Self-Supervised Representation Learning for Continuous Control

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.**

I. Introduction

Many state-of-the-art approaches for various computer vision tasks incorporate a visual representation learning step. Often this pre-training is performed using 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 pixel-based robotic control. However, it is not entirely clear what is the right supervision required for such tasks.

Alternatively 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) and robotics applications, without the need for expensive supervised labels. These approaches learn representations from observations in the domain of interest. The common thread in these approaches is the construction of auxiliary losses that push the model to learn useful structural priors like temporal consistency, view invariance etc. 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, both approaches have certain drawbacks. In TCN, the embedding is conditioned on a single frame. This makes it difficult for learning motion cues. Likewise, PVE requires the tuning of a set of priors for each environment. Furthermore, representation learned using these techniques have yet to match the performances of policies trained directly on true state.

[R] Google AI Resident (http://g.co/airesidency)

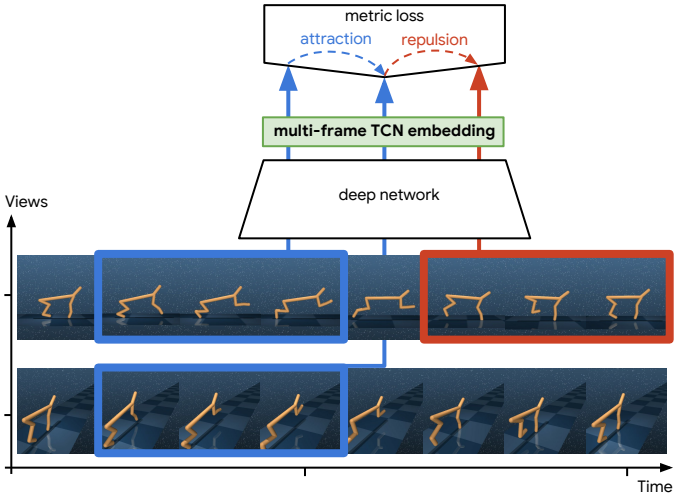

Fig. 1. Given two viewpoints of the same event, we sample clips from the video and embed them to produce multi-frame Time-Contrastive Network (mfTCN) embeddings for each view. In order to train the deep network, we consider the clips at the same time from different views to be similar and clips from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

However, despite the limitations of recent self-supervised techniques, such approaches learn representations with 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.

The contributions of this paper are:

- We introduce a multi-frame variant of TCN which we call multi-frame Time-Contrastive Network (mfTCN), that encodes temporal latent states (such as velocity, angular velocity, acceleration, etc) better.
- We show that these representations can be used as input for PPO[5] trained policies on simulated robotic control tasks in DeepMind Control Suite [6].
- We also show that the policies learned using mfTCN are competitive with true-state based policies.

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] used physics based priors for representation learning and showed that incorporating these improved performance for model-based RL. In a follow-on work, 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 this 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 and also showcase transfer of policies learnt in simulation to real robots.

Self-supervised learning from visual observations There have been multiple successes in using self-supervised pre-training approaches to learn visual representations useful for the task of robotics and reinforcement learning in recent years. Watter et al. [14] learn a locally linear latent space that allows them use optimal control algorithms to follow trajectories in the embedding space. van Hoof et al. [15] use variational auto-encoders to stabilize a reinforcement learning system based on visual and tactile data streams. Finn et al. [16] 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. [17] 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. [18] 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. 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. Dosovitskiy et al. [19] show that they can learn to take actions in an environment by predicting future state changes.

III. Approach

Our approach consists of two phases: first is the task-agnostic representation learning phase followed by the task-specific control policy learning phase.

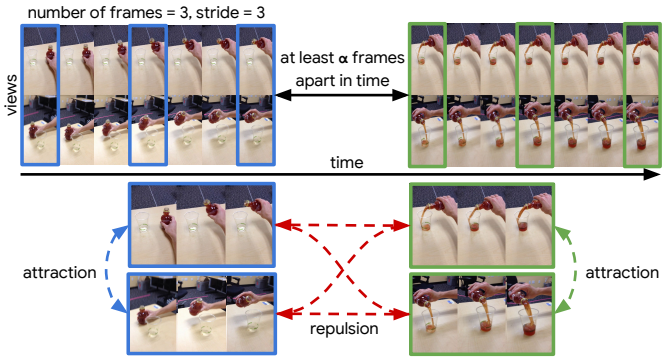

Fig. 2. In the above figure, we showcase how we sample a batch for training our model. As shown in the top row, we sample clips from both views simultaneously where each clip consists of 3 frames with a stride of 3 between each frame. We ensure that the clips are not overlapping and have at least α time 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.

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).

We are provided with multiple videos as input from which an agent can observe and learn good representations of the world. 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. 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, single frame 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.

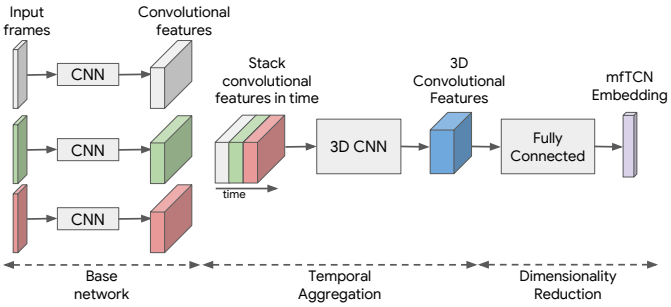

Fig. 3. Architecture used to extract mfTCN embeddings from a given sequence of frames. The 3D convolution layer is added to aggregate temporal information across frames and encode motion cues in the mfTCN embedding.

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 use the n-pairs loss introduced in [20] to train our network. For explanation purposes, we borrow the terms "anchor, positive and negative" from the triplet loss [21] 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. Clips sampled from different views but at the same time are positive while clips sampled at a different time are considered as negatives irrespective of the view. This heuristic provides the required supervisory signal to train our model.

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 angles and end-effector positions) as input. We want our learned representation to be a drop-in replacement for the true states required for a particular task. We choose a 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. Precise information about these quantities are important to solve continuous control tasks. 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. 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 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 [22]. 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 [4] 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).

TABLE I

RESULTS ON REGRESSING CARTPOLE POSITIONS AND VELOCITIES FROM MFTCN EMBEDDINGS

	MSE			MSE in Static Attributes			MSE in Motion Attributes	
Lookback window	Average	Position	Motion	x_{cart}	$sin(\theta_{pole})$	$cos(\theta_{pole})$	$\dot{x}_{cart}$	$\dot{\theta}_{pole}$
1	0.0911	0.0052	0.2201	0.0045	0.0037	0.0072	0.3716	0.0686
2	0.0401	0.0019	0.0974	0.0014	0.0027	0.0015	0.1456	0.0492
4	**0.0198**	**0.0013**	**0.0476**	**0.0008**	**0.0020**	**0.0013**	**0.0748**	**0.0203**

Since, we learned an MFTCN model from multiple views, we do not need to retrain a different 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).

TABLE II

RESULTS ON CARTPOLE SWINGUP TASK

Input to PPO	# look-back 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

C. Policy learned on TCN 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 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. 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 [23]) 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. We show that this approach to learning robust visual representations allows us to use policy learning algorithms effectively on the learned representations as opposed to the true state. 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 representations based on any new states that the agent encounters as it starts learning control policies.

ACKNOWLEDGEMENT

We thank Sergey Levine and Vincent Vanhoucke for reviews and constructive feedback. We are grateful to Martin Riedmiller and Rico Jonschkowski for their help with the Position-Velocity Encoder models.

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] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

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

[22] 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.

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