Learning Actionable Representations from Visual Observations

Debidatta Dwibedi [R]   Jonathan Tompson   Corey Lynch [R]   Pierre Sermanet

Google Brain

Abstract— In this work we explore a new approach for robots to teach themselves about the world simply by observing it. In particular we investigate the effectiveness of learning task-agnostic representations for continuous control tasks. We extend Time-Contrastive Networks (TCN) that learn from visual observations by embedding multiple frames jointly in the embedding space as opposed to a single frame. We show that by doing so, we are now able to encode both position and velocity attributes significantly more accurately. We test the usefulness of this self-supervised approach in a reinforcement learning setting. We show that the representations learned by agents observing themselves take random actions, or other agents perform tasks successfully, can enable the learning of continuous control policies using algorithms like Proximal Policy Optimization (PPO) using only the learned embeddings as input. We also demonstrate significant improvements on the real-world *Pouring* dataset with a relative error reduction of 39.4% for motion attributes and 11.1% for static attributes compared to the single-frame baseline. Video results are available at https://sites.google.com/view/actionablerepresentations

I. INTRODUCTION

Supervised learning is a powerful approach for learning useful visual representations for many vision tasks, such as image classification, object detection, and semantic segmentation. As such, many state-of-the-art deep-learning approaches for computer vision incorporate pre-training on a supervised surrogate loss, where ample training data is readily available; most commonly the large-scale ImageNet [1] or COCO [2] classification datasets.

It would be useful if one could likewise take advantage of supervised pre-training for image observation-based robotic control. However, there are three problems that complicate the application of pre-training in this domain. Firstly, it is not entirely clear what semantic labels or surrogate loss is applicable to the robotic control task at hand. In particular, semantic labeling of a scene is often unrelated to robotic control. The set of invariances required for strong classification performance (such as invariance to object pose, intra-class variation, etc) may be unsuitable for a robotic policy. Furthermore, such latent factors are intrinsically necessary to solve the task. Secondly, constructing a data collection pipeline and collecting labels for every new robotic task is prohibitively expensive. Finally, most robotic policies execute on domains that have little overlap with those of large-scale datasets like ImageNet or COCO; they will for instance have little or no coverage on robotic laboratory environments. Additionally, when a general-purpose robot discovers a new environment with previously unseen objects,
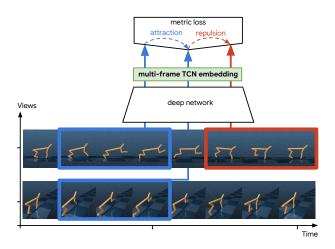
[R] Google AI Resident (g.co/airesidency)

Fig. 1. Given two viewpoints of the same event, we sample clips from the video and embed them to produce multi-frame Time-Contrastive Network (mfTCN) embeddings for each view. In order to train the deep network, we consider the clips at the same time from different views to be similar and clips from different time steps to be different.

it needs to adapt and learn about it. This image domain shift often necessitates additional data collection. Another motivation for learning solely from observation as opposed to learning with access to states or task rewards is to tap into new learning signals. For example one can learn to predict how things move in the world even for things that are out of its control. And this is not limited to learning from demonstration but also from non-demonstration events or objects in the world.

As an alternative to supervised learning, recent self-supervised approaches, such as Time-Contrastive Networks [3] (TCN) or Position Velocity Encoders [4] (PVE), have shown encouraging results when it comes to constructing robust visual representations suitable for Reinforcement Learning (RL) robotics applications, without the need for expensive supervised labels. These methods make use of readily available images directly from the domain of interest. They do so by constructing surrogate metric-learning losses that incorporate a number of structural priors (like temporal consistency, view invariance, etc.) that have been shown to efficiently and compactly encode latent state factors required for policy learning. PVE enabled learning continuous control policies in simulated environments while TCN was shown to help in imitation learning where a robot arm is used to pour liquids into a vessel. However, in the case of TCN, multi-frame state is not encoded, as the embedding is conditioned on a single frame. Likewise, PVE suggest a set of priors

that must be tuned for each environment. Additionally, self-supervised techniques have yet to match performance of policies trained directly on true state (rather than image observations).

However, despite the limitations of recent self-supervised techniques, such approaches learn representations that have a number of desirable properties that we wish to exploit in this work: 1. the embedding can be used to discriminate between different states (including motion attributes) visited in the course of an observation without the need for explicit state labels. 2. the embedding should be robust to changes in viewpoint, thus enabling third-person learning-from-demonstration for which large amounts of readily available training data exists (e.g. YouTube videos). 3. the embedding should be amenable to online adaption in new environments, without the need for additional labels.

In this work we present a variant of TCN, which we call multi-frame Time-Contrastive Network (mfTCN), that encodes temporal latent state (such as velocity, angular velocity, acceleration, etc). We show empirically that our embedding efficiently encodes multi-frame latent state from both real-world and simulation data, by showing that simple linear regressors and classifiers are able to obtain accurate state estimates from a mfTCN embedding. Finally, we show that this demonstration can be used as a basis for PPO[5] trained policies on simulated robotic control tasks in DeepMind Control Suite [6], and that mfTCN-based policies reward performance matches those trained from exact state representations.

The contributions of this paper are:

- Introducing a multi-frame variant of TCN which works better for both static and motion attributes classification.
- Showing RL policies can be learned from pixels using mfTCN while outperforming pixel-based ones learned from scratch or using PVEs.
- We also show that the learned policies are competitive with true (proprioceptive) state based policies. Hence, we refer to our representations as *actionable* as they not only encode both static and motion information present in the proprioceptive state but can also be used directly for continuous control tasks.

II. Related Work

Continuous control environments State-of-the-art performance of Reinforcement Learning (RL) algorithms has improved significantly in recent years and end-to-end, from-pixel policies have shown success on a number of benchmarks including Atari games [7] and continuous control [8]. Tassa et al. [6] introduced a compelling set of benchmark environments called Control Suite, which we make use of in this work. They are based on tasks initially introduced in the Mujoco environment by Todorov et al. [9]. Another popular benchmark for similar tasks is OpenAI's Gym [10].

Learning state representations using priors Learning useful state representations, in either an unsupervised or semi-supervised setting, has been a long-studied field in robotic control. Scholz et al. [11] showed that incorporating physics-based priors in the input state improved performance for model-based RL. Jonschkowski et al. [12] used similar physics-based *robotic-priors* to learn state representations consistent with the dynamics of the physical task. This framework was extended in [4] to include additional prior terms to capture multi-frame dynamics and where state representations were learned from visual observations only. Similarly to our work, state representations are learned from the observations produced by random agent actions in a simulated environment. Lesort et al. [13] introduce the reference point prior in their work in addition to the priors introduced in [4]. Concurrent to this work, [14] also demonstrate the importance of representing motion in continuous control tasks. They do so by using the optical flow predicted by a network as an additional input to the policy.

Self-supervised learning using images There have been multiple successes in using unsupervised and semi-supervised pre-training approaches to learn visual representations useful for the task of robotics and reinforcement learning in recent years. Watter et al. [15] learn a locally linear latent space that allows them use optimal control algorithms to follow trajectories in the embedding space. van Hoof et al. [16] use variational auto-encoders to stabilize a reinforcement learning system based on visual and tactile data streams. Finn et al. [17] successfully learn a model that encodes an input image in a low dimensional space. The model is trained to reconstruct the input image. The learned embedding is provided as input along with the true state of the robot. This joint representation enables the robot to preform complicated tasks like rice scooping and looping hooks which were not possible without the visual input. Munk et al. [18] propose an approach to map their input to a useful hidden state using "predictive priors" before training their Actor-Critic model using reinforcement learning. Agrawal et al. [19] introduce an interesting framework in which an agent first learns a model of the world by observing the effect of the random actions it takes. This learned representation can then be used to perform tasks that require multi-step decision making. Pinto et al. [20] show that it is possible to learn good visual representations by taking pre-defined actions with a robot in the real world. As the learned representations get better, this should also enable the robot to perform more complicated tasks. Recent work on imitation learning Yu et al. [21] even advocates directly learning policies from pixels directly by performing meta-learning to adapt to different demonstrations.

Self-supervised learning on videos Researchers have also had success in learning useful representation by using videos as input. Sermanet et al. [3] use time as a supervisory signal to learn the structure present in videos to learn a robust task-agnostic visual representation. Pathak et al. [22] also train a visual classifier that helps an agent identify intermediates states an agent needs to visit to complete a task. This classifier is trained using an self-supervised training objective based on temporal coherency. Finn et al. [23] introduced a dataset of robot-object interactions in the real-world. They learn concise visual representations by predicting the future

frames in a video. Babaeizadeh et al. [24] extend the above work by predicting the future frames in a stochastic manner. Dosovitskiy et al. [25] show that they can learn to take actions in an environment by predicting future state changes.

Auxiliary losses to improve representations While the above approaches highlight the advantage of using robust representations learned prior to the reinforcement learning, there has been recent work on adding auxiliary tasks to the RL objective to learn models that perform better [26], [27], [28]. Multi-task learning has also been shown to be helpful in learning more robust internal representations that ultimately lead to improvement in performance [29], [30], [31]. While in this work, we only consider the scenario where we learn task-agnostic representations before learning the control policy, we can adapt our approach to a multi-task setting where the policy and representation learning both improve jointly.

Time-Contrastive Networks (TCNs) In this work we extend TCNs introduced by Sermanet et al. [3] which uses time as a supervisory signal. Using TCNs it is possible to discriminate between various states that an agent might encounter while leanring to perform a task. The representations have been shown to be useful for imitation learning tasks. In traditional control systems, the input is the position and velocity of various objects or parts of the agent. Since, we will be using the learned representation to train agents to learn control policies, one desirable property of the embedding would be to encode both position and velocity. In the original TCN, the model was capable of encoding the state of the world and the position of various objects/agent parts. However, it was difficult for TCN to encode motion cues or velocity of objects/agent parts because it was conditioned on only a single frame. In this work, we extend TCNs to the multi-frame setting by embedding multiple-frames at each time step we expect to learn a representation that encodes both static and motion attributes.

III. APPROACH

Videos have the potential to be a rich source of data for a robot increasing the knowledge about its environment many fold. It is however difficult to learn control policies directly from pixels based only on the rewards that an agent receives. This is where robust visual representations can come in handy. Learning from visual observations seems to be a key ingredient in how humans pick up motor skills. Often times humans transfer skills to each other by performing demonstrations of the task. We also see examples of doctors teleoperating surgical instruments using visual feedback as opposed to having direct access to the position and velocities of the instruments. We envision a future where robots will be able to take advantage of visual feedback in conjunction with input from their other sensors. A good visual representation might also hold the key for transfer of skill from robots to humans via demonstrations in virtual reality. From a continual learning perspective, we want to learn a visual representation that gets updated even when no demonstrations are provided. The agent learns by observing changes in its environment.
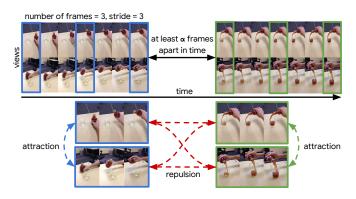

Fig. 2. In the above figure, we showcase how we sample a batch for training our model. As shown in the top row, we sample clips from both views simultaneously where each clip consists of 3 frames with a stride of 3 between each frame. We ensure that the clips are not overlapping and have at least α time steps gap between them. With just two samples, we can see the strength of the self-supervisory signal which forces the model to look for differences in similar looking frames while also looking for similarities between different views.

To that end, we propose a task-agnostic approach to learn such representations from observations that allows us to learn control policies from pixels. We essentially disentangle the representation learning phase from the task-specific control policy learning phase. We present details for both these sections below.

A. Learning Representations from Observations

In this phase, an agent first learns representations in a task-agnostic manner. The agent can learn from a variety of observations: from passive observations of its environment, from demonstrations by other agents, and from observing itself act in the world. This task-agnostic learning is versatile in the sense that it allows the model to learn from the world and from other agents (i.e. even when it does not have access to true state or even rewards) but also from itself. In this work, we restrict ourselves to the multi-view setting where for each observation we have two synchronized views (the model could be trained with more views as well).

Formally, we are provided with multiple videos as input from which an agent can observe and learn good representations of the world. Each video v has been collected from multiple synchronized viewpoints $v_1, v_2, ..., v_k$. As in [3], we use time as a supervisory signal. In Figure 2, we show how we create a training batch from the given videos. We consider clips at time t from all given viewpoints to be similar to each other (they will attract each other in embedding space). Additionally, these clips are also considered dissimilar to any clip that is beyond α steps in time in any view (dissimilar clips will repulse each other in embedding space). To encourage the network to learn representations that encode the above intuition, we learn a metric space with the embeddings produced by a base network. Unlike [3], instead of embedding a single frame at each timestep we embed multiple frames at each timestep. To do so, we introduce two hyperparameters: number of frames that are embedded together denoted as n and the stride between these

frames denoted as s. At each time step t, we embed the current frame and the previous $n-1$ frames chosen with a stride of s between the frames. This means at each timestep the network has a fixed lookback window of $(n-1) \times s + 1$ frames.

The motivation for embedding multiple frames is to allow the network to reason not only about the states of objects but also exploit the motion cues present in a scene. As demonstrated in Sec. IV and predictably, TCN has difficulty encoding motion since it embeds a single frame at a time. We alleviate this problem in the multi-frame version by embedding multiple frames jointly, which makes it easier to encode motion cues and velocity of objects.

We present the architecture of the deep network used in mfTCN in Figure 3. This network takes a clip that is a sequence of frames as input and jointly embeds them in a low-dimensional space. We describe different parts of the network below:

1) Base Network: We use a convolutional neural network (CNN) as the base network which will be used to extract low-dimensional representations from raw pixels. We will learn mfTCN embeddings on top of these representations. The base network encodes the hidden features h_t at time t from an input frame I_t:

$$h_t = CNN(I_t)$$

2) Temporal Aggregation: There are several approaches that can be used to aggregate temporal information from features extracted from frames at different time steps. We can perform temporal averaging, use temporal convolutions [32], [33] or use recurrent neural networks [34]. We choose to use 3D temporal convolutions to perform temporal aggregation in our model but other architectures may also be used.

$$\phi_t = Conv3D(h_t, h_{t-s}..., h_{t-(n-1) \times s})$$

3) Dimensionality Reduction: After the temporal aggregation step, we end up with a convolutional feature map that has 4 dimensions(time, height, width, channels). In order to learn a compact representations of entire clips, we reduce the dimensionality by first perfoming spatial averaging and then using a fully connected layer.

$$mfTCN_t = FC(\phi_t)$$

4) Time-Contrastive Learning: We use the n-pairs loss introduced in [35] to train our network. For explanation purposes, we borrow the terms "anchor, positive and negative" from the triplet loss [36] literature because similar concepts are used for the n-pairs loss. In general, an anchor-positive pair is a pair of samples that we want to be closer to each other in embedding space than the anchor-negative pair. The loss aims at learning a metric embedding which clusters data samples of the same category closer to each other while pushing them farther from samples from different category in the learned embedding space. To be able to use the n-pairs loss in a time-contrastive setting, we sample non-overlapping clips at the same timestep from multiple views. We obtain k

examples (equal to the number of views) at each timestep. These examples are considered as positives. If we have sampled at n timesteps, then we get $k \times (n-1)$ examples are negatives for the k positives at each timestep (See Figure 2). In other words, clips sampled from different views but at the same time are positive while clips sampled at a different time are considered as negative irrespective of the view. This heuristic provides the required supervisory signal to train our model. In Figure 2, we can see that with just two samples from two views we are able to generate 6 supervisory labels: 2 pairs of embeddings that should be close to each other and 4 pairs of embeddings that should be distant from each other. This number grows combinatorially with the number of timesteps we sample in a given batch. This rich supervisory signal coupled with the context provided by embedding multiple frames jointly allows us to learn good representations from visual observations.

5) Architectural Differences between TCN and mfTCN: There are a couple of architectural differences between the original TCN [3] and mfTCN. First, there is a 3D convolution layer in mfTCN that takes convolutional feature maps from different time steps as input. Second, we do not have a spatial softmax [37] layer in mfTCN which was used by the original TCN to reduce dimensionality.

B. Learning Control Policies

In order to test the utility and robustness of our embeddings we decide to perform continuous control tasks on top of these learned representations. Continuous control algorithms usually take the true state (joint angle, positions etc) as input. We want our learned representation to be a drop-in replacement for the true states required for a particular task. In order to do so , we choose PPO [5] as the on-policy optimization algorithm that allows us to learn continuous control policies.

IV. EXPERIMENTS

A. Regression to velocities and positions of Cartpole

We aim to show that the mfTCN embeddings are able to encode velocities and positions of objects and parts of agents. It is important to measure how well our model is able to capture this information because if we expect to be able to use these embeddings to perform continuous control tasks they need to be encoding precise information about these entities.

In order to measure this quantitatively, we collect multi-view videos of the *Cartpole* environment from the Deepmind Control Suite [6]. We use the default multiple camera setup in their environment. We learn a multi-frame TCN embedding from the videos collected where the agent is taking random actions. The random actions follow the same distribution as [4]. We evaluate if these embeddings are able to encode the position and velocity of different objects present in the scene. In order to do so, we follow the experimental setup of [4] in which they train regressors on top of their embeddings using the true states provided by the simulator. Like Position Velocity Encoders, we also concatenate the
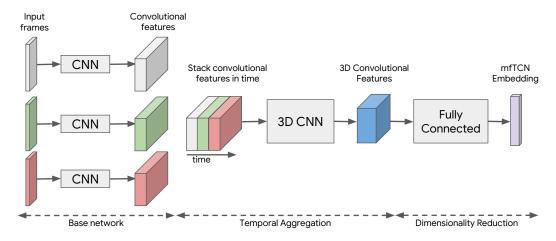
Fig. 3. Architecture used to extract mfTCN embeddings from a given sequence of frames. The 3D convolution layer is added to aggregate temporal information across frames and encode motion cues in the mfTCN embedding.

difference between embeddings at time t and time $(t - 1)$ along with our embedding. We train 3 fully connected layers of 256 dimensions on top of the learned embedding with an Adam optimizer and learning rate of 0.001. The trained regressors are then shown frames from the validation set and predict the true states of the agent. We use a train/test split of 160000 and 20000 frames for this task. The results of this experiment are reported in Table I. We observe that the multi-frame embeddings are able to encode the true position and velocity better. Additionally, the Mean Squared Error (MSE) decreases when we increase the size of the embedding learned. Interestingly, we do not explicitly train for the embedding to encode position or velocity but it stands to reason the model needs to encode them, among other attributes, to differentiate between clips at different timesteps while still knowing that multiple views at the same time should encode the same state.

B. Policy learned on TCN embedding from Self-Observation

In this experiment we consider the scenario where an agent is able to observe itself in its environment. We consider the *Swing-up* task in the Cartpole domain of Deepmind's Control Suite. The task is to swing the pole up by applying forces at the base of the cart and then to balance it without deviating too much from the center of the base. The physical model is similar to the one presented in [38]. In order to collect data to train the mfTCN, the agent performs random actions with random initial states. We use the default camera setup. Note in this setup the second camera is moving with the agent.

We use the Proximal Policy Optimization (PPO) [5] algorithm to learn a policy on top of the learned mfTCN embeddings. Usually the true states (like positions of objects, velocities of joints etc.) are provided as input to learn a policy. We replace the true states with the learned low-dimensional representation as input to PPO. We report the mean and standard deviation of rewards for 100 episodes/rollouts at test time. Following [6], we rollout for 1000 steps for each episode.

The results of this experiment are reported in Table II.

We observe that if we train both the representation learning network and the policy network in parallel on raw pixels the agent is not able to learn the task(row 3). As a baseline, we use embeddings from the Position Velocity encoders to train a PPO policy and that results in the agent performing better(row 4). We observe that the mfTCN trained on one frame at the same resolution(row 5) is able to outperform both raw pixels and PVE as input by a large margin. The performance improves significantly when we jointly embed 5 frames(row 6). This highlights the advantages of embedding multiple frames jointly as opposed to only using a single frame. We find performance of mfTCN improves again when we provide input at a higher resolution(rows 7 and 8).

Since, we learned an MFTCN model from multiple views, we do not need to retrain another representation learning model to train another policy for the second camera, which in this case is moving. We use the same mfTCN network and train another PPO policy and find that the version of mfTCN which embeds 5 frames jointly(row 10) works much better with the moving camera as compared to the one which embeds only one frame(row 9).

C. Policy learned on TCN embedding from Observing Other Agents

Contrary to the previous experiment, we consider the scenario where an agent is able to observe other similar agents performing a given task. In particular, we consider the *Cheetah* environment which is a tougher control task than *Cart-pole* in terms of having a larger state space and possible number of actions. We are provided with demonstration videos of the *Cheetah* agent walking successfully. These demonstrations were generated by training a PPO policy on the true state of the agent. We use the default camera setup for this environment. We train our mfTCN model on these videos only. One manner in which these demonstrations differ from the previously considered scenario where an agents performs random actions is that the agent only ever sees successful examples of walking and does not see the plethora of states a Cheetah encounters while it is learning

TABLE I

RESULTS ON REGRESSING CARTPOLE POSITIONS AND VELOCITIES FROM mFTCN EMBEDDINGS

Embedding size	Lookback window	MSE			MSE in Static Attributes			MSE in Motion Attributes	
		Average	Position	Motion	x_{cart}	$sin(\theta_{pole})$	$cos(\theta_{pole})$	$\dot{x}_{cart}$	$\dot{\theta}_{pole}$
8	1	0.1312	0.0201	0.2979	0.0238	0.0085	0.0279	0.4964	0.0993
8	2	0.0435	0.0017	0.1062	0.0011	0.0025	0.0016	0.1587	0.0537
8	4	**0.0207**	**0.0014**	**0.0497**	0.0008	0.0020	0.0014	0.0788	0.0207
32	1	0.0911	0.0052	0.2201	0.0045	0.0037	0.0072	0.3716	0.0686
32	2	0.0401	0.0019	0.0974	0.0014	0.0027	0.0015	0.1456	0.0492
32	4	**0.0198**	**0.0013**	**0.0476**	0.0008	0.0020	0.0013	0.0748	0.0203
128	1	0.0636	0.0040	0.1528	0.0030	0.0035	0.0056	0.2396	0.0661
128	2	0.0419	0.0018	0.1020	0.0011	0.0027	0.0015	0.1489	0.0552
128	4	**0.0211**	**0.0014**	**0.0505**	0.0007	0.0022	0.0014	0.0783	0.0227

TABLE II

RESULTS ON CARTPOLE SWINGUP TASK

Input to PPO	# lookback Frames	From Pixels	Resolution	Cumulative Reward Mean	Std.
Random State	1			121.45	11.98
True State	1			861.41	3.46
Pixels (CNN)	1	✓	96×48	283.82	42.88
PVE [4]	1	✓	96×48	457.27	51.16
mfTCN	1	✓	96×48	550.98	53.40
mfTCN	5	✓	96×48	701.30	80.14
mfTCN	1	✓	160×160	759.33	77.77
mfTCN	5	✓	160×160	787.47	67.80
mfTCN (moving)	1	✓	160×160	691.77	85.28
mfTCN (moving)	5	✓	160×160	811.10	41.80

TABLE IV

ATTRIBUTES CLASSIFICATION AND ALIGNMENT METRICS ON THE POURING DATASET: mFTCN CONSISTENTLY OUTPERFORMS OTHER BASELINES.

Model	Lookback window	Static Error (%)	Motion Error (%)	Alignment Error (%)
Random	1	54.2	-	-
Inception	1	51.9	-	-
Shuffle&Learn[39]	1	27.0	-	-
TCN [3]	1	22.2	-	-
mfTCN[1]	1	18.9	30.2	16.2
mfTCN[7]	7	17.3	24.8	14.3
mfTCN[13]	13	**16.8**	**18.3**	11.3
mfTCN[29]	29	19.4	20.9	**8.9**

to walk. This is similar to the real-life imitation learning settings where it is easy to gather successful demonstrations. The big question is: can we learn a representation useful for learning the task at hand by only observing successful demonstrations? With this experiment we show that it is possible to do so for the *Walk* task in the *Cheetah* domain. Similar to the above section, we report the mean and standard deviation of rewards for 100 episodes/rollouts at test time. Following [6], we rollout for 1000 steps for each episode.

TABLE III

RESULTS ON CHEETAH WALK TASK

Input to PPO	Cumulative Reward Mean	Std.
Random State	28.31	3.62
True State	390.16	44.85
Pixels (CNN)	146.14	29.51
mfTCN	360.50	76.52

D. Classification results on Pouring dataset

To evaluate the usefulness of our learned representations on real-world data we conduct experiments on the *Pouring* dataset introduced in [3]. Additionally this allows us to compare the mfTCN model with its TCN cousin using the results reported in [3]. The dataset consists of videos of humans pouring different liquids into cups. The dataset has two synchronized views of the captured scene: a fixed first-person view and a moving third person view (see Fig. 2 for an example). We use a pre-trained InceptionV3 [40] as the base network and extract convolutional features from *Mixed_5d* to train the mfTCN model (we do not fine-tune the weights below that as in [3]). In particular, we are interested in observing the effect of varying the size of lookback window of mfTCN on the classification of attributes relevant for the *Pouring* task. we describe those attributes in detail below.

In the original dataset, frames are manually labeled with answers to questions that should be pertinent to the task of pouring. The original dataset had the following questions and answers:

1) Is the hand in contact with the container? (yes or no)
2) Is the container within pouring distance of the recipient? (yes or no)
3) What is the tilt angle of the pouring container? (90, 45, 0 and -45 degrees)
4) Is the liquid flowing? (yes or no)
5) Does the recipient contain liquid? (yes or no)

These questions, though relevant to the task, are restricted to information contained only in a single-frame. We augment the set of questions with motion based questions that are also relevant to pouring but will typically require more than 1 frame to answer. The motion based questions are all binary

TABLE V

Model	Lookback window	Error in Static Attributes (%)					Error in Motion Attributes (%)			
		contact	angle	distance	has liquid	liquid flowing	container down	container up	hand reaching	hand receding
Random	1	49.9	74.5	48.9	49.2	48.4	-	-	-	-
Inception	1	47.4	71.8	45.2	48.8	49.2	-	-	-	-
Shuffle & Learn [39]	1	17.2	46.3	17.8	25.7	28.0	-	-	-	-
TCN [3]	1	8.0	**35.9**	**9.0**	24.7	35.5	-	-	-	-
mfTCN[1]	1	**7.2**	39.5	9.5	27.4	14.6	34.5	30.6	30.5	29.6
mfTCN[7]	7	8.2	39.7	10.6	22.1	10.3	25.7	25.6	25.8	25.0
mfTCN[13]	13	8.2	38.7	9.1	**18.5**	**10.2**	**22.6**	**19.7**	**17.2**	**17.6**
mfTCN[29]	29	8.6	37.5	11.2	16.1	10.3	25.4	20.3	16.2	20.7

questions (yes or no):

1) Is the hand reaching towards the container?
2) Is the hand receding from the container?
3) Is the container in contact with the hand going up?
4) Is the container in contact with the hand coming down?

Note that these labels are only used for the purposes of evaluation but never during training. Following the experimental setup of [3], we use a nearest neighbor classifier to quantitatively evaluate the usefulness of the learned embeddings. Given a sequence from the evaluation set, we retrieve the embedding nearest neighbor of each frame in the set of embeddings from all frames of all other sequences of the evaluation set (the current sequence is excluded). The labels of the selected nearest neighbor are used to classify the attributes for the current frame. We also report the an alignment error metric that measures how well two views of a video are aligned in time. We take advantage of the fact that frames in both videos are in sync with each other to get alignment labels for free. For any two views of the same demonstration, we first embed all frames using mfTCN. Then for each frame in the first view we look for the nearest neighbours in the second view. We retrieve the time index of the nearest neighbours ($time_{knn}$) to see how well they are aligned. The alignment error for a particular frame i is defined as follows:

$$error_i = \frac{abs(|time_i - time_{knn}|)}{sequence\ length}$$

In Table IV, we report the attributes classification and average alignment error metrics on the Pouring dataset. We report classification error for static and motion attributes separately to identify better the contributions of the mfTCN model over the TCN baseline (which operates with only 1 frame as input). We also report in more details the error rates of each attribute in Table V. We find that mfTCN outperforms other baselines and obtains the best classification results with a window of size 13 (mfTCN[13]). We report the results found in [3] and compare mfTCN[1] to the TCN baseline. These two models are equivalent in that they both use a single frame as input, however their architectures differ (details in Sec. III-A.5) which explains the observed performance difference. As expected, a window size greater than 1 yields significant improvements in classification of motion attributes (from

30.2% error down to 18.3% error with mfTCN[13]). More surprisingly perhaps we also observe an error reduction for the static attributes. One explanation can be found in Table V by observing that attributes "has liquid" and "liquid flowing" get a significant boost of performance, we hypothesize that these attributes can be recognized both using static and motion cues, hence taking motion into account helps. For the alignment error, we find that the model with the largest input window (mfTCN[29]) performs best. One interpretation is that it is easier to disambiguate clips in the embedding space if there is more context available to the model. Overall, we find that mfTCN consistently outperforms the TCN baseline.

E. Discussion

We show that our approach can encode the proprioceptive states of an agent from pixels. Like [3], we expect the approach to also learn rich representations about relevant objects in the environment. One drawback of the present approach is that the embedding can choose to fixate on some objects in the environment while ignoring others. Even though in our experiments we only used the learned embeddings to learn control policies, in a more practical setting one should use both the embedding as well as the proprioceptive states as input. Although our representation learning approach is self-supervised, it still relies on being presented with a reasonable coverage of possible states. Such issues may be alleviated with an explicit exploration strategy (like intrinsic motivation [41]) or expert demonstrations.

V. Conclusion

In this paper, we extended TCN by allowing it to embed multiple frames jointly. We show that by doing so we get better estimates of positions and velocities which leads to better performance in continuous control tasks. Embedding multiple frames also leads to significant performance improvement on the *Pouring* dataset.

We show that this approach to learning robust visual representations allows us to use policy learning algorithms effectively directly from video as opposed to true states. The results on the simulated environments are encouraging and we aim to use this model to learn more robust policies on real robots. In the future, we also want to be able to refine the representation learned based on any new data that the agent encounters after taking actions.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[3] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine, "Time-contrastive networks: Self-supervised learning from video," *Proceedings of International Conference in Robotics and Automation (ICRA 2018)*.

[4] R. Jonschkowski, R. Hafner, J. Scholz, and M. Riedmiller, "Pves: Position-velocity encoders for unsupervised learning of structured state representations," *arXiv preprint arXiv:1705.09805*, 2017.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[6] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, *et al.*, "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[9] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[11] J. Scholz, M. Levihn, C. Isbell, and D. Wingate, "A physics-based model prior for object-oriented mdps," in *International Conference on Machine Learning*, 2014, pp. 1089–1097.

[12] R. Jonschkowski and O. Brock, "Learning state representations with robotic priors," *Autonomous Robots*, vol. 39, no. 3, pp. 407–428, 2015.

[13] T. Lesort, M. Seurin, X. Li, N. D. Rodríguez, and D. Filliat, "Unsupervised state representation learning with robotic priors: a robustness benchmark," *arXiv preprint arXiv:1709.05185*, 2017.

[14] A. Amiranashvili, A. Dosovitskiy, V. Koltun, and T. Brox, "Motion perception in reinforcement learning with dynamic objects," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 156–168. [Online]. Available: http://proceedings.mlr.press/v87/amiranashvili18a.html

[15] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in neural information processing systems*, 2015, pp. 2746–2754.

[16] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 3928–3934.

[17] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 512–519.

[18] J. Munk, J. Kober, and R. Babuška, "Learning state representation for deep actor-critic control," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 4667–4673.

[19] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems*, 2016, pp. 5074–5082.

[20] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta, "The curious robot: Learning visual representations via physical interactions," in *European Conference on Computer Vision*. Springer, 2016, pp. 3–18.

[21] 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.

[22] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, F. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *ICLR*, 2018.

[23] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in neural information processing systems*, 2016, pp. 64–72.

[24] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine, "Stochastic variational video prediction," *arXiv preprint arXiv:1710.11252*, 2017.

[25] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *arXiv preprint arXiv:1611.01779*, 2016.

[26] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell, "Loss is its own reward: Self-supervision for reinforcement learning," *arXiv preprint arXiv:1612.07307*, 2016.

[27] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[28] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[29] L. Pinto and A. Gupta, "Learning to push by grasping: Using multiple tasks for effective learning," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2161–2168.

[30] H. Arora, R. Kumar, J. Krone, and C. Li, "Multi-task learning for continuous control," *arXiv preprint arXiv:1802.01034*, 2018.

[31] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4499–4509.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4489–4497.

[33] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4724–4733.

[34] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[35] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems*, 2016, pp. 1857–1865.

[36] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015.

[37] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Learning visual feature spaces for robotic manipulation with deep spatial autoencoders."

[38] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

[39] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and learn: unsupervised learning using temporal order verification," in *European Conference on Computer Vision*. Springer, 2016, pp. 527–544.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[41] A. G. Barto, "Intrinsic motivation and reinforcement learning," in *Intrinsically motivated learning in natural and artificial systems*. Springer, 2013, pp. 17–47.