed_5d” (initialized with ImageNet pre-trained weights), followed by 2 convolutional layers, a spatial softmax layer [32] and a fully-connected layer. The embedding is a fully connected layer with 32 units added on top of our custom model. This embedding is trained either with the multi-view TC loss, the single-view TC loss or the Shuffle & Learn loss [24]. For the TCN models, we use the triplet loss from [29] without modification and with a gap value of 0.2. Note that in all experiments, negatives always come from the same sequence as positives. We also experiment with other metric learning losses, namely Npairs [33] and Lifted Structured [34], and show that results are comparable. We use the output of the last layer before the classifier of an ImageNet-pretrained Inception model [35] (a 2048-dimensional vector) as a baseline in the following experiments, and call it "Inception-ImageNet". Since the custom model is initialized from ImageNet pre-training, it is a natural point of comparison which allows us to control for any invariances that are introduced through ImageNet training rather than other approaches. We compare TCN models to a Shuffle & Learn baseline trained on our data, using the same hyper-parameters taken from the paper (tmax of 60, tmin of 15, and negative class ratio of 0.75). Note that in our implementation, neither the Shuffle & Learn baseline nor TCN benefit from a biased sampling to high-motion frames. To investigate the differences between multi-view and single-view, we compare to a single-view TCN, with a positive range of 0.2 seconds and a negative multiplier of 2.

4.1.2 Quantitative Evaluation

We present two metrics in Table 1 to evaluate what models are able to capture. The alignment metric measures how well a model can semantically align two videos. The classification metric measures how well a model can disentangle pouring-related attributes, that can be useful in a real robotic pouring task. All results in this section are evaluated using nearest neighbors in embedding space. Given each frame of a video, each model has to pick the most semantically similar frame in another video. The "Random" baseline simply returns a random frame from the second video.

The sequence alignment metric is particularly relevant and important when learning to imitate, especially from a third-party perspective. For each pouring test video, a human operator labels the key frames corresponding to the following events: first frame with hand contact with pouring container,
first frame where liquid is flowing, last frame where liquid is flowing and last frame with hand contact with the container. These keyframes establish a coarse semantic alignment which should provide a relatively accurate piecewise-linear correspondence between all videos. For any pair of videos \((v_1, v_2)\) in the test set, we embed each frame given the model to evaluate. For each frame of the source video \(v_1\), we associate it with its nearest neighbor in embedding space taken from all frames of \(v_2\). We evaluate how well the nearest neighbor in \(v_2\) semantically aligns with the reference frame in \(v_1\). Thanks to the labeled alignments, we find the proportional position of the reference frame with the target video \(v_2\), and compute the frame distance to that position, normalized by the target segment length.

In Table 1, we find that the multi-view TCN model outperforms all baselines. We observe that single-view TCN and Shuffle & Learn are on par for the classification metric but not for the alignment metric. We find that general off-the-shelf Inception features significantly under-perform compared to other baselines. Qualitative examples and t-SNE visualizations of the embedding are available in Appendix A.2.

4.2 Learning Object Interaction Skills

In this section, we use the TCN-based reward function described in Sec. 3.2 to learn robotic imitation behaviors from third-person demonstrations through reinforcement learning. We evaluate our approach on two tasks, plate transfer in a simulated dish rack environment (Fig. 3-left, using the Bullet physics engine [36]) and real robot pouring from human demonstrations (Fig. 3-right).

4.2.1 Task Setup

The simulated dish rack environment consists of two dish racks placed on a table and filled with plates. The goal of the task is to move plates from one dish rack to another without dropping them. This requires a complex motion with multiple stages, such as reaching, grasping, picking up, carrying, and placing of the plate. We record the human demonstrations using a virtual reality (VR) system to manipulate a free-floating gripper and move the plates (Fig. 3-1). We record the videos of the VR demonstrations by placing first-view and third-view cameras in the simulated world. In addition to demonstrations, we also record a range of randomized motions to increase the generalization ability of our TCN model. After recording the demonstrations, we place a simulated 7-DoF KUKA robotic arm inside the dish rack environment (Fig. 3-3) and attach a first-view camera to it. The robot camera images (Fig. 3-2) are then used to compute the TCN reward function. The robot policy is initialized with random Gaussian noise.

For the real robot pouring task, we first collect the multi-view data from multiple cameras to train the TCN model. The training set includes videos of humans performing pouring of liquids recorded on smartphone cameras and videos of robot performing pouring of granular beads recorded on two robot cameras. We not only collect positive demonstrations of the task at hand, we also collect various interactions that do not actually involve pouring, such as moving cups around, tipping them over, spilling beads, etc, to cover the range of possible events the robot might need to understand. The pouring experiment analyzes how TCNs can implicitly build correspondences between human and robot manipulation of objects. The dataset that we used to train the TCN consisted of \(\sim 20\) minutes of humans performing pouring tasks, as well as \(\sim 20\) additional minutes of humans manipulating cups and bottles in ways other than pouring, such as moving the cups, tipping them over, etc. In order for the TCN to be able to represent both human and robot arms, and implicitly put them into correspondence, it must also be provided with data that allows it to understand the appearance of robot arms. To that end, we added data consisting of \(\sim 20\) minutes of robot arms manipulating cups in pouring-like settings. Note that this data does not necessarily need to itself illustrate successful pouring tasks: the final demonstration that is tracked during reinforcement learning consists of a human successfully pouring a cup of fluid, while the robot performs the pouring task with orange beads. However, we found that providing some clips featuring robot arms was important for the TCN to acquire a representation that could correctly register the similarities between human and robot pouring. While the TCN is trained with approximately 1 hour of pouring-related multi-view sequences, the robot policy however is only learned from a single liquid pouring video provided by a human (Fig. 3-4). With this video, we train a 7-DoF KUKA robot to perform the pouring of granular beads as depicted in Fig. 3-6. We compute TCN embeddings from the robot camera images (Fig. 3-5) and initialize the robot policy using random Gaussian noise. We set the initial exploration higher on the wrist joint as it contributes the most to the pouring motion.
Figure 3: **Left:** Simulated dish rack task. (1) Third-person VR demonstration of the dish rack task. (2) View from the robot camera during training. (3) Robot executing the dish rack task. **Right:** Real robot pouring task. (4) Third-person human demonstration of the pouring task. (5) View from the robot camera during training. (6) Robot executing the pouring task.

Figure 4: **Left:** Learning progress of the pouring task, using a single 3rd-person human demonstration, as shown in Fig. 3 (right). This graph reports the weight in grams measured from the target recipient after each pouring action (maximum weight is 189g) along with the standard deviation of all 10 rollouts per iteration. The robot manages to successfully learn the pouring task using the multi-view TCN model after only 10 iterations. **Right:** Comparing types and amounts of supervision. Self-supervised imitation (“TC+self”) outperforms human-supervised imitation (“Human”).

### 4.2.2 Quantitative Evaluation

Fig. 4 shows the pouring task performance of using TCN models for reward computation compared to the same baselines evaluated in the previous section. After each roll-out, we measure the weight of the beads in the receiving container. We perform runs of 10 roll-outs per iteration. Results in Fig. 4 are averaged over 4 runs per model (2 runs for 2 fixed random seeds). Already after the first several iterations of using the multi-view TCN model (mvTCN), the robot is able to successfully pour significant amount of the beads. After 10 iterations, the policy converges to a consistently successful pouring behavior. In contrast, the robot fails to accomplish the task with other models. Interestingly, we observe a low performance for single-view models (single-view TCN and Shuffle&Learn) despite being trained on the exact same multi-view data as mvTCN. We observe the same pattern in Fig. 8 when using a different human demonstration. This suggests taking advantage of multi-view correspondences is necessary in this task for correctly modeling object interaction from a 3rd-person perspective. The results show that mvTCN does provide the robot with suitable guidance to understand the pouring task. In fact, since the PILQR [30] method uses both model-based and model-free updates, the experiment shows that mvTCN not only provides good indicators when the pouring is successful, but also useful gradients when it isn’t; while the other tested representations are insufficient to learn this task. This experiment illustrates how self-supervised representation learning and continuous rewards from visual demonstrations can alleviate the sample efficiency problem of reinforcement learning.

### 4.2.3 Qualitative Evaluation

As shown in our supplementary video[3], both dish rack and pouring policies converge to robust imitated behaviors. In the dish rack task, the robot is able to gradually learn all of the task components, including the arm motion and the opening and closing of the gripper. It first learns to reach for the plate, then grasp and pick it up and finally carry it over to another dish rack and place it there. The learning of this task requires only 10 iterations of 10 roll-outs per iteration. This shows that the TCN reward function is dense and smooth enough to efficiently guide the robot to a complex imitation policy.

In the pouring task, the robot starts with Gaussian exploration by randomly rotating and moving the cup filled with beads. The robot first learns to move and rotate the cup towards the receiving container, however missing the target cup and spilling large amount of the beads in the early iterations. After several more iterations, the robot learns to be more precise and eventually it is able to consistently pour most of the beads in the last iteration. This demonstrates that our method can efficiently learn tasks with non-linear dynamic object transitions, such as movement of the granular media and liquids, an otherwise difficult task to perform using conventional state estimation techniques.

4.3 Self-Regression for Human Pose Imitation

In the previous section, we showed that we can use the TCN to construct a reward function for learning object manipulation with reinforcement learning. In this section, we also study how the TCN can be used to directly map from humans to robots in real time, as depicted in Fig. 2 (right): in addition to understanding object interaction, we can use the TCN to build a pose-sensitive embedding either unsupervised, or with minimal supervision. The multi-view TCN is particularly well suited for this task because, in addition to requiring viewpoint and robot/human invariance, the correspondence problem is ill-defined and difficult to supervise. Apart from adding a joints decoder on top of the TCN embedding and training it with a self-regression signal, there is no fundamental difference in the method. Throughout this section, we use the robot joint vectors corresponding to the human-to-robot imitation described in Fig. 2 (left) as ground truth. Human images are fed into the imitation system, and the resulting joints vector are compared against the ground truth joints vector.

<table>
<thead>
<tr>
<th>Supervision</th>
<th>Robot joints distance error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random (possible) joints</td>
<td>42.4 ± 0.1</td>
</tr>
<tr>
<td>Self</td>
<td>38.8 ± 0.1</td>
</tr>
<tr>
<td>Human</td>
<td>33.4 ± 0.4</td>
</tr>
<tr>
<td>Human + Self</td>
<td>33.0 ± 0.5</td>
</tr>
<tr>
<td>TC + Self</td>
<td>32.1 ± 0.3</td>
</tr>
<tr>
<td>TC + Human</td>
<td>29.7 ± 0.1</td>
</tr>
<tr>
<td>TC + Human + Self</td>
<td>29.5 ± 0.2</td>
</tr>
</tbody>
</table>

Table 2: **Imitation error for different combinations of supervision signals.** The error reported is the joints distance between prediction and groundtruth. Note perfect imitation is not possible.

By comparing different combinations of supervision signals, we show in Table 2 that training with all signals performs best. We observe that adding the time-contrastive signal always significantly improves performance. In general, we conclude that relatively large amounts of cheap weakly-supervised data and small amounts of expensive human supervised data is an effective balance for our problem. Interestingly, we find that the self-supervised model (TC+Self) outperforms the human-supervised one. It should however be noted that the quantitative evaluation is not as informative here: since the task is highly subjective and different human subjects imitate the robot differently, matching the joint angles on held-out data is exceedingly difficult. We invite the reader to watch the accompanying video for examples of imitation, and observe that there is a close connection between the human and robot motion, including for subtle elements of the pose such as crouching: when the human crouches down, the robot lowers the torso via the prismatic joint in the spine. In the video, we observe a complex human-robot mapping is discovered entirely without human supervision. This invites to reflect on the need for intermediate human pose detectors when correspondence is ill-defined as in this case. In Fig. 9 we visualize the TCN embedding for pose imitation and show that pose across humans and robots is consistent within clusters, while being invariant to viewpoint and backgrounds. More analysis is available in Appendix A.3.

5 Conclusion

In this paper, we introduced a self-supervised representation learning method (TCN) based on multi-view video. The representation is learned by anchoring a temporally contrastive signal against co-occurring frames from other viewpoints, resulting in a representation that disambiguates temporal changes (e.g., salient events) while providing invariance to viewpoint and other nuisance variables. We show that this representation can be used to provide a reward function within a reinforcement learning system for robotic object manipulation, and to provide mappings between human and robot poses to enable pose imitation directly from raw video. In both cases, the TCN enables robotic imitation from raw videos of humans performing various tasks, accounting for the domain shift between human and robot bodies. Although the training process requires a dataset of multi-viewpoint videos, once the TCN is trained, only a single raw video demonstration is used for imitation.
References


A Appendix

A.1 Reinforcement Learning Details

Let \( p(u_t|x_t) \) be a robot policy, which defines a probability distribution over actions \( u_t \) conditioned on the system state \( x_t \) at each time step \( t \) of a task execution. We employ policy search to optimize the policy parameters \( \theta \). Let \( \tau = (x_1, u_1, \ldots, x_T, u_T) \) be a trajectory of states and actions. Given a cost function \( c(x_t, u_t) \), we define the trajectory cost as \( c(\tau) = \sum_{t=1}^{T} c(x_t, u_t) \). The policy is optimized with respect to the expected cost of the policy

\[
J(\theta) = \mathbb{E}_p[c(\tau)] = \int c(\tau)p(\tau) d\tau,
\]

where \( p(\tau) \) is the policy trajectory distribution given the system dynamics \( p(x_{t+1}|x_t, u_t) \).

One policy class that allows us to employ very efficient reinforcement learning methods is the time-varying linear-Gaussian (TVLG) controller \( p(u_t|x_t) = N(K_t x_t + k_t, \Sigma_t) \). In this work, we apply a reinforcement learning method PILQR \([30]\) to learn these TVLG controllers on a real robot, which combines model-based and model-free policy updates for an efficient learning of tasks with complex system dynamics.

Let \( S(x_t, u_t) = c(x_t, u_t) + \sum_{j=t+1}^{T} c(x_j, u_j) \) be the cost-to-go of a trajectory starting in state \( x_t \) by performing action \( u_t \) and following the policy \( p(u_t|x_t) \) afterwards. In each iteration \( i \), PILQR performs a KL-constrained optimization of \( S(x_t, u_t) \):

\[
\min_{p^{(i)}} \mathbb{E}_{p^{(i)}}[S(x_t, u_t)] \text{ s.t. } \mathbb{E}_{p^{(i-1)}}[D_{KL}(p^{(i)} \parallel p^{(i-1)})] \leq \epsilon,
\]

where limiting the KL-divergence between the new policy \( p^{(i)} \) and the old policy \( p^{(i-1)} \) leads to a better convergence behavior. The optimization is divided into two steps. In the first step, we perform a fast model-based update using an algorithm LQR-FLM \([37]\), which is based on the iterative linear-quadratic regulator (iLQR) \([38]\) and approximates \( S(x_t, u_t) \) with a linear-quadratic cost \( \hat{S}(x_t, u_t) \). In the second step, the residual cost-to-go \( \tilde{S}(x_t, u_t) = S(x_t, u_t) - \hat{S}(x_t, u_t) \) is optimized using a model-free method PI\(^2\) \([39, 40]\) to produce an unbiased policy update.

A.2 Objects Interaction Analysis

Here, we visualize the embeddings from the ImageNet-Inception and multi-view TCN models with t-SNE using a coloring by groundtruth labels. Each color is a unique combination of 5 attribute values defined earlier, i.e. if each color is well separated the model can identify uniquely all possible combinations of our 5 attributes. Indeed we observe in Fig. 5 some amount of color separation for the TCN but not for the baseline.

---

\(^4\)We use a cost function instead of the reward function as it is more common in the trajectory optimization theory.
Figure 5: **t-SNE colored by attribute combinations**: TCN (left) does a better job than ImageNet-Inception (right) at separating combinations of attributes.

<table>
<thead>
<tr>
<th>Method</th>
<th>alignment error</th>
<th>classif. error</th>
<th>training iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>28.1%</td>
<td>54.2%</td>
<td>-</td>
</tr>
<tr>
<td>Inception-ImageNet</td>
<td>29.8%</td>
<td>51.9%</td>
<td>-</td>
</tr>
<tr>
<td>single-view TCN (triplet)</td>
<td>26.6%</td>
<td>23.6%</td>
<td>738k</td>
</tr>
<tr>
<td>Shuffle &amp; Learn[24]</td>
<td>20.9%</td>
<td>23.2%</td>
<td>743k</td>
</tr>
<tr>
<td>multi-view TCN (lifted)</td>
<td>19.1%</td>
<td>21.4%</td>
<td>927k</td>
</tr>
<tr>
<td>multi-view TCN (triplet)</td>
<td>17.5%</td>
<td>20.3%</td>
<td>47k</td>
</tr>
<tr>
<td>multi-view TCN (npairs)</td>
<td>17.5%</td>
<td>19.3%</td>
<td>224k</td>
</tr>
</tbody>
</table>

Table 3: **Pouring alignment and classification errors**: these models are selected using the classification score on a small labeled validation set, then ran on the full test set. We observe that multi-view TCN outperforms other models with 15x shorter training time. The classification error considers 5 classes related to pouring: "hand contact with recipient", "within pouring distance", "container angle", "liquid is flowing" and "recipient fullness".

### A.3 Pose Imitation Analysis

#### A.3.1 Pose Imitation Data

The human training data consists of sequences distinguished by human subject and clothing pair. Each sequence is approximately 4 minutes. For the label-free TC supervision we collected approximately 30 human pairs (about 2 hours) where humans imitate a robot but the joint labels are not recorded, along with 50 robot sequences with random motion (about 3 hours, trivial to collect). For human supervision, we collected 10 human/clothing pairs (about 40 minutes, very expensive collection) while also recording the joints labels. Each recorded sequence is captured by 3 smartphone cameras fixed on tripods at specific angles (0°, 60° and 120°) and distance. The validation and testing sets each consist of 6 human/clothing pairs not seen during training (about 24 minutes, very expensive collection).

The distance error is normalized by the full value range of each joint, resulting in a percentage error. Note that the Human Supervision signal is quite noisy, since the imitation task is subjective and different human subjects interpret the mapping differently. In fact, a perfect imitation is not possible due to physiological differences between human bodies and the Fetch robot. Therefore, the best comparison metric available to us is to see whether the joint angles predicted from a held-out human observation match the actual joint angles that the human was attempting to imitate.
Figure 6: **Label-free pouring imitation:** nearest neighbors (right) for each reference image (left) for different models (multi-view TCN, Shuffle & Learn and ImageNet-Inception). These pouring test images show that the TCN model can distinguish different hand poses and amounts of poured liquid simply from unsupervised observation while being invariant to viewpoint, background, objects and subjects, motion-blur and scale.

### A.3.2 Models

We train our model using the 3 different signals as described in Fig. 2. The model consists of a TCN as described in Sec. 4 to which we add a joint decoder network (2 fully-connected layers above TC embedding: $\rightarrow 128 \rightarrow 8$, producing 8 joint values). We train the joints decoder with L2 regression using the self-supervision or human supervision signals. The model can be trained with different combinations of signals; we study the effects of each combination in Fig. 4 (right). The datasets used are approximately 2 hours of random human motions, 3 hours of random robot motions and 40 minutes of human supervision, as detailed in Appendix A.3.1. At test time, the resulting joints vector can then directly be fed to the robot stack to update its joints (using the native Fetch planner) as depicted in Fig. 2 (right). This results in an end-to-end imitation from pixels to joints without any explicit representation of human pose.

### A.3.3 Supervision Analysis

As shown in Fig. 2 (right), our imitation system can be trained with different combinations of signals. Here we study how our self-supervised imitation system compares to the other possible combinations of training signals. The performance of each combination is reported in Table 2 using the maximum amounts of data available (10 sequences for Human supervision and 30 sequences for TC supervision), while Fig. 4 (right) varies the amount of human supervision. Models such as the “TC + Self” or “Self” do not make use of any human supervision, hence only appear as single points on the vertical axis. Models that do not include TC supervision are simply trained as end-to-end regression problems. For example the “Self” model is trained end-to-end to predict internal joints from third person observations of the robot, and then that model is applied directly to the human imitation task. For reference, we compute a random baseline which samples joints values within physically possible ranges. In general, we observe that more human supervision decreases the L2 robot joints error. It is interesting to note that while not given any labels, the self-supervised model (“TC + Self”) still significantly outperforms the fully-supervised model (“Human”). The combination of all supervision signals performs the best. Overall, we observe that adding the TC supervision to any other signal significantly decreases the imitation error. In Fig. 10 we vary the amount of TC supervision provided and find the imitation error keeps decreasing as we increase the amount of data. Based on these results, we can make the argument that relatively large amounts of cheap weakly-supervised data and small amounts of expensive human supervised data is an effective balance for our problem.
non-extensive analysis of viewpoint and scale invariance in Sec. [A.4] seems to indicate that the model remains relatively competitive when presented with viewpoints and scales not seen during training.

A.3.4 Analysis by Joint

In Fig. [11] we examine the error of each joint individually for 4 models. Interestingly, we find that for all joints except for "shoulder pan", the unsupervised "TC+Self" models performs almost as well as the human-supervised "TC+Human+Self". The unsupervised model does not seem to correctly model the shoulder pan and performs worse than Random. Hence most of the benefits of human supervision found in Fig. [4] (right) come from correcting the shoulder pan prediction.

A.3.5 Qualitative Results

We offer multiple qualitative evaluations: k-Nearest Neighbors (kNN) in Fig. [7] imitation strips in Fig. [16] and a t-SNE visualization in Fig. [9]. Video strips do not fully convey the quality of imitation, we strongly encourage readers to watch the videos accompanying this paper. kNN: In Fig. [7] we show the nearest neighbors of the reference frames for the self-supervised model “TC+Self” (no human supervision). Although never trained across humans, it learned to associate poses such as crouching or reaching up between humans never seen during training and with entirely new backgrounds, while exhibiting viewpoint, scale and translation invariance. Imitation strips: In Fig. [16] we present an example of how the self-supervised model has learned to imitate the height level of humans by itself (easier to see in supplementary videos) using the "torso" joint (see Fig. [11]). This stark example of the complex mapping between human and robot joints illustrates the need for learned mappings, here we learned a non-linear mapping from many human joints to a single "torso" robot joint without any human supervision. t-SNE: We qualitatively evaluate the arrangement of our learned embedding using t-SNE representations with perplexity of 30 and learning rate of 10. In Fig. [9] we show that the agent-colored embedding exhibits local coherence with respect to pose while being invariant to agent and viewpoint. More kNN examples, imitation strips and t-SNE visualizations from different models are available in Sec. [A.5].
Figure 8: Learning progress of the pouring task, using a single 3rd-person human demonstration that is different from the one shown in Fig. 3 (right). This graph reports the weight in grams measured from the target recipient after each pouring action (maximum weight is 189g) along with the standard deviation of all 10 rollouts per iteration. The robot manages to successfully learn the pouring task using the multi-view TCN model after 20 iterations.

A.4 Imitation Invariance Analysis

In this section, we explore how much invariance is captured by the model. In Fig. 14, we test the L2 imitation error from new viewpoints (30°, 90°, and 150°) different from training viewpoints (0°, 60°, and 120°). We find that the error increases but the model does not break down and keeps a lower accuracy than the Human model in Fig. 4 (right). We also evaluate in Fig. 15 the accuracy while bringing the camera closer than during training (about half way) and similarly find that while the error increases, it remains competitive and lower than the human supervision baseline. From these experiments, we conclude that the model is somewhat robust to viewpoint changes (distance and orientation) even though it was trained with only 3 fixed viewpoints.

A.5 Imitation Examples

t-SNE: We qualitatively evaluate the arrangement of our learned embedding using t-SNE representations with perplexity of 30 and learning rate of 10. In this section we show the embedding before and after training, and colorize points by agent in Fig. 13 and by view in Fig. 12. The representations show that the embedding initially clusters views and agents together, while after training points from a same agent or view spread over the entire manifold, indicating view and agent invariance.
Figure 9: t-SNE embedding colored by agent for model "TC+Self". We show that images are locally coherent with respect to pose while being invariant to agent or viewpoint.

Figure 10: Varying the amount of unsupervised data: increasing the number of unsupervised sequences decreases the imitation error for both models.
Figure 11: **L2 robot error break-down by robot joints.** From left to right, we report errors for the 8 joints of the Fetch robot, followed by the joints average, followed by the joints average excluding the "shoulder pan" joint.

Figure 12: **t-SNE embedding before (left) and after (right) training, colored by view.** Before training, we observe concentrated clusters of the same color, indicating that the manifold is organized in a highly view-specific way, while after training each color is spread over the entire manifold.
Figure 13: t-SNE embedding before (left) and after (right) training, colored by agent. Before training, we observe concentrated clusters of the same color, indicating that the manifold is organized in a highly agent-specific way, while after training each color is spread over the entire manifold.

Figure 14: Testing TC+Human+Self model for orientation invariance: while the error increases for viewpoints not seen during training ($30^\circ$, $90^\circ$ and $150^\circ$), it remains competitive.

Figure 15: Testing for scale invariance: while the error increases when decreasing the distance of the camera to the subject (about half way compared to training), it remains competitive and lower than the human-supervised baseline.
Figure 16: **Self-supervised imitation examples.** Although not trained using any human supervision (model "TC+Self"), the TCN is able to approximately imitate human subjects unseen during training. Note from the rows (1,2) that the TCN discovered the mapping between the robot’s torso joint (up/down) and the complex set of human joints commanding crouching. In rows (3,4), we change the capture conditions compared to training (see rows 1 and 2) by using a free-form camera motion, a close-up scale and introduction some motion-blur and observe that imitation is still reasonable.