Optimization of Deep Reinforcement Learning with Hybrid Multi-Task Learning

Nelson Vithayathil Varghese
Department of Electrical, Computer, and Software Engineering
Ontario Tech University
Oshawa, ON, L1G 0C5 Canada
nelson.vithayathilvarghese@ontariotechu.net

Qusay H. Mahmoud
Department of Electrical, Computer, and Software Engineering
Ontario Tech University
Oshawa, ON, L1G 0C5 Canada
qusay.mahmoud@ontariotechu.ca

2021 IEEE International Systems Conference (SysCon) | 978-1-6654-4439-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/SysCon48628.2021.9447080

***Abstract*—As an outcome of the technological advancements occurred within artificial intelligence (AI) domain in recent times, deep learning (DL) has been established its position as a prominent representation learning method for all forms of machine learning (ML), including the reinforcement learning (RL). Subsequently, leading to the evolution of deep reinforcement learning (DRL) which combines deep learning's high representational learning capabilities with current reinforcement learning methods. Undoubtedly, this new direction has caused a pivotal role towards the performance optimization of intelligent RL systems designed by following model-free based methodology. Optimization of the performance achieved with this methodology was majorly restricted to intelligent systems having reinforcement learning algorithms designed to learn single task at a time. Simultaneously, single task-based learning method was observed as quite less efficient in terms of data, especially when such intelligent systems required operate under too complex as well as data rich conditions. The prime reason for this was because of the restricted application of existing methods to wide range of scenarios, and associated tasks from those operating environments. One of the possible approaches to mitigate this issue is by adopting the method of multi-task learning. Objective of this research paper is to present a parallel multi-task learning (PMTL) approach for the optimization of deep reinforcement learning agents operating within two different by semantically similar environments with related tasks. The proposed framework will be built with multiple individual actor-critic models functioning within each environment and transferring the knowledge among themselves through a global network to optimize the performance.**

Keywords—deep reinforcement learning, transfer learning, multi-tasking, actor-mimic.

I. Introduction

Over the last few decades, reinforcement learning domain has been well established its position as a vital topic within technological areas such as robotics and intelligent agents [1]. The core target of RL is focused on finding an optimal way by which RL agents should explore their environment , and which in turn cause the agent to choose most optimal actions to obtain achieve the highest possible reward at any given time step *t* [2]. Supported by recent the advancements within the field of ML domain, RL has been established itself as a major ML prototype which governs how a RL agent should act optimally under its operating environment. When it comes to the performance metric analysis of other ML classes like supervised learning, and unsupervised learning, RL based systems are found to be possessing suboptimal levels. This was predominantly due to restrictions encountered by the DRL agent in deriving the optimal policy from the large state-action space of the operating environment. At the same time, inception of DL with its very high level of representational learning capability has provided a new direction within the field of RL namely, DRL. Subsequent to this, wide adoption of DRL based systems were employed across multiple domains related to continuous action control,3D first-person environments, and gaming. Within the context of gaming, DRL based models are proven to be quite successful with their ability to surpass the scores achieved by the human-player on well-known video games like Atari as well as on various popular board-based games namely chess and Go [3].

Despite the optimal the results achieved within systems with single-task oriented learning, the performance of intelligent agent is observed to be suboptimal or average within more data rich and complex and 3-dimensional game environments. Recommended and possible ways to optimize the performance RL agent under such an environment is by the application of multi-task-based learning. According to this, a group of tasks that are bonded close enough from the operating environment would be explored and analyzed simultaneously by multiple intelligent agents. Operation of each of these individual agents will be guided by a DRL algorithm like A3C (Asynchronous Advantage Actor-Critic) [4]. Under this, each of these individual agents designed as neural networks will be sharing its network parameters with a global network at regular intervals asynchronously. After accumulating the learning parameters from all of the agents, a new batch of network parameters are calculated by the global network. Following this, same will be shared with to individual agents. The main purpose this step is for ensuring that enhance the overall level of agent's performance by distributing positive knowledge or shared learning across all of the closely bonded tasks operating under unique environment. The most widely accepted multi-task learning technique within the RL is referred as the parallel based multi-task learning, and with this single agent to aims to learn r a collection of tasks [5]. This approach mainly relies on the architecture used by the DRL model which is based on a single learner (also known as critic) coupled to various actors. With this method, actors independently generate its own learning trajectories or traversal paths in the environment, described by a set of network parameters, and further exchanges the same with learner module (or the critic) either in a synchronous or asynchronous fashion. Subsequent to this, updated parameter list

978-1-6654-4439-2/21/$31.00 ©2021 IEEE

from the learner module will be gathered by all the actors so as to have the updated parameters before initiating next learning cycle. By following the above-mentioned step, it is ensured learnings gathered from previous steps of actions are by each agent are shared mutually among them. This would in turn act as a driving force to accelerate the overall learning momentum of the RL intelligent agent.

The rest of this paper is organized as follows. Section II discussed the related work with special emphasis on three state-of-the-art solutions, namely: Distral, IMPALA and PopArt. Section III details the proposed hybrid multi-task learning model by explaining its architecture and functioning. The implementation of the proposed hybrid multi-task learning model is presented in section IV.Details of the of experiments and associated results are presented in section V. Finally, Section VI concludes the paper and offers ideas for future work.

II. Related Work

Distral (DIStil and TRAnsfer Learning) is one of the well-known approaches developed by google DeepMind for the purpose of multi-task training. It is a prototype made for the purpose of concurrent RL with more than one task [6]. The key design objective was to build a generic model with the intention of distilling the centroid policy first, and following this transfer the commonality details and behavior patterns of multiple workers operating within multi-task RL context. Rather than following a parameter sharing based policy among the multiple worker agents operating in the environment, Distral's design methodology mainly focus on distributing a distilled policy to individual workers, and this policy should conceive the commonality in behavior across multiple related tasks. Once the distilled policy is derived, then same can be used to govern the task-specific policies by adopting regularization with the help of Kullback-Leibler (KL) divergence [7]. By this method, initially knowledge gathered from one task would be distilled in the form of a shared policy, subsequently same knowledge could be transferred to other related tasks operating in the environment. By adopting this methodology, each of the individual workers would be trained independently to solve own task, in such a way that each of the workers could be staying more in line with shared policy. Training for this policy will be conducted with the help of distillation process that serves as centroid for all the individual task policies [6]. This approach is found to present impressive results in terms of the transfer of knowledge within complex 3 dimensional operating environments for RL problems.

Empirically it has been observed that Distral approach often outperforms the traditional methods, by a significant margin, that are oriented on parameter sharing policy of neural networks towards achieving multi-tasking or transfer learning. The two key reasons behind this are mentioned as below. Firstly, its due to the level of impact distillation has got on the process of optimization. It is more prevalent while adopting KL divergences as prime method to regularize task models' output in deriving the distilled model extracted from each of the policies of individual tasks. Secondly, application of distilled model itself as a means to regularize for the purpose of training the individual task models within the environment. More importantly, application of the distilled model as a method to regularize, comes with the notion of regularizing collection of individual workers in a much impactful manner by stressing on task policies by more margin than at the level of parameter [8].

Google DeepMind came up with another well-known multi-task learning approach by the name IMPALA (Importance Weighted Actor-Learner Architecture). It is based on the idea of having a distributed agent architecture which is designed by adopting the model of a single RL agent with only one set of parameters. The core design characteristic of IMPALA model is about operating environment flexibility. This model is designed with the ability to not only in the efficient utilization of resources with in single-machine oriented training environment but also it can be scaled to operate with multiple machines without the need to sacrifice both data efficiency and utilization of resource. By following a novel off-policy correction method by the name V-trace, IMPALA is capable of gaining quite stable learning trajectory with very high throughput level by combining both decoupled acting as well as learning [9]. In general, the DRL model's architecture follows the notion of a single learner (also known as critic) clubbed with many numbers of actors. Under this ecosystem, initially each of the individual actor creates its own learning parameters called trajectories, and subsequently shares that knowledge with the learner (critic) by following a queue mechanism. The learner subsequently accumulates the same kind of knowledge trajectories from all of the multiple actors operating within the environment, which eventually acts as source of information to prepare the central policy. Before starting the next learning cycle (trajectory), all of the individual actors operating within the environment gathers the updated policy parameters details from the learner(critic module) [9]. This approach is quite analogous to the popular RL algorithm named A3C.The architecture of the IMPALA was inspired hugely by same algorithm. RL algorithm model used within the IMPALA follows a topology of a system having a group of actors and learners who build knowledge through collaboration.

Design of the IMPALA leverages actor-critic based model to derive a policy π and, a baseline value function named $V\pi$. Major units of IMPALA system consists of a group of actors that generates g trajectories of experience in a continuous manner. In addition to this, there could be at least one or more than one learner that leverage the generated trajectories shared from the individual actors to learn the policy π, which is an off-policy. At the beginning of every individual trajectory, an actor initially updates its local policy μ to the latest learner policy π. Subsequently, each individual actor would adopt and run that policy for n number of steps in its operating environment. Upon completion of these n steps, each of the individual actors sends another set of information consisting of - trajectory of states, actions, and rewards together with related policy distributions to the learner. In this manner, the learner will have the opportunity to continuously update its policy π each time whenever the actors share their trajectory information from the environment. In this fashion, IMPALA architecture gathers experiences from different individual learners within the environment ,which are further passed to a central learner module. Following this, central learner calculates the gradients, and then generates a model having a framework of independent actors as well as learners. One of the major characteristics of the IMPALA architecture is its operational flexibility which allows the actors

to be present either on the same machine or it can be even distributed across numerous machines.

A third approach by the name PopArt proposed by Google DeepMind came out as a solution to mitigate the issues associated with existing IMPALA model. PopArt was aimed at to address the reasons behind the suboptimal performance factors and thereby enhance the RL in multi-task-oriented environments. The core design objective of PopArt is to reduce the impacts of distraction dilemma problem associated with the IMPALA model and, thereby stabilize the learning process in a better way to facilitate the adoption of multi-task RL techniques [10]. The term distraction dilemma refers about the probability of learning algorithms getting distracted only by a fraction of few tasks from the large pool of multiple tasks to be solved. This scenario in turn leads to the challenges related to resource contention. It is about establishing a right balance between the necessities of multiple tasks operating within same environment competing for limited number of resources offered from single learning system. The design methodology of PopArt model is based on the original IMPALA architecture model by adding multiple CNN layers combined with other techniques like word embeddings with the help of recurrent neural network of type long-short term memory (LSTM) [10].

PopArt model functions by gathering the trajectories from each of the individual tasks to the RL agent's updates. By this manner, PopArt model make sure that every agent within the environment will have its own role, subsequently proportional impact during dynamics of overall learning. The key design aspect of the PopArt model relies on the fact that modifying the weights of the neural network, will be based on the output of all tasks operating within the environment. During first stage of operation, PopArt estimates both mean as well as the spread of the ultimate targets such as the score of a game across all tasks under consideration. Following this, PopArt capitalize on these estimate values to normalize the targets before making update on network's weights. This approach in turn makes the whole learning process more stable and robust. With the set of various experiments conducted with popular Atari games' environment, PopArt has demonstrated its capabilities and improvements over other multi-task RL architectures [11].

III. Proposed Hybrid Multi-Tasking Learning Method

Major motivation behind the proposed hybrid multi-task learning approach is to address and mitigate some of the key challenges associated with DRL multi-tasking, which are not fully covered by the state of the art. To this end, our proposed approach would be attempting to address and mitigate the optimization bottlenecks posed by challenges such as partial observability, amount of the training data samples required, effective exploration and also the amount training time needed to achieve acceptable levels of performance [12].

The proposed approach named hybrid multi-task learning model is an attempt to address the aforementioned aspects by extending the basic actor-critic model to two different operating environments with a high level of semantic similarity. The key aspect the of A3C algorithm is its ability to learn multiple instantiations of a single target task simultaneously, and also its ability to improve the model's performance by transferring the knowledge between multiple instantiations [4]. The proposed hybrid multi-task learning approach will be leveraging this key aspect and will attempt to achieve this objective across, two different by semantically similar environments with related tasks. The hybrid approach proposed here will be heavily relying on the applicability of the multi-threaded capability of the A3C algorithm across semantically related tasks running in two different environments. The proposed hybrid multi-task learning approach could be treated as a model running two threads of the A3C algorithm, wherein each thread will be managing the multiple instantiations of the tasks running in each environment. Each of these individual threads would consider itself as a subtask such as A and B, with each of them sharing its individual learning with the learner, which is a global network, in an asynchronous manner. Further on, the learner will be converging the knowledge from both of these threads and deducing a new policy, which will be applied back on the individual agent worker threads. The key design aspect of the hybrid multi-task model is to enhance the performance of the RL agent through a joint-learning through multi-task learning approach by using the deep reinforcement learning. Fig. 1 shows the high-level architecture model of the proposed hybrid multi-tasking approach.

Hybrid multi-task learning model deploys multi-threaded asynchronous variants of A3C algorithm. The objective behind designing this model is to find way to train deep neural network policies reliably and without large resource requirements. For the development of the hybrid multi-task learning model, we are using a prototype based on eight asynchronous actor-learners by using multiple CPU threads on a Windows 10 based cloud server machine having GPU support hosted by Paperspace.Over the course of execution, this model asynchronously attempts to derive and optimize the global policy based on the observations that multiple actors-learners running in parallel are likely to be exploring different parts of the environment. At an individual actor-learner module level, it is possible to have different exploration policies in each module to maximize this diversity. In this way, having different exploration policies in different threads of actor-learner module, the overall changes being made to the global network parameters by different actor-learners applying asynchronous updates in parallel are likely to be less correlated.

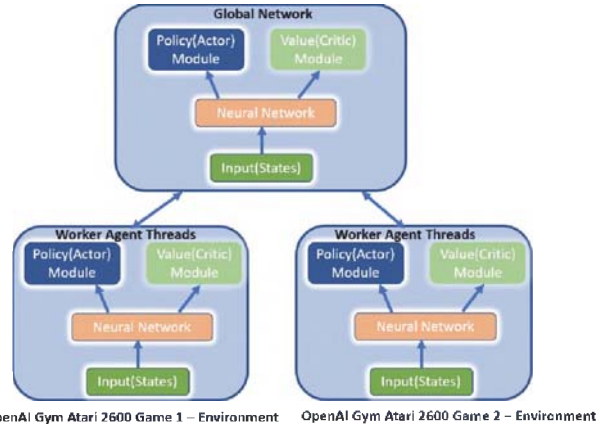

Fig. 1. Architecture of the Proposed Hybrid Multi-task Learning Model

This model is designed to run on a single machine with a standard multi-core CPU/GPU support, and applied to a variety of Atari 2600 domain games for the testing. The semantic similarity aspect of the related tasks running two different gaming environments is the most vital factor to achieve the above-mentioned objectives, which otherwise give challenges in terms of negative knowledge transfer.Negative Transfer is considered to be one of the key challenges while dealing with the multi-tasking within the reinforcement learning domain. The main idea of knowledge transfer learning in a multi-task context is that transferring knowledge accumulated from learning from a set of source samples under one agent may improve the performance of another agent while learning on the target task [13]. However, the direction knowledge transfer can have impact the on overall learning progress and thereby performance of the agent in either way, either positive or negative . This impact will be more prevalent depending on the magnitude of either difference or similarity between the source tasks and target tasks. Accordingly, the transferred knowledge could create a negative impact or a positive impact respectively [11].

Having multiple environments with a high level of semantic similarity would in-directly improve the partial observability by exchanging the learning across the agent's operating environment [14]. Similarly, having multiple actor-critic models operating simultaneously across two semantically similar environments would mitigate RL agent's issues associated with effective exploration, raining samples and the training time required to reach an optimized performance level [15].

A. Actor Module

The actor controls how a DRL agent should behave in an environment by taking each individual state of the environment as its input and outputs the best action. This way, an actor controls the behavior of the agent behaves by learning the optimal policy. Policy-based algorithms like Policy Gradients and REINFORCE try to find the optimal policy directly without the Q -value as the intermediate step. Actor module will be implemented as a neural network having a function approximation capability to evaluate the best action to be taken in each state

B. Critic Module

The critic module, on the other hand, plays its role in the evaluation of the merit of an action by computing the value function . . The does this by following a value-based approach with a help a neural network-based function approximation. The result is that the overall architecture will learn to play the game more efficiently than the two methods separately.

C. Actor-Critic Methodology

Unlike some simpler techniques which are based on either value-iteration (such as Q-learning) methods or policy-gradient (PG) methods, the actor-critic methodology algorithm attempts to combine the best parts of both the methods, which are the algorithms that predicts both the value function $V(s)$ as well as the optimal policy function $\pi(s)$. The learning agent uses the value of the value function generated by critic module to update the optimal policy function created by the actor. In this context, the policy function refers the probabilistic distribution of the action space. To be exact, the learning agent determines the conditional probability $P(a|s;\phi)$ which otherwise means the parametrized probability that the agent chooses the specific action a while in state s.

D. A3C Model – Asynchronous Advantage Actor-Critic Model

The agents (also known as individual worker agents) are trained in parallel in their respective instances of the environment and update a global network periodically, which holds shared parameters received from all the individual workers. The updates are not happening simultaneously and that's where the asynchronous comes from. After each update, the individual worker agents reset their parameters to those of the global network and continue their independent exploration and training for n steps until they update themselves again. By following this approach, the information flows not only from the worker agents to the global network but also between the individual agents as each agent resets his weights by the global network, which has the information of all the other agents. The Fig. 2 represents the actor-critic model adopted by the A3C algorithm.

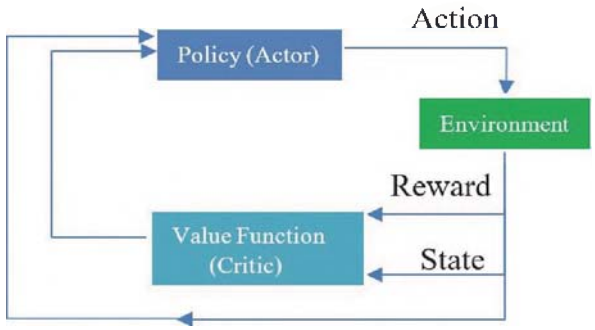

Fig. 2. Actor-Critic Model

By using the value of Advantage instead, the agent also learns how much better the rewards were than its expectation. This gives a new-found insight to the agent into the environment and thus the learning process is better. The advantage metric is given by the following expression.

$$\text{Advantage: } A = Q(s, a) - V(s) \quad (1)$$

where Q refers to the Q value calculated by the critic module based on the actual reward and TD error following an actor's policy based chosen action.

IV. Prototype Implementation

This section details the methodology adopted towards the prototype implementation of the proposed hybrid multi-task learning model that is based on the A3C algorithm. Throughout the implementation, prototype was tested with various games under the Atari 2600 environment provided within the OpenAI Gym [16]. The Gym library is a toolkit made by OpenAI for developing and comparing RL algorithms. First stage of the multi-tasking model was constructed by adopting the A3C algorithm for the gaming environment Breakout-v0. The high - level architecture of the model is based on the actor-critic methodology. In our context, actor is a neural network that parameterize the policy (π (a | s) and critic is another neural

network that parameterize the value function V(s). The policy network outputs the policy (π) based on which actor chooses an action within the environment, and value network outputs the value function(V). Each of these networks have their own respective weights which are often represented by notations such as Θp and Θv.

$$\pi(a \mid s, \Theta p) = \textit{Neural Network}\ (\text{input: } s, \textit{weights}: \Theta p) \quad (2)$$

$$V(s, \Theta v) = \textit{Neural Network}\ (\text{inputs, } \textit{weights}: \Theta v) \quad (3)$$

In order to accommodate and handle graphic intense Atari 2600 gaming environment neural network-based model was used for the validation. At the root level, this environment will employ a pair of convolutional neural network (CNN) models to implement both actor and critic modules for a single worker. There will be multiple instances of the CNN class objects to implement the multiple -worker threads used within the multi-task model. In a similar fashion, global network also deployed as a pair CNN to support the implementation of actor-critic modules at the global network level.

Fig. 3 outlines the high-level architecture view of the multi-task model having three worker threads of execution coordinated and managed by a global network. Each of these individual blocks are made up of a pair of CNN networks, each for actor (policy generating module) and critic (value function generating module) modules. In another words, A3C utilizes multiple worker agents attacking the same game environment while being initialized differently. This indirectly points that each of these agents start at a different point in their environment so they will go through the same environment in different ways to solve the same problem.

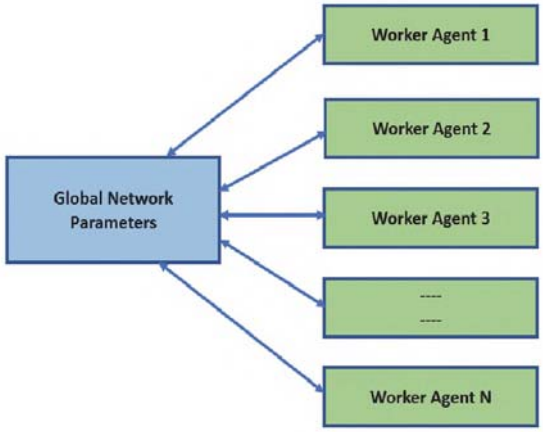

Fig. 3. A3C Multi-Task Worker Agents Model

Within the A3C based multi-task worker agent environment, each of the individual worker agents are managed by the global network directly. Under this scheme, initially each of the worker is reset with parameter values shared by the global network, later on worker interacts with its own individual copy of the environment. Even though each of the worker agents are operating within the same game environment, they are being initialized differently. This gives an opportunity for each of these agents to start at a different point in their environment.

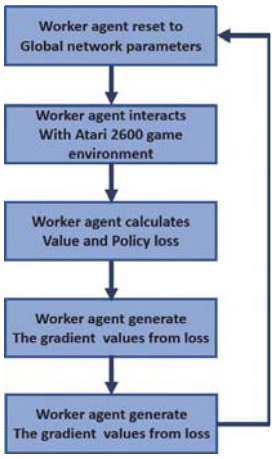

Fig. 4. Training Workflow of Worker Agent

During the course of its operation, each worker agent plays a fixed number of game episodes and calculates value and respective policy loss. As these modules, both actor and critic, being implemented using the neural network, gradient values are calculated from the losses incurred during its operation. These gradient values will be shared with global network after the work agent finishes a fixed number of game episodes. The Fig. 4 shows the training workflow of each agent with global network. The algorithm that drives operation of A3C multi-task worker agents' model is given below.

while not done:

$$\boldsymbol{a = sample\ an\ action\ a \sim \pi_\theta(a|s)}$$

$$\boldsymbol{s`, r, done = Perform\ action\ a - env.step(a)}$$

$$\boldsymbol{G = r + \gamma V(s`)}$$

$$\boldsymbol{L_p = -(G - V(s))\log\left(\pi(a|s, \theta_p)\right)}$$

$$\boldsymbol{L_v = (G - V(s))\text{^}2}$$

$$\boldsymbol{\theta_p = \theta_p - \alpha * d_{L_p} / d_{\theta_p}}$$

$$\boldsymbol{\theta_v = \theta_v - \alpha * d_{L_v} / d_{\theta_v}}$$

During the operation, each of the worker agent loop through each step of the game, and samples the action and update the weights of both the neural networks- actor and critic. Algorithm runs until a preset number of episodes of game are played, wherein initially an action a is sampled from the actor (policy network). Further on, upon completion of that action respective reward *(r)* and new state *s'* are calculated. Based on the new state reached, total discounted future return *(G)* is calculated by applying the discount factor (gamma). Based on this each of the individual neural networks calculates its policy loss *(Lp)*, and value loss (*Lv*) [17]. Further on, neural network uses the gradient descent to update the respective network weights (*Θp* – policy network weight and *Θv* – value network weight) to minimize the

loss. At the root level, this environment will employ a pair of CNN based models to implement both actor and critic modules for a single worker. Similarly, global network is also deployed as a pair of CNN to support the implementation of actor-critic modules at the global network level. These neural network models act as a function approximator by processing each screen shot of the game as it's input. During the experiments, the evaluation of the multi-task learning model was performed on a cloud server machine having multiple CPU cores and a GPU capability.

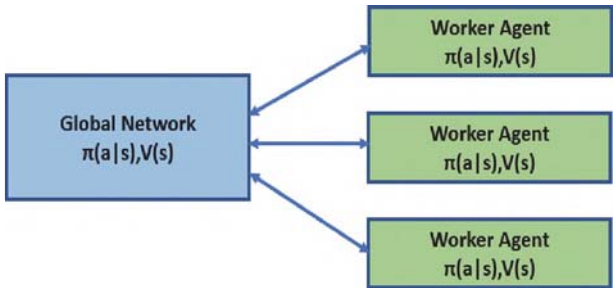

Fig. 5. Parameter Sharing of Global Network with Worker Agents

The Fig. 5 shows the ecosystem of the multi-task worker agents operating under guidance of a global network, which itself is a pair of neural networks, one for the policy network or actor module and another for the value network or critic module. Now every so often, this global network is going to send its weights to a set of worker agents each with their own copy of policy and value network. Further on each of these individual worker agents will be playing a few episodes of game under its environment using its network weights from own experience. From its own experience, each worker agent can calculate its own policy gradient updates and value updates. Knowledge of these updates will be limited to only these individual worker agents. Eventually worker agents send its gradient values to the global network so that global network can update its weights accordingly. Every so often the global network gives its new updated parameters back to its working agents, so worker agents are always working with a relatively recent copy of the global network. In this working model, worker threads play episodes of games under its respective environments, find the errors and calculate the update gradients which will be shared with global network on a regular basis.

V. Experiments and Evaluation Results

A3C provides a multi-threaded and asynchronous approach to deep reinforcement learning [18]. This algorithm gives the capability to have a model to be trained with multiple, different explorations of a single target task, providing data sparsity and avoiding the use of memory replay [19]. Evaluation of the hybrid multi-task learning model will be conducted with the Atari 2600 environment.Following gaming environments will be used for the evaluation of the proposed model, BeamRider-v4, Breakout-v0, BeamRider-v0, and Pong-v0. During the first stage of evaluation, the performance of the RL agent will be measured individually on each of these gaming environments to generate the initial test statistics. Fig. 6 to Fig. 9 show the test results for A3C algorithm based the multi-task worker model for the each of the ATARI 2600 games. The graphs were generated within TensorBoard (TensorFlow's visualization toolkit), the numbers on x-axis represent the global steps in millions (taken by the agent), and numbers on the y-axis represent the rewards (game score). The same convention applies to Figures 11 to 14.

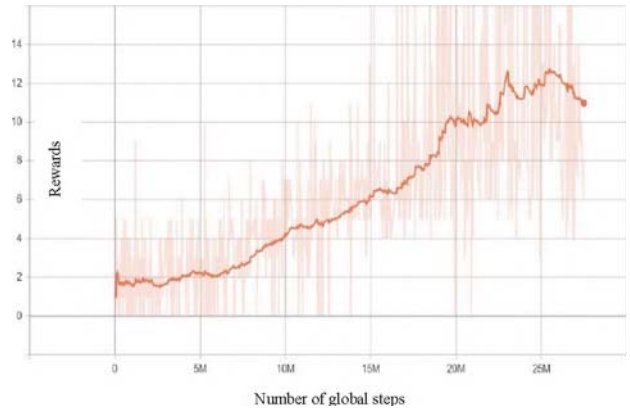

Fig. 6. Breakout-v0 Test Results with 8 Workers

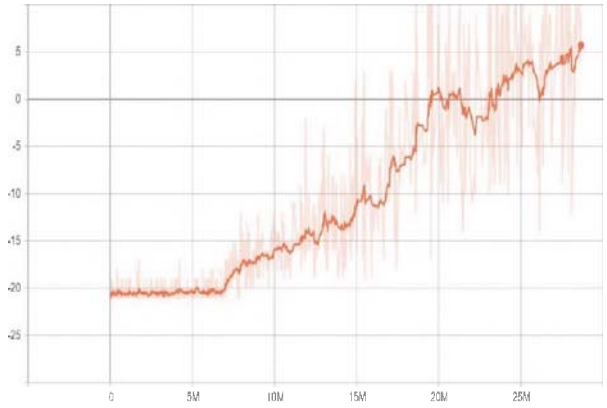

Fig. 7. Pong-v0 test Results with 8 Workers

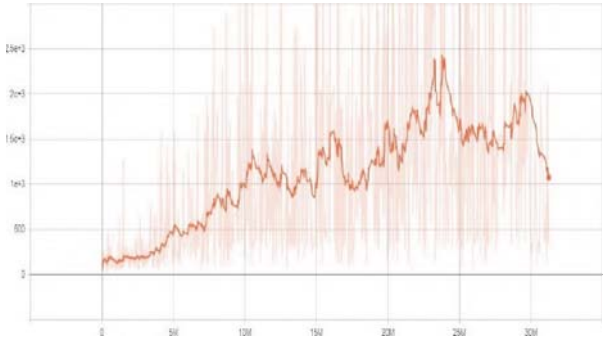

Fig. 8. DemonAttack-v0 Test Results with 8 Workers

These tests were conducted by using a multi-task environment having eight worker threads, with each one having its own

individual copy of environment but different from one another in terms of the view of the gaming environment.

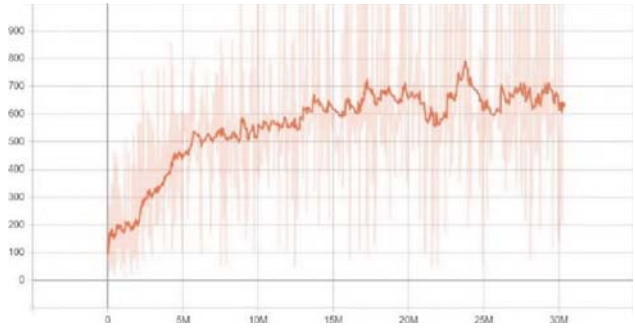

Fig. 9. SpaceInvader-v0 Test Results with 8 Workers

In the next stage, multi-task learning model will be applied to both of these two gaming environments simultaneously by combining the learnings generated by each of the individual and distinct learning environment with the help of global network. Fig. 10 shows the prototype for the same. This approach would validate the impact of hybrid multi-task model's performance when multiple environments with high level of semantic similarity are executed in parallel under the same global network

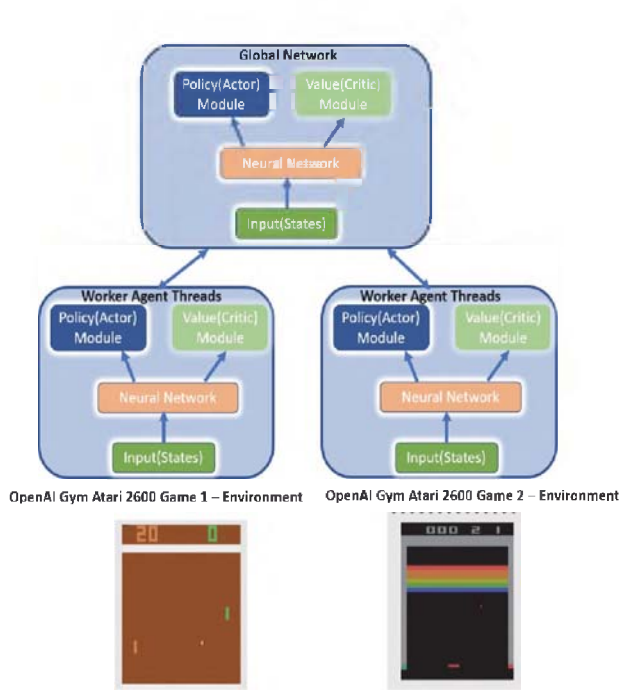

Fig. 10. Hybrid Multi-task Worker Model with Breakout-v0 and Pong-v0

For the purpose of experiments with hybrid multi-task learning model, we created two test pairs, where in each test pair will have two games each with high level of semantic similarity. Accordingly, test pair-1 was created with Breakout-v0 and Pong-v0 games, similarly test pair-2 was created with DemonAttack-v0 and SpaceInvader-v0 games. The Fig. 11 to Fig. 14 show the respective test results obtained.

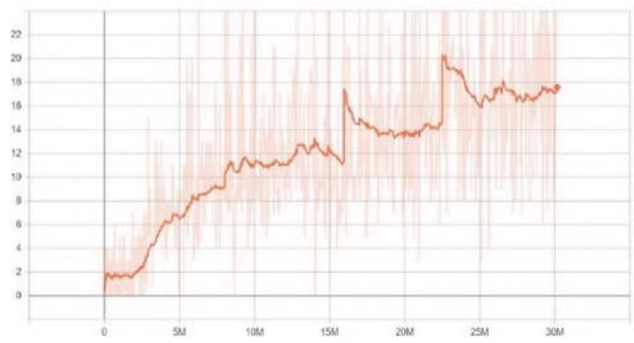

Fig. 11. Breakout-v0 Test Results with Hybrid Multi-task Learing Model

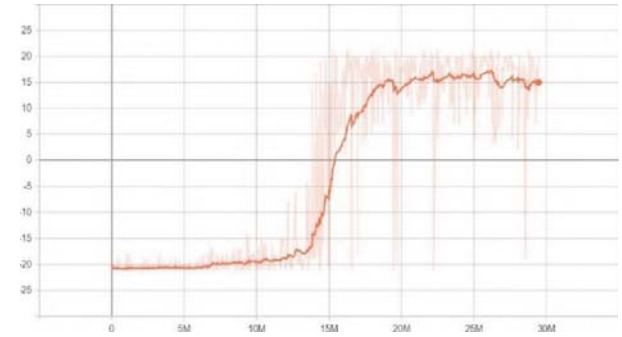

Fig. 12. Pong-v0 Test Results with Hybrid Multi-task Learing Model

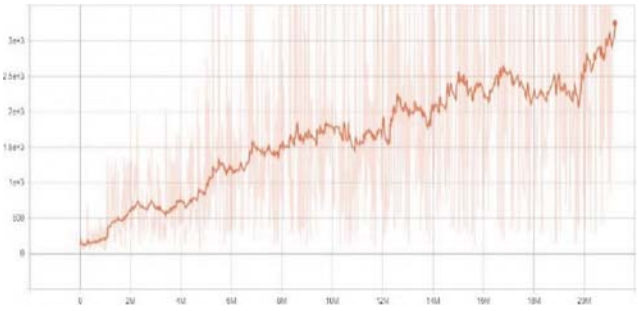

Fig. 13. DemonAttack-v0 Test Results with Hybrid Multi-task Learing Model

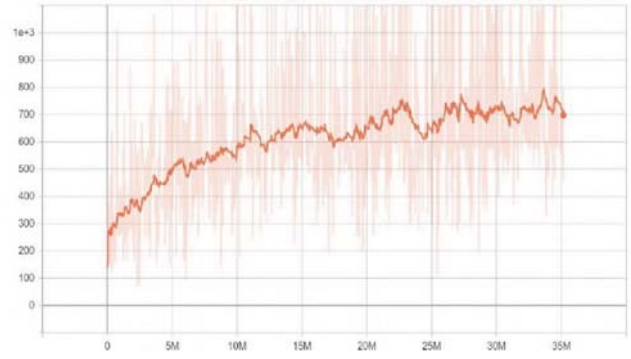

Fig. 14. SpaceInvader-v0 Test Results with Hybrid Multi-task Learing Model

In contract to the first stage of testing, as indicated by Fig. 6 to Fig. 9, where gradients shared to the global network by worker agents are all of same type, in the hybrid environment we have two different types of worker threads. As it is anticipated, performance of individual games under the hybrid environment were not on par with standalone performance results obtained with first stage of testing. As and when the games progress, we could see the impact of positive knowledge sharing among these two tasks that are trained jointly. Due to the semantic similarity among them, updates shared by the global network could mitigate some of the key challenges associated with partial observability in comparison to a single game-based environment. Based on the test results obtained with each of the sets that we mentioned earlier, we could see that each of the games under each test set could boost its performance over the course of the training. By this we are able to establish that our hybrid multi-task model is able to learn multiple similar gaming tasks simultaneously without degradation in performance for any one of the individual gaming tasks

VI. Conclusion and Future Work

In this paper, we introduced a hybrid multi-task learning model for optimizing the performance of DRL agents. We present how this model is able to combine the multi-task learnings from multiple different DRL agents operating within two different but semantically similar environments that are running with related tasks. Initial stage experiments are conducted by applying the DRL algorithm A3C to Atari 2600 gaming environments to draw the initial results. With the results obtained from the experiments, we could establish that hybrid multi-task learning model could learn multiple similar gaming tasks running simultaneously, without causing any degradation in performance for any one of the agents associated with those tasks. The semantic similarity of the related tasks running in two different environments is the most vital factor to reduce the challenges posed in terms of the possible negative knowledge transfer. For future work, we plan to conduct the experiments with the hybrid A3C model with more complex gaming environments having higher number of worker threads. Additionally, we also would like to investigate on the steps to mitigate the impacts of negative knowledge transfer and catastrophic forgetting in deep reinforcement multi-task learning under a GPU cloud-based machine environment to draw strong conclusions on hybrid multi-task learning model.

References

[1] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in neural information processing systems*, 1996, pp. 1038-1044.

[2] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning,* Vols. 8(3-4), pp. 279-292, 1992.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland and G. Ostrovski, "Human-level control through deep reinforcement learning," *nature,* vol. 518, no. 7540, pp. 529-533, 2015.

[4] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928-1937.

[5] A. Mujika, "Multi-task learning with deep model based reinforcement learning," *arXiv preprint arXiv:1611.01457,* 2016.

[6] 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. 4496-4506.

[7] E. Parisotto, J. L. Ba and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342,* 2015.

[8] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734.*

[9] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley and I. Dunning, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *arXiv preprint arXiv:1802.01561,* 2018.

[10] M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt and H. van Hasselt, "Multi-task deep reinforcement learning with popart," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3796-3803.

[11] N. Vithayathil Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics,* vol. 9, no. 9, p. 1363, 2020.

[12] T. T. Nguyen, N. D. Nguyen and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE transactions on cybernetics,* 2020.

[13] E. Parisotto, J. L. Ba and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342,* 2015.

[14] D. S. Chaplot, L. Lee, R. Salakhutdinov, D. Parikh and D. Batra, "Embodied Multimodal Multitask Learning," *arXiv preprint arXiv:1902.01385,* 2019.

[15] T.-L. Vuong, D.-V. Nguyen, T.-L. Nguyen, C.-M. Bui, H.-D. Kieu, V.-C. Ta, Q.-L. Tran and T.-H. Le, "Sharing experience in multitask reinforcement learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, 2019, pp. 3642-3648.

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

[17] Lazyprogrammer, "https://github.com/," GitHub, 24 08 2017. [Online]. Available: https://github.com/lazyprogrammer. [Accessed 21 10 2020].

[18] N. D. Nguyen, T. T. Nguyen and S. Nahavandi, "A Visual Communication Map for Multi-Agent Deep Reinforcement Learning," *arXiv preprint arXiv:2002.11882,* 2020.

[19] J. Zou, T. Hao, C. Yu and H. Jin, "A3C-DO: A Regional Resource Scheduling Framework based on Deep Reinforcement Learning in Edge Scenario," *IEEE Transactions on Computers,* 2020.