electronics

MDPI

Review

A Survey of Multi-Task Deep Reinforcement Learning

Nelson Vithayathil Varghese * and Qusay H. Mahmoud

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

* Correspondence: nelson.vithayathilvarghese@ontariotechu.net

Received: 13 July 2020; Accepted: 19 August 2020; Published: 22 August 2020

Abstract: Driven by the recent technological advancements within the field of artificial intelligence research, deep learning has emerged as a promising representation learning technique across all of the machine learning classes, especially within the reinforcement learning arena. This new direction has given rise to the evolution of a new technological domain named deep reinforcement learning, which combines the representational learning power of deep learning with existing reinforcement learning methods. Undoubtedly, the inception of deep reinforcement learning has played a vital role in optimizing the performance of reinforcement learning-based intelligent agents with model-free based approaches. Although these methods could improve the performance of agents to a greater extent, they were mainly limited to systems that adopted reinforcement learning algorithms focused on learning a single task. At the same moment, the aforementioned approach was found to be relatively data-inefficient, particularly when reinforcement learning agents needed to interact with more complex and rich data environments. This is primarily due to the limited applicability of deep reinforcement learning algorithms to many scenarios across related tasks from the same environment. The objective of this paper is to survey the research challenges associated with multi-tasking within the deep reinforcement arena and present the state-of-the-art approaches by comparing and contrasting recent solutions, namely DISTRAL (DIStill & TRAnsfer Learning), IMPALA(Importance Weighted Actor-Learner Architecture) and PopArt that aim to address core challenges such as scalability, distraction dilemma, partial observability, catastrophic forgetting and negative knowledge transfer.

Keywords: reinforcement learning; deep learning; neural networks; transfer learning; multi-tasking; deep reinforcement learning; actor-mimic; policy distillation; distraction dilemma; exploration

1. Introduction

Reinforcement learning (RL) has established its position as a vital technology in domains such as robotics and intelligent agents [1]. The major objective of reinforcement learning is to address the problem of how reinforcement learning agents should explore their environment, and thereby learn to take optimal actions to achieve the highest reward (end goal) while in a given state [2]. At any given timestep T and state S, the optimal goal of an agent is to deduce a policy π, basically a mapping from a state to an action, which maximizes the accumulated future reward over a given horizon in the environment. Most of the earlier researches conducted towards the RL agent's performance optimization were based on the concept of linear function approximations to enhance RL algorithms' generalization ability under complex environments [3]. Reinforcement learning has cemented its position as one of the machine learning paradigms that deal with an RL agent's behavior pattern within an environment, in terms of how it should act in that environment, so as to maximize the associated future reward. In comparison to the performance of machine learning systems based on contexts such as supervised and unsupervised learning, oftentimes performance of traditional RL agents was not optimal. This was primarily due to difficulties related to deducing the optimum

policy out of the massive state-space associated with the environment of RL problems. Given the fact that the field of reinforcement learning has been very well established over the last few decades, the application of deep learning in the field of reinforcement learning has given a new dimension to it. Especially, the inception of deep learning has given a new dimension to the representation learning approach in comparison to the traditional feature engineering-based methodology. In general, the feature engineering process used to be carried out manually and oftentimes considered to be quite time-consuming, less accurate, and incomplete. Deep neural networks with deeper hidden layer architecture can automatically learn the representations from data inputs to uncover the underlying compositional hierarchies with high accuracy. This is primarily due to the distributed representation methodology used by deep learning, which unveils the interdependencies between the input values and associated features. This new approach automates the representation learning with deep learning and provides an end-to-end learning. In the recent past, there has been significant progress in many of the research areas related to deep learning. The most recent research advancements that happened within the neural network class, such as convolutional neural networks (CNN), have especially helped to achieve impressive results with applications related to vision and image processing [4].

With the above-stated developments in the deep learning arena, in recent years there has been significant advancement in the field of deep reinforcement learning (DRL). As a result of these developments, deep RL agents have been applied to various areas, such as continuous action control, 3D first-person environments, and gaming. Especially in the field of gaming, DRL agents surpassed the human-level performance on classic video games, such as Atari, as well as board games, such as chess and Go [5]. Notably, reinforcement learning played a vital role in the development of Google DeepMind's AlphaGo program, which beat the world's top-level Go player in 2016 [6]. Similarly, in the year 2015, a model-free approach based on the deep Q-network (DQN) method combined a classical RL method with deep convolutional neural networks (CNN) and learned to play several Atari 2600 games with above-human level performance. While the performance results on the aforenoted tasks were impressive, it was predominantly based on a single task performance approach, wherein an RL agent is trained on each task or gameplay individually.

Despite the impressive results with a single-task based approach, the RL agent was found to be less efficient in environments that are more complex and richer in data, such as 3D environments. One of the directions to improve the efficiency of the RL agent is by multi-tasking-based learning. One of the well-known definitions states that "Multi-task Learning is an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias" [7]. During multi-task learning, a set of closely related tasks will be learned concurrently by individual agents with the help of a deep reinforcement algorithm, such as A3C (asynchronous advantage actor-critic). With this approach, at regular intervals, the neural network parameters of each of these individual agents will be shared with a global network. By combining the learning parameters of all individual agents, the global network derives a new set of parameters that will be shared with all agents. The key objective of this approach is to enhance the overall performance of the RL agent by transferring the learning (shared knowledge) among multiple related tasks running within the same environment. One of the key aspects of multi-task learning is that the RL agent should develop a library of general knowledge and learn general skills that can be shared as well as used across a variety of related tasks. Additionally, there should be a balance between the resource necessities of multiple tasks competing for the available resources within the learning system.

The majority of the bottlenecks that slow down the advancement of multitask learning within deep reinforcement learning are related to factors such as effective system resource management within the multiple tasks and scalability [8]. Scalability is closely related to the two major weaknesses of the RL algorithms. Firstly, training of RL algorithms usually takes a considerable amount of time and also needs more data samples to converge to an acceptable result. Secondly, an RL agent that is trained on a specific task can only be used with the same task [9]. By considering the above two drawbacks of typical RL algorithms, multi-task learning within the deep reinforcement domain must

be able to provide better results in comparison to single-task learning of the individual tasks. Similarly, there should be a proper balance established between the resource requirements of multiple tasks that often compete for the limited resources of a single learning system used by the RL agent. This is more closely related to the fact that, from an agent's perspective, the importance of a particular task increases with the scale of the reward observed in that task, and it can vary arbitrarily across tasks during multi-task learning.

One of the most widely accepted multi-task learning methodologies within reinforcement learning is named parallel multi-task learning, in which a single RL agent is used to master a group of diverse tasks [10]. The core idea behind this approach is that architecture used by the deep reinforcement learning models uses a single learner (critic) combined with different actors. Further on, each of the individual actors generates their learning trajectories (which are a set of parameters) and sends them to the learner either synchronously or asynchronously. After this stage, each of the actors retrieves the latest set of policy parameters from the learner before the next learning trajectory begins. With this approach, learning from each of the individual tasks will be shared with every other task, which internally improves the overall learning momentum of the RL agent.

Contributions

The primary objective of this survey is to present details on the integration of multi-tasking featuring deep reinforcement learning as well as the state-of-the-art in the deep reinforcement learning (DRL) arena with a parallel multi-tasking feature. To this end, this survey has been organized in such a way that it starts with the details of canonical reinforcement learning and various challenges associated with it. As part of the survey efforts, multiple literature survey papers were examined that are predominantly related to the foundations of deep reinforcement learning and its applicability. One of these surveys analyzed the foundations of reinforcement learning, which covers core elements such as dynamic programming, temporal difference learning, exploration vs. exploitation, function approximation, and policy optimization [11]. A second survey analyzed how deep reinforcement learning could revolutionize the field of artificial intelligence and pave the steps towards the development of intelligent agents that can understand the visual world in a better way [12]. Additionally, this survey also attempted to give an outlook on the various related algorithms and the distinct advantages of deep reinforcement learning within the context of reinforcement learning. Subsequently, a couple of additional survey papers were examined, which mainly focused on the applicability of deep reinforcement learning in specific domains such as autonomous driving [13]. Additionally, another area that was examined was the applicability of deep reinforcement learning to the improvement of target-driven visual navigation in indoor scenes. Further on, the survey touched on the basic aspects of multi-tasking, and how this particular aspect has been related to the deep reinforcement learning (DRL) domain [14]. Following this, special attention was paid to surveying various methodologies that were attempted for the implementation of the multi-tasking aspect within the reinforcement learning arena. In light of this, multiple research papers were analyzed to survey the multi-tasking methodologies adopted within the reinforcement learning arena. One such effort was based on a research paper that detailed sparse multi-task reinforcement learning [15]. A second survey effort was focused on multi-agent reinforcement learning in Markov games, based on a study that investigated multi-agent learning in complex task environments [16].

Subsequently, the survey focus was directed toward addressing multiple research challenges associated with the application of multi-tasking in deep reinforcement learning; it then finally examined the three major state-of-the-art solutions that are implemented to overcome some of those challenges. Throughout this literature survey, the key focus remained on investigating various methodologies that are related to multi-tasking-related aspects. Analysis of each one of these methodologies was carried out by focusing on its applicability in the context of deep reinforcement learning. There is a growing amount of literature that is specially focused on multiple aspects within deep reinforcement learning, including multi-tasking [17]. As part of investigating such recent survey efforts from the literature, multiple

survey papers were analyzed. One of the surveys analyzed predominantly focused on the integration of multi-agent-based learning features into deep reinforcement learning, often referred to by the term multiagent deep reinforcement learning (MDRL) [18]. There is research literature that examined the applicability of transferring learning across multiple reinforcement learning tasks towards achieving the multi-tasking capability within the deep reinforcement learning. The key focus of this literature was on identifying a framework by which an RL agent can benefit from transfer learning by leveraging experiences gained from previous tasks from a common learning environment [19]. In addition to this, there are also surveys from the literature that examined the applicability of knowledge reuse within the multiagent system under reinforcement learning. The objective of this approach was to find ways of joining the advantages of using transfer learning within a multiagent environment for reinforcement learning problems [20]. A couple of more surveys were also surveyed the recent efforts in this direction, and the core methodology adopted among those was based on multiagent-based learning [21].

In comparison to all the aforementioned surveys analyzed from the literature, our survey is different as we explain here. To this end, the contributions of this paper are:

1. Examining the state-of-the-art within the deep reinforcement learning domain concerning the multi-tasking aspect.
2. Single source reference for implementation details as well as comparison study of three of the major solutions developed to incorporate the multi-tasking aspect with deep reinforcement learning.
3. Provides details on most of the methodologies attempted to bring the multi-tasking feature into the deep reinforcement learning arena under a single survey.
4. Survey of the majority of the research challenges that are being encountered concerning the adaptation of multi-tasking into deep reinforcement learning.

The remaining sections of this paper are structured as follows. Section 2 presents an overview of reinforcement learning (RL), with details provided on various related aspects. The objective of Section 3 is to present details about various approaches used within deep reinforcement learning to incorporate the multi-tasking aspect. Further on, Section 4 focuses on the various key research challenges that act as the major bottlenecks within deep reinforcement learning. Section 5 discusses three state-of-the-art solutions named DISTRAL [22], PopArt [10], and IMPALA [23], which incorporate the multi-tasking aspect into the deep reinforcement learning arena. Additionally, this section also provides a brief comparative study of these three approaches. Finally, concluding remarks are presented in Section 6.

2. Overview of Reinforcement Learning

Reinforcement learning (RL) is one of the machine learning paradigms dealing with sequential decision-making that involves mapping situations to actions in a way that maximizes the associated reward. Within RL ecosystems, the learner, which is also known as an agent, is not explicitly instructed on which actions to take at each timestep t, but instead the RL agent must follow a trial-and-error method to identify which actions generate the most reward. One of the most challenging aspects of the RL is that actions that have already been carried out may affect not only the immediate reward but also the further states and, through that, all subsequent rewards. Reinforcement learning distinguishes itself from other machine learning methods by the above two characteristics—trial-and-error search and delayed reward [1]

2.1. Reinforcement Learning Setup

A standard reinforcement learning setup consists of an agent situated within an environment E, where an agent will be interacting with the environment in discrete timesteps. At each of these timesteps t the agent will be in a state S_t and will be performing an action within the environment. Further on, the environment responds by updating the current state S_t to a follow-up state S_{t+1} with a new timestep $_{t+1}$ and also gives a reward to the agent, indicating the value of performing an action in the preceding state S_t [1]. Figure 1 below represents the standard ecosystem for a reinforcement

learning environment at any given timestep t. By performing multiple actions in a sequential learning manner in a sequence of associated states s, with related actions a, respective follow-up states s′ and rewards r, several episodes of tuples of $<s, a, s', r>$ are generated. At any state S_t, the goal of the agent is to determine a policy π that can create a state to an action mapping, which maximizes the accumulated reward over the lifetime of the agent for that particular state [9].

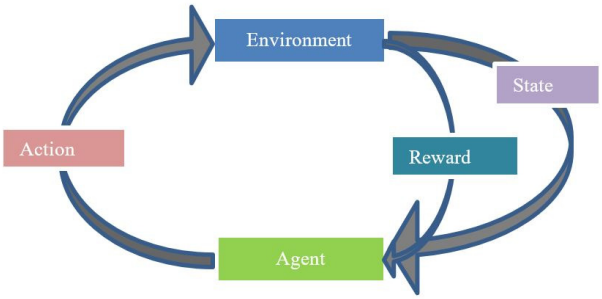

Figure 1. Reinforcement learning ecosystem.

2.2. The Markov Property

Formally, the reinforcement learning environment is considered as the mathematic representation of a Markov decision process (MDP) [24]. MDP consists of the following major components:

- Set of all possible states that an agent can be while in the environment, represented by S.
- Set of all possible actions that an agent can take while in a state s € S, denoted by A.
- Transition dynamics function T defined as $T(s, a, s') = Pr(S_{t+1} = s' | S_t = s, A = a)$. Since the actions are considered part of a probability distribution, here T represents distribution over the possible resulting state by taking a specific action while in a given state s.
- A reward function, R, associated with a state transition by taking a specific action $R(S_t, a_t, S_{t+1})$.
- A discount factor γ $[0,1]$, which will be used for the calculation of discounted future rewards associated with each state transition. Generally, a low discount factor value will be applied for expected future rewards for state transitions leading to the nearby states, whereas a high discount factor value will be applied for rewards associated with actions leading to states that are far from the current state [24].

Reinforcement learning models are denoted by the Markov decision process because often such models make the Markov assumption. The core idea behind the Markov assumption is that if one knows the current state one is in, then the history (sequence of actions and states that took the agent to the current state) does not matter. Going with this key assumption, the core concept underlying each of the RL problems is the Markov decision property—which says that only the current state will have an influence on the next state, and given the current state, the future is independent of the past. In another way, it can be interpreted as any action taken at state S_t can be solely based on the state immediately preceding it, S_{t-1}, but totally independent of all other states $\{S_0, S_1, \ldots\ldots, S_{t-2}\}$ [25]. In the context of RL, the term policy π is used to define a mapping from a state to a related action that is defined over the probability distribution of actions.

This can be denoted as $\pi(s)$: $S -> Pr(A = a|S)$. A policy is considered to be an optimum policy $\pi^*(s)$ for a state s if the specified action taken from that particular state can lead to the maximum expected discounted future reward [24]. In theory, the final objective behind each of the reinforcement learning (RL) agents is to solve the MDP by deducing an optimum policy.

2.3. Key Challenges of Reinforcement Learning

Some of the major challenges related to reinforcement learning (RL) can be summarized as follows:

- By heavily relying only on the reward values, an agent needs to follow a brute-force strategy to derive an optimal policy.
- For every action taken while in a particular state, the RL agent needs to deal with the complexities related to the maximum expected discounted future reward for that action. This scenario is often denoted as the (temporal) credit assignment problem [26].
- With environments with a 3D nature, the size of the continuous state and action pairs can be quite large.
- Observations of an agent from a complex environment heavily depend solely on its actions, which can contain strong temporal correlations.

2.4. Multi-Task Learning

The traditional learning methodology followed in machine learning is to learn one task at a time. Under this methodology, complex and large problems are broken into small and independent subproblems that are learned separately, then eventually all of this learning is combined towards the overall solution of the problem [27]. There could be occasions in which this approach can be less productive, especially when dealing with complex real-world scenarios (such as autonomous driving systems) that have a source of information with a lot of interdependent tasks. For these kinds of situations, if multiple tasks can learn together and then share their knowledge among themselves, eventually that would make the generalization performance of the overall system increase to a greater extent in comparison to the traditional approach explained above. Multitask learning (MTL) is defined as an inductive transfer mechanism with the key objective to improve generalization performance [28]. The core objective behind multi-tasking is to follow a learning-to-learn methodology so as to leverage the domain-related information accumulated by training the individual, related tasks in parallel with a shared representation of the system [7]. In this way, the knowledge that is acquired during each task learning can be utilized and thereby help other tasks be learned better. Eventually, with this approach multitask learning improves the overall generalization performance, which can be applied across many domains including RL, and can be used with different learning algorithms within the RL arena. From the perspective of reinforcement learning, multi-task learning is an approach intended to optimize the performance of an agent under the assumption that performance bottle-neck problems experienced by the agent are drawn from the same distribution. When it comes to deep reinforcement learning, multi-tasking could be applied from various levels, such as single agent–multiple tasks and multiple agents–multiple tasks.

3. Deep Reinforcement Learning with Multi-Tasking

In recent years, deep reinforcement learning (DRL) has emerged as the state-of-the-art in many benchmark tasks as well as real-world problems, due to which a growing level of attention has been paid to various methods for its optimization. The following sections discuss various approaches and techniques developed for multi-task DRL that are presented in related works.

3.1. Transfer Learning Oriented Approach

Before the inception of deep learning into the reinforcement learning arena, most of the early research efforts on the development of the multi-task-oriented algorithm within reinforcement learning attempted to use the assistance from transfer learning. The core idea behind transfer learning is about transferring knowledge across different but related source and target tasks to improve the performance of machine learning (ML) algorithms used for learning the target task. Transfer within the reinforcement learning predominantly focuses on deriving various methods to transfer knowledge from a set of source tasks to a target task. This approach has shown good results when the similarity levels within the source and target tasks were similar. If the similarity level between the source and target tasks is quite high, then the transferred knowledge can be quite easily used by the underlying learning algorithm to solve the target task efficiently. This is due to the reason that under such

a situation, learning algorithms could achieve optimal performance by leveraging the transferred knowledge rather than relying on more data samples for learning the target task. By leveraging the above methodologies, transfer learning methods have been already applied to single agent-based RL algorithms [29].

There were also research attempts related to extending the same methodologies concerning the multi-agent systems, wherein agents interact with other agents acting in the same environment, and then use the knowledge resulting from their actions as well [30]. In general, multi-agent systems are based on a joint policy that the agents learned in the source task (training task), and then use this policy knowledge to formulate the initial policy of the agents in the target task towards the same. Transfer of knowledge is done differently between the source and target tasks with the help of multiple transfer methods, such as instance transfer, representation transfer, or parameter transfer. In each of these methods, underlying transfer algorithms rely heavily on prior knowledge learned when solving a set of similar source tasks and use it as a reference to bias the learning process on any new task.

3.2. Learning Shared Representations for Value Functions

This is an approach quite similar in nature to the transfer learning methodology. This method was developed based on the function approximation capability of neural networks and their application into the reinforcement learning domain [31]. The major factor behind the success of deep neural networks with reinforcement learning was due to deep learning algorithms' key ability to distill meaningfully representations from high-dimensional input states associated with the environment [32]. This key factor scaled up the applicability of RL to more complex environments and scenarios that were previously impossible or demanded a great level of feature engineering [31]. The ability to develop a good abstraction of the environment and the agent's role within that environment are the pivotal factors behind the success of this approach [33]. The core idea behind this approach is based on the assumption that different tasks that an RL agent needs to learn during its life may have a shared structure and in-built redundancy. If these common factors can be abstracted, then it could play a vital role in speeding up the entire learning process. Learning shared representations is a way to achieve this objective through learning robust, transferable abstractions of the environment that generalize over a set of tasks encountered by the agent while in the environment [34].

The value function is one of the key ideas within the reinforcement learning domain, and is being used primarily in conjunction with functional approximators to generalize over large state-action spaces associated with an agent's environment [26]. Value functions are being calculated and used as a key measure to indicate how good a particular state is. Value functions exhibit a compositional structure concerning the state space and goal states [35]. Additionally, earlier researches have shown that value functions can capture and represent knowledge beyond their current goal that can be leveraged or re-used for future learning [36]. By leveraging the state-action value space of common structures shared among different tasks that an RL agent will be handling during its lifetime while in an environment, optimal value-functions can be learned. This can be achieved by accommodating the common structure mentioned above into the popular value iteration and policy-iteration procedures named fitted Q-iteration and approximate policy iteration, respectively.

3.3. Progressive Neural Networks

This is an approach quite similar in nature to the transfer learning methodology. This method was developed based on the function approximation capability of neural networks. One of the major challenges associated with the optimization of multi-tasking learning within the DRL arena was related to leveraging the transfer of learning, and also how to avoid catastrophic forgetting. As a solution to this problem, various researches have been conducted, and one such step forward in this direction is an approach named progressive neural networks. It has the ability to protect itself from catastrophic forgetting and can also leverage prior knowledge with the help of lateral connections to previously learned features. The progressive neural network is a multi-tasking methodology developed by

DeepMind using the concept of lateral features transferring that leverages on neural networks [37]. The key characteristic of the model proposed by this methodology is that it possesses the ability to learn new tasks and also maintain previous knowledge learned with the help of progressive neural networks. The idea of having a continuous chain of progressive neural networks is to facilitate the transfer of knowledge across a series of tasks.

Conceptually, progressive neural networks have been designed with the following goals:

- Have a system with the ability to incorporate prior knowledge during the learning process at each layer of the feature hierarchy;
- Develop a system with immunity to a catastrophic forgetting scenario.

One of the biggest advantages of this approach is that progressive networks have the ability to retain a group of pre-trained models throughout the entire training cycle [37]. In addition to this, progressive networks can also learn lateral connections from the pre-trained model to extract useful features for new tasks. This kind of approach with a progressive nature brings richer compositionality, and also allows easy integration of prior knowledge at each layer of the feature hierarchy. This type of continual learning allows the agents to not only learn a series of tasks that are experienced in sequence but simultaneously possess the ability to transfer knowledge from previous tasks to improve convergence speed [38]. Progressive networks integrate these features into the model architecture where catastrophic forgetting is prevented by instantiating a new neural network (a column) for each individual task that is being solved during an agent's lifetime in the environment. Along this, knowledge transfer is enabled through lateral connections to the list of features from the previously learned columns [37]. At any timestep, whenever a new task is learned, the model adds a new column of knowledge into its existing framework in the form of a new neural network unit. Further on, this new unit will be used during the learning of successive tasks. Each column (neural network unit) will be trained to solve a particular Markov decision process (MDP) [37]. One of the possible downsides associated with this methodology is that it could be computationally expensive due to its growing size as the learning cycle progresses.

3.4. PathNet

PathNet is a multi-task reinforcement learning approach that was developed with the objective of achieving artificial general intelligence (AGI) by combining the aspects of transfer learning, continual learning, and multitask learning [38]. It is based on a neural network algorithm that uses multiple agents that are embedded in the neural network. The objective of each of these agents is to identify which parts of the network to re-use while learning new tasks [7]. Agents are the pathways (also known as genotypes) within the neural network that determine the subset of parameters that are used during the learning process [39]. These parameters, which are used for the forward propagation of the learning process, often undergo modification during the backpropagation stage of the PathNet algorithm. During learning the learning process, a tournament selection genetic algorithm will be used for the selection of pathways through the neural network. Agents execute actions within the neural network and build the knowledge on how effectively existing parameters in the environment of the neural network can be re-used for new actions (tasks). Agents often work in parallel with other agents who are learning other tasks and share parameters among them for positive knowledge transfer; otherwise, they update the disjoint parameters that are causing negative knowledge transfer [39].

A PathNet architecture consists of a deep neural network having L layers, with each layer having M modules. Each of these modules will be a neural network. The integrated outputs of the modules from each of the layers will be passed into the active modules in the next layer. In every layer, there will be a maximum number of modules (typically 3 or 4) that are allowed for each of the pathways [39]. The final layer within the neural network of each of the tasks that are being learned is unique and will not be shared with any other task within the environment. One of the advantages of the PathNet is that with this approach a neural network can quite efficiently reuse existing knowledge instead of learning

from scratch for each task. This feature could be extremely useful in the context of reinforcement learning, where there are numerous interrelated tasks present in state space [39]. Research regarding PathNet have exhibited positive results for the knowledge transfer for binary MNIST dataset (Modified National Institute of Standards and Technology), CIFAR-100 dataset (Canadian Institute For Advanced Research), and SVHN dataset (The Street View House Numbers) supervised learning classification tasks, and a set of Atari and Labyrinth reinforcement learning tasks.

3.5. Policy Distillation and Actor-Mimic

Policy distillation (PD) and actor-mimic (AM) are the two approaches that leverage the concept of distillation towards achieving multi-task deep reinforcement learning.

3.5.1. Policy Distillation

Distillation is an approach related to minimizing computational costs of ensemble methods [40]. An ensemble is nothing but a set of models whose prediction values are combined by following a weighted averaging or voting method [41]. Ensemble methods have been one of the significant research areas in the past decade, and some of the popular ensemble methods include names such as bagging, boosting, random forests, Bayesian averaging, and stacking [41]. Two of the disadvantages associated with most of the ensembles are that they are often large in terms of memory size needed, and slow due to the time required to execute them at run-time. To cope with these disadvantages, the distillation technique was proposed, which is based on a model compression methodology. The key idea used behind this methodology was to compress the function that is learned by a complex model (often an ensemble) into a much scaled-down, faster model that has comparable performance with the original ensemble [41]. Later on, the same methodology was mapped into the neural networks domain [42].

By following the concept of model compression that was explained above, policy distillation can be viewed as a technique used to extract the policy of a reinforcement learning agent. Further on this policy will be used to train a new network that performs at the expert level with a smaller size and with higher efficiency [40]. Furthermore, the same methodology can be extended to consolidate multiple task-specific policies into a single policy for the RL agent. Early researches of policy distillation were done with the reinforcement learning algorithm named DQN (deep Q-network). The policy distillation technique was successfully used for transferring one or more active policies from deep Q-networks to an untrained network. DQN is one of the popular state-of-the-art model-free approach used for reinforcement learning by using deep neural networks, which operates within an environment with discrete action choices. This algorithm was shown to surpass human-level performance on a group of diverse Atari 2600 games [31].

Distillation can be applied both at a single task level (single game policy distillation) as well as a multi-task level as a knowledge transfer method from a teacher model T to a student model S. Under the single task policy distillation, data generation will be done by the teacher network (a trained DQN agent) and further on supervised training will be carried out by the student network. In order to achieve multi-task policy distillation, n different DQN-based single-game experts (agents) are trained separately [40]. After this, these agents individually generate the inputs and targets and store these data in different memory buffers. Further on, the distillation agent learns from these n data stores sequentially.

3.5.2. Actor-Mimic

One of the key aspects of an intelligent agent is its capability to act in multiple environments and transfer the knowledge accumulated from past experiences to new situations. Actor-mimic is such an approach that mainly concentrates on multitask and transfer learning aspects. These capabilities enable an intelligent agent (RL agent) to learn how to act with multiple tasks simultaneously, and then generalize this accumulated knowledge to new domains [43]. In general, actor-mimic can be viewed as a method for training a single deep policy network by using a group of related source tasks. A model

that was trained with this method was found to reach expert-level performance on many games. More importantly, with a significant level of similarity between the source and target tasks, features that are learned during the training of the source tasks can be used well for generalization while training the target tasks [44].

The actor-mimic approach leverages both deep reinforcement learning and model compression techniques to train a single policy network. The objective of such training is to make the network learn how to act in a set of distinct tasks under the guidance of several expert teachers. Further on, representations learned by this deep policy network can be used for generalizing to new tasks with no prior expert guidance. This approach was predominantly tested within the arcade learning environment (ALE) [45]. Often, actor-mimic is treated as part of the larger imitation learning class of methods that are based on the idea of using expert guidance to teach an agent how to act within an environment. Under the imitation learning methodology, a policy will be trained to directly mimic an expert's behavior during sampling the actions from the mimic agent [43].

3.6. Asynchron ous Advantage Actor-Critic (A3C)

A3C (asynchronous advantage actor-critic) is an algorithm that was introduced by DeepMind, which proposed a parallel training approach. As per this methodology, there will be multiple agents (also known as workers) that are executing in parallel on multiple instances of the same environment [46]. These multiple workers running in parallel environments update a global value function in an asynchronous fashion. During the training, at any particular time-step t, all these parallel agents will be experiencing a variety of different states, which almost makes the learning of all agents unique. As a result of this uniqueness factor, A3C provides agents with an effective as well as efficient exploration of the entire state space within the environment [47]. Originally, A3C was an extension of the actor-critic method, wherein there will be two independent neural network components named actor and critic, each with its loss functions. An actor can be considered as a function approximator that guides on how to act, as it is being judged by RL methods, such as Q-learning or in REINFORCE. In both of these methods, a neural network will be computing either a function that leads to a policy or directly calculating the policy itself. The role of the critic is more like evaluating the effectiveness of the policy made by the actor and giving feedback for further enhancement of the policy [46]. Figure 2 is the representation of a typical actor-critic model upon which A3C is based.

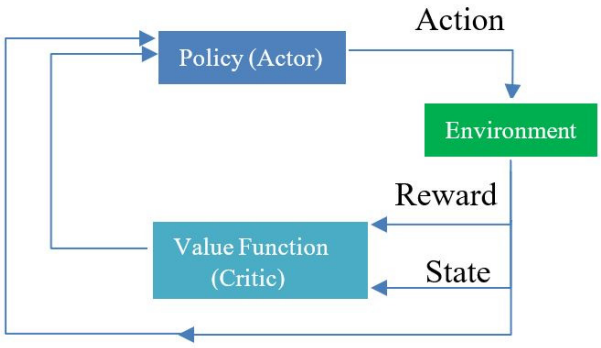

Figure 2. Actor-critic model.

The critic generally estimates the value function that could be either an action-value (the Q value) or a state-value (the V value), whereas the actor updates the policy distribution according to the feedback provided by the critic (such as with policy gradients). For every action taken in a state, there will be an advantage value, A, that will be calculated by subtracting the Q value term from the V value. The advantage value A tells us how much better it is to take a specific action compared to any other possible action at a given state. The advantage actor-critic method falls into two categories: asynchronous advantage actor-critic (A3C) and the advantage actor-critic (A2C). From a multi-tasking perspective, the impact of A3C comes from its aspects of parallelized and

asynchronous architecture to take advantage of parallelism jointly with the actor-critic method [48]. According to this, multiple actor-learners (critics) will be dispatched to individual instantiations of the environment wherein each of these instantiations interacts with the environment and collects their individual experience, and finally asynchronously pushes their gradient updates to a central target network (global network) [46]. Update to this single global network, with which all other different and independent threads are linked, is often done after a determined number of steps (typically after every 20 steps) when gradients are accumulated. By following the methodology of training different task instantiations simultaneously, A3C brings the scarcity factor to the training data and removes the need for a memory replay [31]. Figure 3 depicts the representation of the high-level architecture of the multi-task execution model of the A3C algorithm.

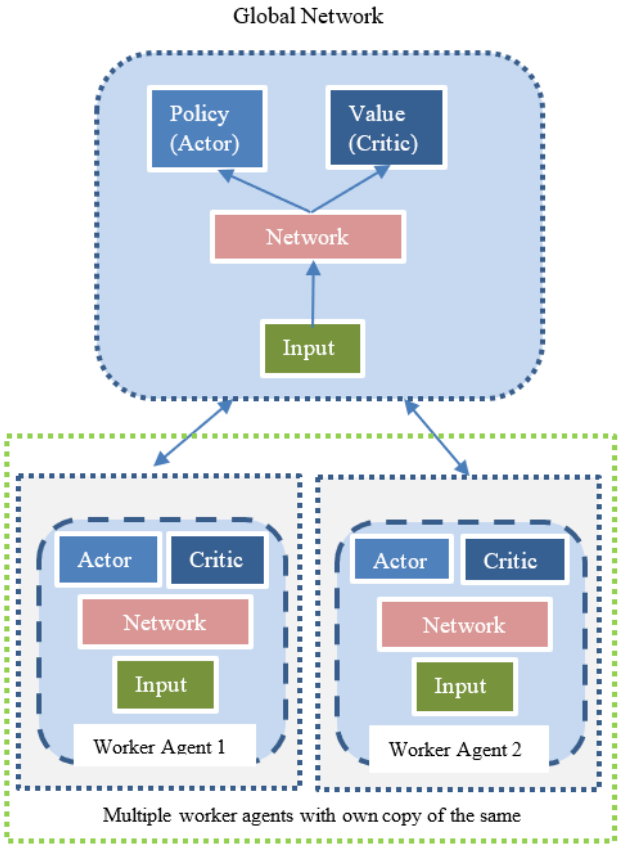

Figure 3. Architecture of the asynchronous actor-critic model.

3.7. Others

This subsection examines various other recent works and frameworks from the literature that are related to reinforcement learning with multi-tasking. One of the recent works analyzed from the literature is related to the use of multi-task reinforcement learning within the area of end-to-end video captioning. Generally, video captioning is being treated as one of the challenging benchmark tasks within computer vision, where traditional end-to-end (E2E) learning is less effective as its performance is often impacted by a factor such as hardware limitations and overfitting [49]. As a mitigation strategy to overcome those limitations, a multitask reinforcement learning-based approach was adopted to train an end-to-end (E2E) video captioning model. The core idea behind this approach is based on the actions of mining and constructing as many effective tasks as possible (which internally represent attributes, rewards, and the captions) from the input video, which will eventually use the search space for the end-to-end (E2E) reinforcement learning network to generate the video captioning model [49]. This model functions as a multitask end-to-end network that combines the multisampling reinforce

algorithm to generate video captions. Empirically, it has been proven that this model can deliver better results than existing methods for video captioning.

The second work analyzed from the literature is about an embodied multimodal multitask learning model that uses the power of deep reinforcement learning. Nowadays deep reinforcement learning is extensively used to train embodied AI agents that interact with a 3D environment. As they interacts with the environment, they receive a first-person view of the environment and further take various navigational actions. Embodied AI navigation agents based on deep reinforcement learning have proven to be quite successful in deriving policies for various multimodal tasks. Some of these tasks include semantic goal navigation and embodied question answering [50].

Another popular framework that was surveyed is MARES, which stands for multitask learning algorithms for Web-scale real-time event summarization. Real-time event summarization involving very large document streams is being considered as a huge research challenge within the natural language processing domain. The key characteristics of MARES are linked with its power of leveraging supervised deep neural networks as well as a reinforcement learning algorithm to make the effective representation learning of documents [51]. MARES can be viewed as a multi-task system, wherein a document modeling module will be shared across tasks and consists of the following two key components: (i) a prediction classifier based on a hierarchical LSTM model that will be used for learning the representations of queries and documents; (ii) with the assistance of a reinforcement learning algorithm, there will be a document filtering model that learns to maximize the long-term rewards [51].

4. Research Challenges

The following section gives an outline of the various challenges associated with multi-task learning within the deep reinforcement learning environment.

4.1. Scalability

Incorporating multi-task learning is being considered as one of the major challenges of artificial intelligence and deep reinforcement learning in particular. The key bottleneck behind this challenge is the scalability factor [8]. This can be linked with one of the key weaknesses of typical RL algorithms. Typically, RL algorithms need more training data samples as well as relatively long training time to converge into an acceptable result [9]. From the multi-tasking context, it should not take N times as many samples or training times to learn N different tasks. The RL agent that possesses the multi-task learning capability should be able to leverage the knowledge acquired from individual task learning and should be able to transfer it across other task learning processes.

4.2. Distraction Dilemma

One major challenge concerning multi-tasking within deep reinforcement learning is related to establishing a balance between the needs of multiple tasks within the environment competing for the limited resources of a single learning system (environment). This is related to the probability of learning algorithms often getting distracted (often called the distraction dilemma) by a few tasks from the set of multiple tasks to solve [10]. Typically, this situation demands a multi-task reinforcement learning (MTRL) system to build the immunity against the distraction dilemma and establish balance concerning mastering individual tasks against the ultimate goal of achieving better generalization. The prime reason behind this distraction scenario occurs because some of the tasks appear more salient to the learning process due to the associated density or magnitude of the rewards for such tasks (in-task rewards) [10]. At different timesteps during the lifecycle of a RL agent within an MTRL system, agents are going to be met with tasks that appear to be more salient to the learning process. This prompts the algorithm to give more attention to such tasks and then focus on those salient tasks at the expense of generality by giving less importance to other tasks in the multi-tasking environment. This indeed

causes the multi-task reinforcement learning (MTRL) agents to often focus on the wrong tasks by ignoring or giving less importance to a few of the other tasks.

4.3. Partial Obeservability

In many of the real-world scenarios, an observation by the RL agent might only capture a small part of the full environment state, and hence the agent might not have the scope to observe the entire environment [8]. With this partial observability factor in the background, the agent still needs to take an optimal action for each of the states. This demands that the agent remember not only the current observation but also the past observation to make the optimal decision on actions. When the state-action space is big, then this challenge additionally involves the challenge to learn and remember a compact representation of the environment with the most salient details from the environment.

4.4. Effective Exploration

In general, reinforcement learning is based on a brute-force method, wherein the RL agent learns by following a continuous trial-and-error-based learning. From the reinforcement learning context, it is described by the terminologies of named exploration and exploitation. RL agents learn by following this method and attempt various actions possible from a particular state to evaluate the optimal action that gives the highest cumulative future reward [8]. Often, a higher level of exploration could be difficult to achieve while attempting to apply reinforcement learning in real-world problems, such as self-driving [17]. Even though methods such as imitation learning and mimicking the desired behavior (simulators) can be used to overcome this, such a trained model often cannot achieve a higher level of generalization while applying it in real environments.

4.5. Catastrophic Forgetting

One of the long-standing goals behind the idea of multi-tasking within the deep reinforcement domain is not only training the agents to learn a series of tasks that are experienced in sequence but also giving them the capability to transfer knowledge from previous tasks to improve convergence speed while training with new tasks [38]. The key bottleneck in achieving the latter goal is a scenario known as catastrophic forgetting (catastrophic interference). This is a scenario related to continuous learning in which deep neural networks will have the probability (tendency) to abruptly lose the information that was learned from a previous task (task A) as information relevant to another task (current task B) is incorporated. Often, this situation arises when the agent is trained sequentially on multiple tasks within an RL environment. The key reason behind the situation is due to the changes in the network parameters (weights) that are related to task A getting overwritten to meet the objectives for task B [38]. This phenomenon is considered to be the biggest barrier to creating artificial general intelligence (AGI), as it negatively impacts the capacity for continual learning. Thus, it is quite important that multi-tasking deep reinforcement learning (MTDRL) agents should be immune to catastrophic forgetting.

4.6. 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 transfer learning in a multi-task context is that transferring knowledge accumulated from learning from a set of source samples may improve the performance of an intelligent agent while learning the target task [43]. However, this knowledge transfer could impact the overall learning progress and performance of the agent either positively or negatively. If there is a considerable difference between the source tasks and target tasks, then the transferred knowledge could create a negative impact. Due to the aforementioned factor, negative knowledge transfer is considered to be one of the key challenges that are suffered by multi-task deep reinforcement learning (MTDRL) methodologies, such as policy distillation [40]. In the following

section, we discuss various solutions that have already been developed to add multi-tasking features into the deep reinforcement learning arena.

5. Review of Existing Solutions

A significant part of the research efforts conducted within the area of multi-task deep reinforcement learning (MTDRL) is by research organizations such as DeepMind and OpenAI. As a result of such research efforts, three major MTDRL solutions, namely DISTRAL, IMPALA, and PopArt have been designed.

5.1. DISTRAL (DIStill & TRAnsfer Learning)

The common norm that was considered to be the baseline for multi-tasking in the reinforcement learning was based on the approach to follow a transfer-oriented methodology, such as sharing the neural network parameters across related tasks in the environment [52]. Often, this approach met with bottlenecks, such as negative knowledge transfer scenarios, and ambiguity on how to design the reward system for various tasks. In a multi-task environment, the reward system should be designed such that there would not be a scenario in which one single task will be leading the learning of the shared model. DISTRAL is a new approach that was developed for multi-task training by considering the above-mentioned concerns. DISTRAL was designed as a framework with the objective of simultaneous reinforcement learning of multiple tasks [22]. The major design focus was on building a general framework for distilling the centroid policy and then transferring common behaviors of individual workers in multi-task reinforcement learning. Instead of the parameter sharing among the various workers within the environment, the key idea behind the design of DISTRAL was to share a distilled policy that can capture common behavior across tasks.

Figure 4 (adopted from [22]) shows the architecture of the DISTRAL model.

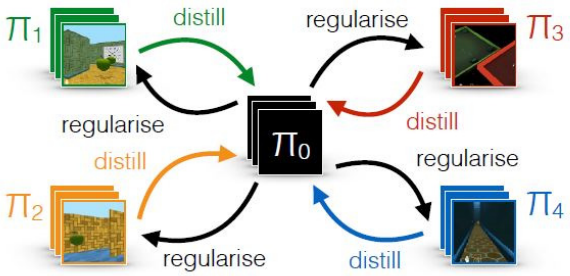

Figure 4. Architecture of the DISTRAL framework.

The design of DISTRAL is based on the notion of a shared policy that distills common behaviors or representations from task-specific policies of individual tasks trained within the environment [42]. After deducing the distilled policy, further on it will be used to guide task-specific policies through regularization by using a Kullback–Leibler (KL) divergence [43]. In this way, firstly knowledge gained in one task is distilled into the shared policy, and then finally transferred to other tasks [40]. With this approach, each worker will be individually trained to solve its own task by staying as close as possible to the shared policy. This shared policy will be trained by using the distillation, which acts as the centroid of all task policies. This method was proven to be quite efficient for the transfer of knowledge in RL problems that involve complex 3D environments. More importantly, this learning process was found to be more robust and stable when applied under deep reinforcement learning.

The DISTRAL approach has proven to outperform the traditional way of using shared neural network parameters for multi-task or transfer reinforcement learning by a large margin [22]. This was predominantly due to the two major factors listed below. Firstly, distillation plays a vital role in the optimization procedure when using KL divergences as a means to regularize the output of task models towards a distilled model deduced from the individual task policies. Secondly, the distilled model

itself is used as a regularizer for training the individual task models within the environment [22]. More importantly, using the distilled model as a regularizer brings the aspect of regularizing the networks of individual workers into a more meaningful level such as at task policies, rather than at the parameters' level [53].

5.2. IMPALA (Importance Weighted Actor-Learner Architecture)

One of the major challenges in achieving multi-task functionality with a single reinforcement learning agent is to handle the increased amount of data that an agent needs to handle, and also the amount of training time required. Based on the research conducted to address the above aspects of multi-tasking within the reinforcement learning domain, DeepMind proposed an architecture named IMPALA (Importance Weighted Actor-Learner Architecture). At the very basic level, IMPALA is a distributed agent architecture developed by adopting a single reinforcement learning agent having a single set of parameters. One of the key aspects of the IMPALA approach is its ability to effectively use the resources in a single-machine training environment, while it can scale to multiple machines without sacrificing data efficiency or resource utilization. By leveraging on a novel off-policy correction method named V-trace, IMPALA can achieve quite stable learning at high throughput by combining decoupled acting and learning [23].

Typically, the architecture of a deep reinforcement learning model is based on a single learner (critic) that is combined with multiple actors. In this ecosystem, every individual actor generates its own learning cycle parameters (also known as trajectories), and then sends that knowledge to the learner (critic) through a queue. The learner collects the same kind of trajectories from all the other actors in the environment and prepares a central policy. Before the next learning cycle (trajectory), every actor retrieves the updated policy parameters from the learner (critic). This approach is quite close to the reinforcement learning algorithm named A3C, and the architecture of IMPALA has been heavily inspired by this algorithm. IMPALA follows a topology of multiple actors and learners can collaborate to build knowledge. Figures 5 and 6 adopted from [23] show the architecture of an IMPALA ecosystem with a single learner and multiple learner configurations, respectively.

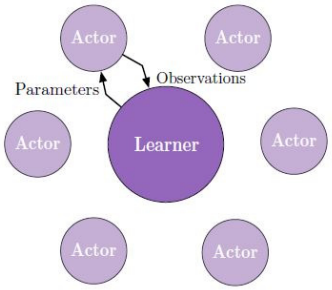

Figure 5. IMPALA single learner model.

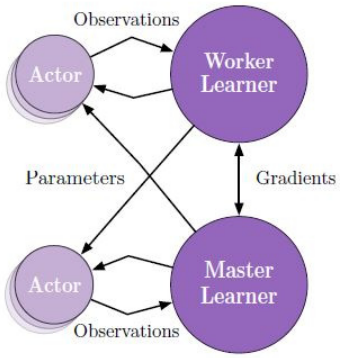

Figure 6. IMPALA multiple synchronous leaners model.

IMPALA follows an actor-critic setup to learn a policy π and a baseline function named Vπ. Major components of the IMPALA system consist of a set of actors who are continuously generating trajectories of experience, and additionally there could be one or more learners that use these generated experiences sent from actors to learn π, which is an off-policy. At the start of each trajectory, an actor will update its own local policy μ to the latest learner policy π [23]. Further on the actor will run that policy for n steps in its environment. At the end of these n steps, the actor sends another trajectory of states, actions, and rewards together with related policy distributions to the learner. In this manner, the learner will be continuously updating its policy π each time trajectory information is collected from the actors within the environment [23]. In this manner, the IMPALA architecture collects experiences from different learners, which are passed to a central learner. Further on this central learner calculates the gradients and generates a model with independent actors and learners. One of the key aspects of this IMPALA architecture is that actors need not be present on the same machine, but can be distributed across many machines.

5.3. PopArt

In recent advancements, reinforcement learning has proven to be capable of surpassing human-level performance on specific tasks. One specific aspect of all these RL agents was that they were all trained with one task at a time, and each additional task that needs to be trained needs the instantiation of the agent. In order to overcome this limitation, there were many pieces of research conducted to enhance RL algorithms with the capability of multiple sequential decision tasks at once. These research attempts to bring in the support for multi-task learning have often confronted various challenges. One such major challenge was related to establishing a balance between the needs of multiple tasks within the environment competing for the limited resources of a single learning system. This is related to the probability of learning algorithms often getting distracted (often called the distraction dilemma) by a few tasks from the set of multiple tasks to solve [10]. Typically, this situation demands the need for a multitask reinforcement learning (MTRL) system to build the immunity against the distraction dilemma and establish a balance concerning mastering individual tasks against the ultimate goal of achieving better generalization. The prime reason behind this distraction scenario occurs because some of the tasks appear more salient to the learning process due to the associated density or magnitude of the rewards for such tasks (in-task rewards). At different timesteps during the lifecycle of an RL agent within an MTRL system, agents are going to be met with tasks that appear to be more salient to the learning process. This prompts the algorithm to give more attention to such tasks, and then focus on those salient tasks at the expense of generality by giving less importance to other tasks in the multi-tasking environment. This indeed causes the multi-task reinforcement learning (MTRL) agents to often focus on the wrong tasks by ignoring or giving less importance to a few of the other tasks [10].

As a solution to this problem, DeepMind proposed a new method named PopArt to improve reinforcement learning in multi-task environments. The core objective of PopArt is to minimize distraction and thereby stabilize learning to facilitate the adoption of MTRL (multi-task reinforcement learning) techniques. The PopArt model was designed based on the original IMPALA (importance weighted actor-learner architecture) architecture model with the combination of multiple convolutional neural network layers with other techniques, such as word embeddings with the recurrent neural network of long-short term memory (LSTM) type [10]. The PopArt methodology works by adapting the contributions from each of the individual tasks to the agent's updates. This way PopArt ensures that all the agents will have their own individual role and thereby a proportional impact on the overall learning dynamics. The key aspect of the PopArt relies on modifying the weights of the neural network based on the output of all tasks within the environment. In the initial stage, PopArt estimates the mean as well as the spread of the ultimate targets, such as the score of a game across all tasks under consideration. Further on, PopArt uses these estimate values to normalize the targets before updating the network's weights. This approach makes the learning process more stable and

robust. In experiments conducted with Atari games, PopArt demonstrated improvements over other multi-task reinforcement learning architectures. The issue concerning the scaling of individual rewards been addressed through reward clipping [31]. Reward clipping is a method used to normalize the rewards across all settings of the environment when there is a discrepancy among the reward systems used between the various environments.

5.4. Comparison of Existing Solutions

The objective of this section is to provide a comparative study of the three existing approaches that are explained in the previous three subsections. DISTRAL was a new approach derived towards the joint training of multiple tasks. Rather than sharing the parameters between the different workers within the system, the core ideology used behind this approach was to share a distilled policy that captures common behavior across tasks. The major design focus remained on building a general framework for distilling the centroid policy and then transferring common behaviors of individual workers in multi-task reinforcement learning. Within the DISTRAL model-based ecosystem, each of the individual workers is trained to solve its own task while it was constrained to stay as close to the shared policy as possible. Simultaneously, the shared policy is trained by the distillation process to always remain as the centroid of all task policies within the ecosystem. The design of DISTRAL is based on the notion of a shared policy that distills common behaviors or representations from task-specific policies of individual tasks trained within the environment [42]. This approach was found to be quite successful and efficient in the transfer of knowledge within the reinforcement learning problems that function under 3D environments and achieved an impressive level of performance in comparison to other related methods [22].

IMPALA design was moreover based on distributed agent architecture developed by adopting a single reinforcement learning agent having a single set of parameters [23]. IMPALA possesses the ability to effectively use the resources in a single-machine training environment, while it can scale to multiple machines without sacrificing data efficiency or resource utilization. At the core level, the architecture of IMPALA was heavily inspired by the famous A3C algorithm, wherein IMPALA follows a topology of multiple actors in which learners can collaborate to build knowledge. The IMPALA system was designed with a set of actors who are continuously generating trajectories of experience, and additionally, there could be one or more learners that use these generated experiences sent from actors to learn π, which is an off-policy. At the start of each trajectory, an actor will update its own local policy μ to the latest learner policy π. Further on the actor will run that policy for n steps in its environment. At the end of these n steps, the actor sends another trajectory of states, actions, and rewards together with related policy distributions to the learner. In this manner, the learner will be continuously updating its policy π each time trajectory information is collected from the actors within the environment. In this manner, the IMPALA architecture collects the experiences from different learners, which are then passed to a central learner. Further on this central learner calculates the gradients and generates a model with independent actors and learners [23]. One of the major drawbacks of the IMPALA model is related to a scenario named distraction dilemma, which triggered the design of a new methodology named PopArt [10].

Whereas architectures such as IMPALA are vulnerable to the distraction dilemma scenario, the key motivating factor behind the design of the PopArt methodology was to come up with an effective method to overcome that bottleneck. The IMPALA model was greatly affected by the distraction dilemma—a major challenge related to establishing a balance between the needs of multiple tasks within the same environment competing for the limited resources of a single learning system. This is internally linked to the probability of learning algorithms getting distracted (often called the distraction dilemma) by a few tasks from the set of multiple tasks to solve. This situation demanded the need for a multitask reinforcement learning (MTRL) system to build immunity against the distraction dilemma and establish a balance concerning mastering individual tasks against the ultimate goal of achieving better generalization. The design of PopArt model was inspired by the original IMPALA architecture

model itself. IMPALA model aintegrates the combination of multiple convolutional neural network layers with other techniques, such as word embeddings with the recurrent neural network of type long-short term memory [10]. The PopArt methodology works by adapting the contributions from each of the individual tasks to the agent's updates [10]. In this way PopArt ensures that all agents will have their role and a proportional impact on the overall learning dynamics. Table 1 provides a comparison summary between the three most popular existing solutions related to the multi-task DRL.

Table 1. Comparison of existing solutions.

Characteristic	Name of Solution		
	PopArt	**IMPALA**	**DISTRAL**
Distraction Dilemma	No	Yes	No
Master Policy	Yes	Yes	Yes
Operation Model	A3C Model	A3C Model	Centroid Policy
Multitask Learning	Yes	Yes	Yes

The key aspect of PopArt relies on modifying the weights of the neural network based on the output of all tasks within the environment. In the initial stage, PopArt estimates the mean as well as the spread of ultimate targets, such as the score of a game, across all tasks under consideration. Further on, PopArt uses these estimate values to normalize the targets before updating the network's weights [10]. This approach makes the learning process more stable and robust in comparison to IMPALA. As evident from the Table 1, two of the solutions are based on the A3C model, whereas the third one, DISTRAL, is oriented on a centroid policy model.

6. Conclusions

This literature review was conducted with the objective of surveying and analyzing various methodologies that are developed for the optimization of the reinforcement learning (RL) agent's multi-tasking learning capabilities with the help of deep reinforcement learning. Throughout this literature survey, the key focus of the analysis was kept on the multi-task related aspect of all the methodologies examined. An analysis of each of these methodologies was carried out by focusing on their applicability in the context of deep reinforcement learning. There is a growing literature that is specially focused on the multi-tasking aspect of deep reinforcement learning. Notably, the multi-tasking aspect of deep reinforcement learning (DRL) is considered to be one of the major research challenges in artificial intelligence these days, and in deep reinforcement learning research in particular. The majority of the literature contents covered in this survey focused on knowledge building by training a reinforcement learning (RL) agent on multiple tasks simultaneously. Further on, by leveraging this knowledge base, they examined how to transfer it across other tasks. A huge part of the research efforts in the area of multi-task deep reinforcement learning (MTDRL) is driven by the research efforts conducted by well-known research organizations such as DeepMind and OpenAI. A general conclusion from all these studies is that the inception of deep learning and neural networks, followed by its application into the reinforcement domain has helped to optimize the multi-tasking learning process to a great extent in comparison to how it used to be within the traditional RL environment. Having said that, multi-tasking and transfer learning remain key challenges within the deep reinforcement domain, which need to be investigated further.

As part of surveying the related work done concerning multi-tasking, various methodologies, such as transfer learning oriented approach, shared representations for value functions, progressive neural networks, PathNet, policy distillation, and actor-mimic and asynchronous advantage actor-critic (A3C) were evaluated from the context of deep reinforcement learning. Along with the analysis of each of the aforementioned methodologies, attempts were made to survey how some of these techniques were used to address the key challenges within the reinforcement domain. In addition to this, the survey also included some of the existing solutions developed by DeepMind named DISTRAL,

PopArt, and IMPALA. One major conclusion derived from this literature survey is about the role of the multi-task learning aspect in the context of optimizing the performance of the deep reinforcement learning (DRL) agent. In particular, multi-task learning, wherein policies learned for each task will be executed in parallel, has the potential to optimize the performance of the RL agent [54]. This is predominantly due to the applicability of the multi-tasking aspect in creating additional room for deep exploration [55]. In addition to this, multi-tasking could also bring additional benefits towards achieving optimal policy composition. It is quite evident from the multi-task-based architecture of all three models referenced that most of the deep reinforcement multi-task challenges, such as scalability, partial observability, and effective exploration are being addressed to a great extent. At the same time, negative knowledge transfer can be caused by the gradient values shared by the individual learners within a multi-task environment; this remains an active research challenge within the MTDRL. When it comes to other challenges, such as catastrophic forgetting and the distraction dilemma, models such as IMPALA are having related issues, but then while designing PopArt, these problems have been taken care of by bringing modules like an RNN model named LSTM into the architecture.

Author Contributions: Writing—original draft preparation, N.V.V.; supervision, and writing—review and editing, Q.H.M. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sutton, R.S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*; MIT Press: Cambridge, MA, USA, 1996.
2. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
3. Konidaris, G.; Osentoski, S.; Thomas, P. Value function approximation in reinforcement learning using the Fourier basis. In Proceedings of the Twenty-fifth AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011.
4. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Twenty-sixth Annual Conference on Neural Information Processing Systems 2012, Lake Tahoe, Nevada, 3–8 December 2012; pp. 1097–1105.
5. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
6. Holcomb, S.D.; Porter, W.K.; Ault, S.V.; Mao, G.; Wang, J. Overview on deepmind and its alphago zero ai. In Proceedings of the 2018 International Conference on Big Data and Education, Honolulu, HI, USA, 9–11 March 2018; pp. 67–71.
7. Caruana, R. Multitask learning. *Mach. Learn.* **1997**, *28*, 41–75. [CrossRef]
8. Ding, Z.; Dong, H. *Challenges of Reinforcement Learning In Deep Reinforcement Learning*; Springer: Berlin/Heidelberg, Germany, 2020.
9. Glatt, R.; Costa, A.H.R. Improving deep reinforcement learning with knowledge transfer. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
10. Hessel, M.; Soyer, H.; Espeholt, L.; Czarnecki, W.; Schmitt, S.; van Hasselt, H. Multi-task deep reinforcement learning with popart. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
11. Li, Y. Deep reinforcement learning: An overview. *arXiv* **2017**, arXiv:1701.07274.
12. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. A brief survey of deep reinforcement learning. *arXiv* **2017**, arXiv:1708.05866. [CrossRef]
13. Sallab, A.E.; Abdou, M.; Perot, E.; Yogamani, S. Deep reinforcement learning framework for autonomous driving. *Electron. Imaging* **2017**, *2017*, 70–76. [CrossRef]
14. Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J.J.; Gupta, A.; Fei-Fei, L.; Farhadi, A. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3357–3364.

15. Calandriello, D.; Lazaric, A.; Restelli, M. Sparse multi-task reinforcement learning. In Proceedings of the 28th Annual Conference on Neural Information Processing Systems Neural Information, Montreal, QC, Canada, 8–13 December 2014; pp. 819–827.
16. Song, M.-P.; Gu, G.-C.; Zhang, G.-Y. Survey of multi-agent reinforcement learning in markov games. *Control Decis.* **2005**, *20*, 1081.
17. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Trans. Cybern* **2020**, *50*, 3826–3839. [CrossRef]
18. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 750–797. [CrossRef]
19. Denis, N.; Fraser, M. Options in Multi-task Reinforcement Learning-Transfer via Reflection. In Proceedings of the Canadian Conference on Artificial Intelligence, Kingston, ON, Canada, 28–31 May 2019; pp. 225–237.
20. da Silva, F.L.; Costa, A.H.R. A survey on transfer learning for multiagent reinforcement learning systems. *J. Artif. Intell. Res.* **2019**, *64*, 645–703. [CrossRef]
21. Palmer, G.; Tuyls, K.; Bloembergen, D.; Savani, R. Lenient multi-agent deep reinforcement learning. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 443–451.
22. Teh, Y.; Bapst, V.; Czarnecki, W.M.; Quan, J.; Kirkpatrick, J.; Hadsell, R.; Heess, N.; Pascanu, R. Distral: Robust multitask reinforcement learning. In Proceedings of the Thirty-first Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9, December 2017; pp. 4496–4506.
23. Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv* **2018**, arXiv:1802.01561.
24. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
25. Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R. Planning and acting in partially observable stochastic domains. *Artif. Intell.* **1998**, *101*, 99–134. [CrossRef]
26. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*; MIT Press Cambridge: Cambridge, MA, USA, 1998; Volume 2.
27. Waibel, A. Modular construction of time-delay neural networks for speech recognition. *Neural Comput.* **1989**, *1*, 39–46. [CrossRef]
28. Pontil, M.; Maurer, A.; Romera-Paredes, B. The benefit of multitask representation learning. *J. Mach. Learn. Res.* **2016**, *17*, 2853–2884.
29. Boutsioukis, G.; Partalas, I.; Vlahavas, I. Transfer learning in multi-agent reinforcement learning domains. In *European Workshop on Reinforcement Learning*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 249–260.
30. Weiss, G. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1999.
31. Mnih, V.; Kavukcuogluy, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef]
32. Yoshua, B. Learning deep architectures for AI. *Found. Trends Mach. Learn.* **2009**, *2*, 1–127.
33. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484. [CrossRef]
34. Borsa, D.; Graepel, T.; Shawe-Taylor, J. Learning shared representations in multi-task reinforcement learning. *arXiv* **2016**, arXiv:1603.02041.
35. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
36. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2011.
37. Rusu, A.A.; Rabinowitz, N.C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; Hadsell, R. Progressive neural networks. *arXiv* **2016**, arXiv:1606.04671.
38. Taylor, M.E.; Stone, P. An introduction to intertask transfer for reinforcement learning. *Ai Mag.* **2011**, *32*, 15. [CrossRef]

39. Fernando, C.; Banarse, D.; Blundell, C.; Zwols, Y.; Ha, D.; Rusu, A.A.; Pritzel, A.; Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv* **2017**, arXiv:1701.08734.
40. Rusu, A.A.; Colmenarejo, S.G.; Gulcehre, C.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; Hadsell, R. Policy distillation. *arXiv* **2015**, arXiv:1511.06295.
41. Buciluǎ, C.; Caruana, R.; Niculescu-Mizil, A. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PS, USA, 20–23 August 2006; pp. 535–541.
42. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
43. Parisotto, E.; Ba, J.L.; Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv* **2015**, arXiv:1511.06342.
44. Akhtar, M.S.; Chauhan, D.S.; Ekbal, A. A Deep Multi-task Contextual Attention Framework for Multi-modal Affect Analysis. *ACM Trans. Knowl. Discov. Data (TKDD)* **2020**, *14*, 1–27. [CrossRef]
45. Bellemare, M.G.; Naddaf, Y.; Veness, J.; Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.* **2013**, *47*, 53–279. [CrossRef]
46. Mnih, V.; Badia, A.P.; Mirza, M.; Grave, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
47. Wang, Y.; Stokes, J.; Marinescu, M. Actor Critic Deep Reinforcement Learning for Neural Malware Control. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 1005–1012.
48. Vuong, T.-L.; Nguyen, D.-V.; Nguyen, T.-L.; Bui, C.-M.; Kieu, H.-D.; Ta, V.-C.; Tran, Q.-L.; Le, T.-H. Sharing experience in multitask reinforcement learning. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 3642–3648.
49. Li, L.; Gong, B. End-to-end video captioning with multitask reinforcement learning. In Proceedings of the 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), Hilton Waikoloa Village, HI, USA, 7–11 January 2019; pp. 339–348.
50. Chaplot, D.S.; Lee, L.; Salakhutdinov, R.; Parikh, D.; Batra, D. Embodied Multimodal Multitask Learning. *arXiv* **2019**, arXiv:1902.01385.
51. Yang, M.; Tu, W.; Qu, Q.; Lei, K.; Chen, X.; Zhu, J.; Shen, Y. MARES: Multitask learning algorithm for Web-scale real-time event summarization. *World Wide Web* **2019**, *22*, 499–515. [CrossRef]
52. Liang, Y.; Li, B. Parallel Knowledge Transfer in Multi-Agent Reinforcement Learning. *arXiv* **2020**, arXiv:2003.13085.
53. Liu, X.; Li, L.; Hsieh, P.-C.; Xie, M.; Ge, Y. Developing Multi-Task Recommendations with Long-Term Rewards via Policy Distilled Reinforcement Learning. *arXiv* **2020**, arXiv:2001.09595.
54. Osband, I.; Blundell, C.; Pritzel, A.; van Roy, B. Deep exploration via bootstrapped DQN. In Proceedings of the Thirtieth Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4026–4034.
55. Mankowitz, D.J.; Žídek, A.; Barreto, A.; Horgan, D.; Hessel, M.; Quan, J.; Oh, J.; van Hasselt, H.; Silver, D.; Schaul, T. Unicorn: Continual learning with a universal, off-policy agent. *arXiv* **2018**, arXiv:1802.0829.