TECLAT: Transition Encoding using Contrastive Learning based Auxiliary Tasks for Sample Efficiency

Sahana Joshi
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, USA
sjoshi330@gatech.edu

***Abstract*—Most model-free reinforcement learning(RL) algorithms in RL employ policy gradient methods to directly learn a parameterized policy. While this avoids the overhead of learning the dynamics of the environment, they are less sample-efficient. Auxiliary tasks are known to improve sample-efficiency by leveraging other training signals present in the environment; the reasoning being that solving auxiliary tasks helps the agent build better representations. Contrastive learning(CL) is a self-supervised learning technique used to build good representations across domains(visual, language and RL). Previous works which have tried CL in RL have used RGB state observations or latent space representations of state as input for computing contrastive loss. The objective of this paper is to extend a CL based auxiliary task baseline (CURL) to encode full transition information, i.e, state, action and next state. The intuition is that this will build representations that have cause and effect embedded in them and will act as additional training signals.**

***Index Terms*—reinforcement learning, auxiliary tasks, contrastive learning, CURL**

I. Introduction and Related Work

Auxiliary tasks are objectives which are different from the main one of maximizing the return. They can be reward-agnostic, and usually are in cases where reward is sparse. Jaderberg et al. [1] introduce two kinds of tasks: control(generate pseudo-rewards when agent changes the environment in a desired way) and reward tasks(train to predict the reward given a state, action information). The agent is trained with a loss which is a weighted sum of losses from the main and auxiliary tasks. van den Oord et al. [2] introduce Contrastive Predictive Coding for representation learning, which combines next state prediction in latent space with a novel contrastive loss function called InfoNCE. They demonstrate the breadth of contrastive objective with results from speech, image, text and RL domains. Srinivas et. al [3] introduce CURL which is the baseline for this paper. CURL creates a contrastive objective on the raw pixel observations and uses this objective as an auxiliary task in the agent training pipeline. The CURL encoder is jointly trained with the return-loss and the contrastive loss. This paper extends CURL by learning a contrastive objective over state transitions, i.e, treating the (state, action, next_state) as the input to the encoder. The intuition is that this would encode cause and effect relations in the encoder because we are trying to extract features from not just the current state, but that from the current state, an example action taken in that state and the resultant next state. The motivation is that since we are building a more enriched encoder, it would be capable of constructing representations which lead to good policies faster, hence improving sample efficiency. Key contributions: Novel encoder to embed transition information.

College of Computing, Georgia Institute of Technology

II. Methodology

A. Baseline : Pixel encoder from CURL

The CURL architecture as in Fig. 1 shows the training of the agent with main and contrastive auxiliary tasks. Observations, which are stacked RGB frames are sampled from the replay buffer. Data augmentation, such as random crop is performed on the observation to generate query and key pairs. The query acts as the *anchor* for CL objective and the key acts as the *positive* sample. Other similarly cropped observations from the mini-batch serve as *negatives*. The CL objective is to match a given query q with its data augmented twin key k from a set of negative samples, which are augmentations of different observations. The Query and Key encoders are both pixel encoders which accepts raw RGB observations as input and produce embeddings for the query and key augmentations. q and k are used to compute the contrastive loss. q is used as the state embedding by the RL objective which computes policy. The Key encoder is the momentum averaged version of the query encoder, which Srinivas et al. [3] cite as being similar to the target network in DQN [4], used to stabilize training of the query encoder.

B. Alpha : Full Transition encoder

In TECLAT, we introduce a novel encoder, shown in which consists of a copy of the pixel encoder network and a MLP network. The encoder takes the state and next state to produce corresponding embeddings. The embeddings are combined with the action and passed to the MLP network to compute a

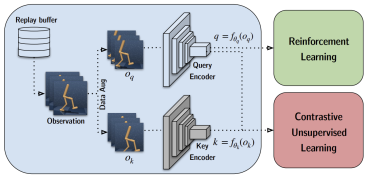

Fig. 1. CURL Architecture

score T_target(s,a,s'). In this pass, we do not accumulate the gradients in the Transition Encoder network.

The sampled transitions' state and next states are also passed to the critic's(and actor's) pixel encoder and the embeddings from this combined with the action are passed to the Transition encoder's MLP network to arrive at a score T(s,a,s'). In this pass, we accumulate gradients in the critic's pixel encoder and the transition encoder's MLP network.

The mean-squared error between T_target and T is used an an auxiliary loss to update the weights of the critic's encoder and the MLP network. The weights of the critic's pixel encoder are periodically moved to the transition encoder's pixel encoder, similar to the the DQN architecture Mnih et. al [4].

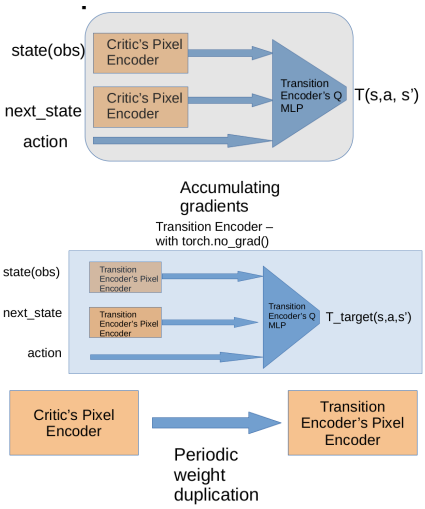

Fig. 2. (top) Critic network computes T, accumulating gradients (middle) Transition network computes T_target (bottom) Periodic weight duplication between Pixel encoders of Transition network and Critic network

This architecture is different from the one presented at the mid-term stage and was brought about by the poor performance obtained by entirely replacing the Pixel encoder with the Transition Encoder which had similar CNN structure as the Pixel encoder(refer mid-term report). The earlier structure did not make much sense because an action is taken based on the state as input and not a full transition. Hence, learning from full transitions is useful as an auxiliary task and not as a replacement for the Pixel encoder.

III. Results

Similar to CURL, we used DeepMind Control Suite(dm2gym) environments. Due to limited resources, we ran experiments on 3 games, tasks: *CartPole, Swingup*, *Cheetah, Run* and *Walker, Walk*. We also limited the number of timesteps to 100k(instead of 1M in CURL paper) for the same reasons.

A. Baseline

The results from the baseline are as shown in the table below. The training and evaluation plots are in Fig. 3

TABLE I
BASELINE RESULTS AT 60K TIMESTEPS

Game/task	Score
Cartpole, Swingup	838.3
Cheetah, Run	321.9
Walker, Walk	481.7

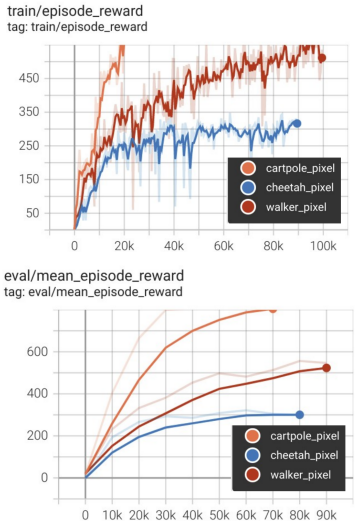

Fig. 3. Baseline: Training and Evaluation plots of reward v/s timestep

B. Alpha: Full transition encoder

The results from the initial implementation of the FullTransition encoder are shown in the table below. Due to training constraints, at the moment, we only have the result for the Walker, Walk task. The training and evaluation plots are in Fig. 4

TABLE II
ALPHA RESULTS AT 60K TIMESTEPS

Game/task	Score
Walker, Walk	15.8

As seen from the results above, the full transition encoder performs very poorly compared to the Pixel encoder and is not able to learn any reasonable policy, let alone one that improves sample efficiency. Upon applying the architecture change described in the previous section, we can see that there is a good boost in performance in training and evaluation as

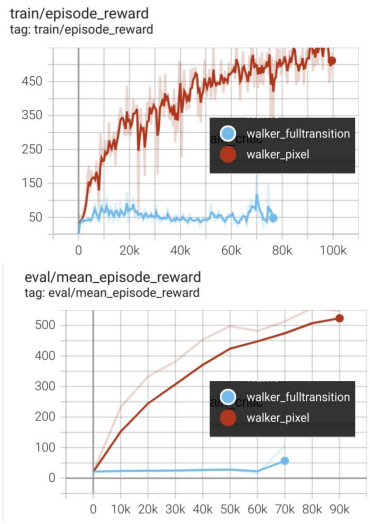

Fig. 4. Old Alpha: Training and Evaluation plots of reward v/s timestep

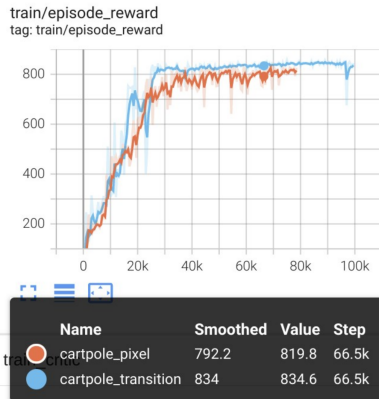

Fig. 5. New Alpha: Training plot of reward v/s timestep

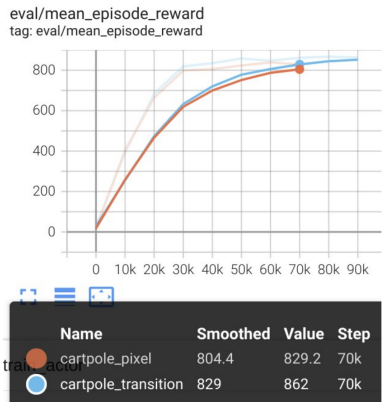

Fig. 6. New Alpha: Evaluation plot of reward v/s timestep

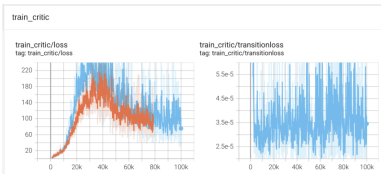

Fig. 7. Multiple losses affecting encoder

seen in Fig. 5 and Fig. 6. The reward curves confirms that the new architecture is effective in learning a decent policy.

There are multiple losses affecting the training of the critic, in turn, the training of its encoder: the value function loss and the newly added transition loss. As seen in the Fig. 7, the vanilla critic loss(left) is of a much higher magnitude than the transition loss. So we formulate the hypothesis below:

If the auxiliary transition loss is actually aiding in learning, then adding a higher weight to this loss should boost performance, i.e, if in an attempt to aggressively reduce the transition loss, better representations arise, then this will lead to sample efficiency.

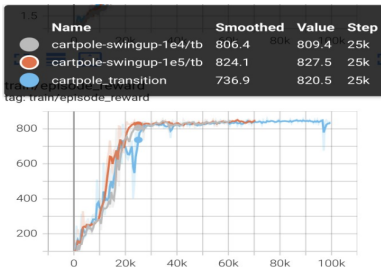

Fig. 8. Increasing weights on transition loss(training): blue: 1, grey: 10k, blue: 100k

To test this hypothesis, we ran experiments with with increasing weights on the transition loss. The results of this experiment are shown in Fig. 8. As seen, increasing the weight boost performance, with a huge jump of 70 points from weight 1 to 10e4 at as early as 25k timesteps, and again a jump of 25 points from weight 10k to 100k. This confirms the hypothesis that transition loss is aiding in learning.

Since there is an increasing trend in performance, there is scope of squeezing gains indefinitely by increasing weights further, but keeping an eye on evaluation performance(Fig. 9) shows that the increase in performance is not as drastic as seen in training(differences as small as 10 points). This difference in training and evaluation trends can be attributed to generalization error arising from potential over-fitting.

Further, we compare the best performing alpha network setting with a weight 100k to the baseline network as shown in Fig. 10. As seen, the alpha network not only outperforms the baseline throughout the training curve, but also attains sample efficiency at as early as 16k timesteps. The experiment was tried with 3 random seeds to ensure consistency.

Unfortunately, the success of the architecture did not generalize to Walker, Walk and Cheetah, Run tasks as shown in

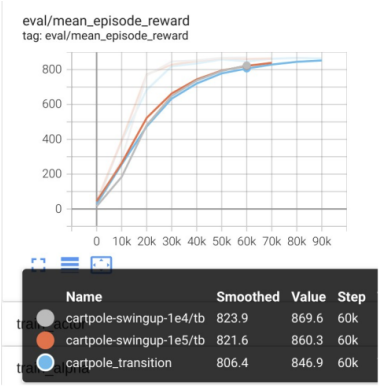

Fig. 9. Increasing weights on transition loss(eval): blue: 1, grey: 10k, blue: 100k

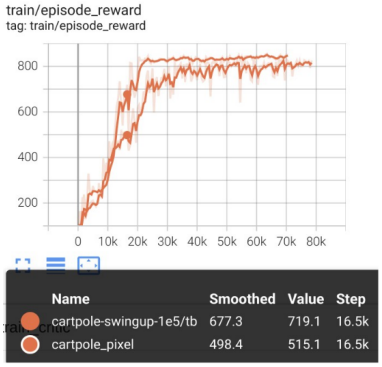

Fig. 10. Sample Efficiency Comparison

Fig. 11 and Fig. 12. This could be because hyper-parameter tuning required for the tasks.

IV. Learnings and Conclusions

In conclusion, we have proposed a novel encoder which extends a strong baseline CURL to learn from full transitions instead of just states. The idea is to extract as much information as possible from interactions with the environment in the attempt learn better representations and attain sample

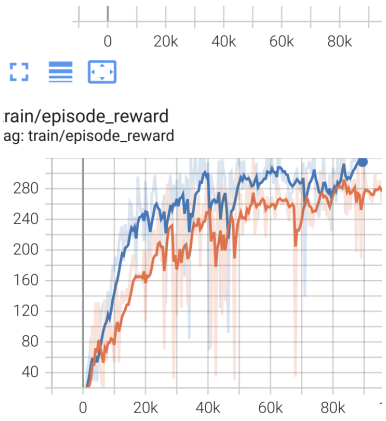

Fig. 11. Poorer performance than baseline on Cheetah,Run task

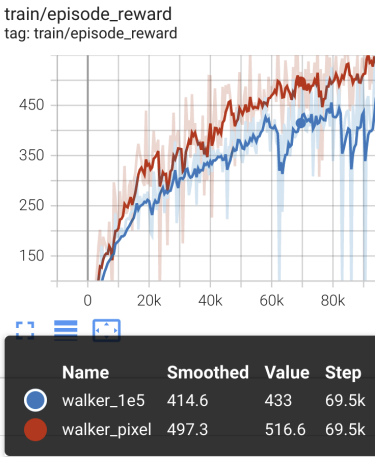

Fig. 12. Poorer performance than baseline on Walker,Walk task

efficiency. We use this encoder to predict a transition score(T) as an auxiliary task and use this loss in tandem with other losses of the architecture(like actor loss, critic loss , entropy loss) to update the weights of the encoder. The architecture proposed imbibes concepts like target network from DQN used for stabilizing learning.

A salient feature of this alpha network is that the transition scores(T and T_target) do not rely on the reward, unlike the value function. So it can prove to be more useful when the environment gives sparse rewards like RTS games like Starcraft2, although we have only tested on environments with fairly dense reward settings in this project.

From the experiments conducted, transition encoding is helping in achieving sample efficiency. It does however require more training time due to additional parameters added. As seen from the evaluation graphs, it is also subject to over-fitting.

Future work would include expanding testing to other games with continuous action space and sparse-reward settings. Ablation studies and hyper parameter tuning on this architecture would comprise of questions like (1) How sensitive is the transition loss weight to the task at hand? (2) How does the weight duplication frequency affect learning? (3) Can other loss functions replace the mean-squared loss used in this project?

References

[1] Max Jaderberg, Volodymyr Mnih , Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver and Koray Kavukcuoglu “REINFORCEMENT LEARNING WITH UNSUPERVISED AUXILIARY TASKS”

[2] Aaron van den Oord, Yazhe Li, Oriol Vinyals, “Representation Learning with Contrastive Predictive Coding”

[3] Aravind Srinivas, Michael Laskin, Pieter Abbeel, “CURL: Contrastive Unsupervised Representations for Reinforcement Learning”.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller Playing Atari with Deep Reinforcement Learning