CS 747: Foundations of Intelligent and Learning Agents **Jul-Nov 2018**

Project Report : Progressive Neural Networks for Multitask Learning

Students : Sneha Bhakare (150050040), Videsh Suman (150040095)

Abstract

Deep neural networks provide rich representations that can enable reinforcement learning algorithms to perform effectively. Learning to solve complex sequences of tasks—while both leveraging transfer and avoiding catastrophic forgetting—remains a key obstacle to achieving human-level intelligence. Through this project, we have explored the area of multitask learning using the novel approach of progressive neural networks. This form of network architecture represents a step forward in the above stated direction: they are immune to forgetting and can leverage prior knowledge via lateral connections to previously learned features. We have evaluated this architecture on the baselines of the asynchronous advantage actor critic algorithm for two Atari games, Pong and Breakout.

1 Introduction

The foundation of deep reinforcement learning was laid by Deep Q-Networks[4], which made it possible for a single agent (a neural network with same hyperparameters) to be trained independently to perform exceptionally well on a whole range of tasks. However, deriving inspiration from the human learning phenomenon, the agent should be able to leverage prior experiences to acquire new knowledge instead of starting from scratch when trained for a different task. This knowledge reusability for similar tasks has been taken care of by disciplines such as representation and transfer learning. Similarly, multitask learning can be helpful when training data is insufficient for a particular task, and the prior knowledge from similar tasks can be used for the current problem at hand. However, these techniques still have severe limitations when comes to learning similar tasks in the same model. In the usual vanilla transfer learning approach, a model is pretrained on a source domain (where data is often abundant), the output layers of the model are adapted to the target domain, and the network is fine-tuned via backpropagation. However, this approach has a major drawback of catastrophic forgetting, while leveraging knowledge transfer.
In 2016, Andrei et al. [6] suggested progressive neural networks (PNN) approach which are supposed to be immune to forgetting and can leverage prior knowledge via lateral connections to previously learned features. In their work they have relied on the asynchronous variant of the advantage actor-critic (A3C) framework introduced by Mnih et al.[3] as their baseline. The A3C algorithm has been posited to have surpassed the previous GPU-based state-of-the-art results on a variety of RL tasks based experiments merely run on a standard multi-core CPU machine in far less time. As a part of this research, we claim to have implemented Progressive Nets as a wrapper around the A3C framework to explore the possibility of multitask learning on two related RL tasks of the Atari games Pong and Breakout. We have

2 Literature

Transfer and multi-task reinforcement learning are recognized as critical challenges in AI research. Many methods for transfer learning rely on linear and other simple models [7], which is a limiting factor to their applicability. With the advent of deep RL, there have been new methods proposed

for multi-task or transfer learning. The actor-mimic approach [5] applied these principles to reinforcement learning, by fine-tuning a DQN multi-task network on new Atari games and showing that some responded with faster learning, while others did not.
Pretraining and finetuning was proposed in [2] and applied to transfer learning in [1], generally in unsupervised-to-supervised or supervised-to-supervised settings. Here, the model is pretrained on a source domain usually where data is abundant, the output layers of the model are adapted to the target domain, and the network is finetuned via backpropagation. This approach was pioneered in [2] by transferring knowledge from a generative to a discriminative model, and has since been generalized with great success [1]. Furthermore, while finetuning may allow us to recover expert performance in the target domain, it is a destructive process which discards the previously learned function.
Asynchronous advantage actor-critic (A3C) [3] is an asynchronous variant of actor-critic. It maintains a policy $\pi(\mathrm{a}_t|s_t;\Theta)$ and an estimate of the value function V $(\mathrm{s}_t;\Theta_v)$. It operates in the forward view and uses mix of n-step returns to update both the policy and the value-function. The policy and the value function are updated after every tmax actions or when a terminal state is reached.
Progressive nets enables agent not only learn to (and remember) a series of tasks experienced in sequence, but also have the ability to transfer knowledge from previous tasks to improve convergence speed. This is achieved by modifying the model architecture: catastrophic forgetting is prevented by instantiating a new neural network (a column) for each task being solved, while transfer is enabled via lateral connections to features of previously learned columns.
A progressive network starts with a single column: a deep neural network having L layers with hidden activations $h_i^{(1)} \in \mathbb{R}^{n_i}$, with n_i the number of units at layer iL, and parameters $\Theta^{(1)}$ trained to convergence. When switching to a second task, the parameters $\Theta^{(1)}$ are frozen and a new column with parameters $\Theta^{(2)}$ is instantiated (with random initialization), where layer $h_i^{(2)}$ receives input from both $h_{i1}^{(2)}$ and $h_{i1}^{(1)}$ via lateral connections. This generalizes to K tasks as follows:

$$h_i^{(k)} = f(W_i^{(k)} h_{i1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i1}^{(j)}),$$

where $W_i^{(k)} \in \mathbb{R}^{n_i \times n_{i-1}}$ is the weight matrix of layer i of column k, $U_i^{(k:j)} \in \mathbb{R}^{n_i \times n_j}$ are the lateral connections from layer $i-1$ of column j, to layer i of column k and h_0 is the network input. f is an element-wise non-linearity: we use $f(x) = max(0, x)$ for all intermediate layers. A progressive network with K = 3 is shown in Figure 1.

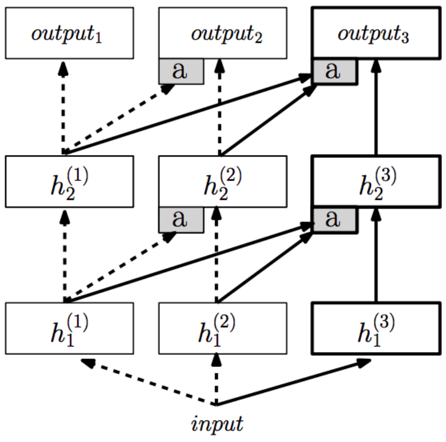

Figure 1: (Source - Andrei et al.[6]) Depiction of a three column progressive network. The first two columns on the left (dashed arrows) were trained on task 1 and 2 respectively. The grey box labelled a represent the adapter layers (see text). A third column is added for the final task having access to all previously learned features.

3 Method

3.1 Asynchronised Advantage Actor-Critic (A3C)

We have implemented Asynchronised Advantage Actor-Critic (A3C) on 2 OpenAI Gym test environments i.e. Pong and Breakout. The frames in the game are the inputs to the A3C network. The input image is preprocessed by converting from RGB to grayscale and downscaling to size 80 x 80. Total 20 processes have been run asynchronously. The model of individual processes is synced with the shared model every 20 time steps. The network architecture used for A3C is mentioned in Table 1. Here, Linear1 layer represents the critic i.e it is the value estimate and Linear2 layer represents the actor i.e. it is the policy. The implementation is done in Pytorch.

Layer	Input Channels	Output Channels	Kernel Size	Stride	Padding	Activation
Conv1	1	32	3 x 3	2	1	elu
Conv2	32	32	3 x 3	2	1	elu
Conv3	32	32	3 x 3	2	1	elu
Conv4	32	32	3 x 3	2	1	elu
GRUCell	800	256				
Linear1	256	1				
Linear2	800	no. of actions				

Table 1: Network Architecture for A3C

The hyperparameters used are discount factor = 0.99, learning rate = 0.0001. The A3C network is trained on Pong and Breakout for 20 million timesteps. The training time required for this was 23 hours.

3.2 Progressive Neural Network

The Progessive Neural Network has two columns corresponding to the two tasks. Each column has network architecture same as A3C. The modification in architecture is that lateral connections are added. The first column is initialised with a pretrained model of task 1 and the weights for this column are frozen. The second column corresponds to task 2.

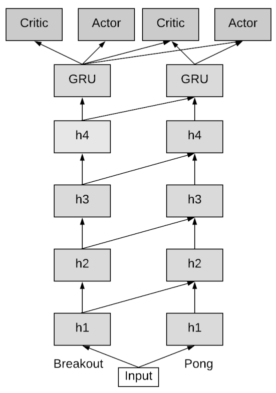

Figure 2: (a) PNN Layout

4 Results

The A3C network is trained on Pong and Breakout for 20 million timesteps as shown in Figure 3.

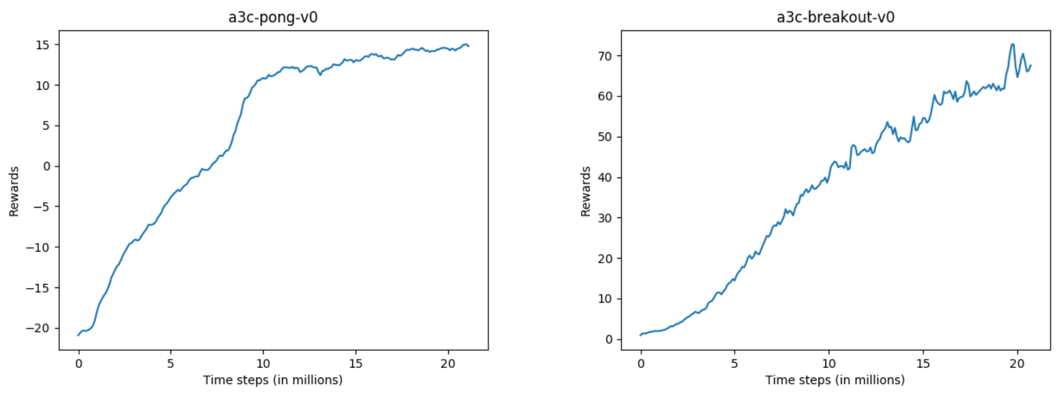

Figure 3: (a) Pong A3C, (b) Breakout A3C

The first column of Progressive Neural Networks is initialized with weights of Breakout at 8 million timesteps, then it trained on Pong for further 6 million timesteps as shown in Figure 4

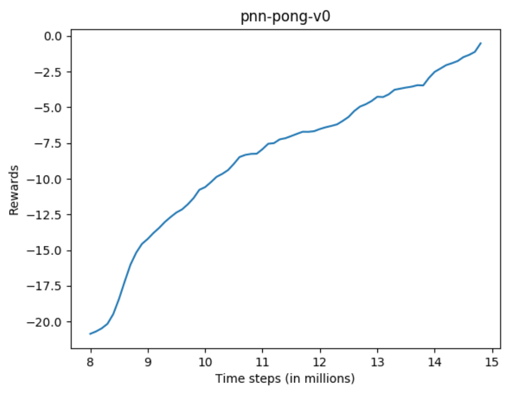

Figure 4: (Pong PNN pretrained on Breakout

The A3C network works well for Pong and Breakout. The PNN network requires more training to perform better than A3C of Pong as well as Breakout.

5 Future Work

In the future, we are interested to study in detail, the transfer analysis of the features being passed on from the source columns to the target task. Unlike finetuning, progressive nets do not destroy the features learned on prior tasks, thus a relevant sensitivity analysis can be helpful in evaluating the impact of the prior frozen tasks (positive or negative) on the target problem. For this, a naive approach is to evaluate the average perturbation sensitivity[6] by inject Gaussian noise at isolated

points in the architecture (e.g. a given layer of a single column) and measure the impact of this perturbation on performance. A significant drop in performance indicates that the final prediction is heavily reliant on the feature map or layer.
Besides, more recent and robust baselines like Actor Critic using Kronecker-Factored Trust Region (ACKTR)[9] and Proximal Policy Optimization[8] algorithms need to be probed as plausible frameworks for the progressive nets method.

References

[1] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. 2012. `http://proceedings.mlr.press/v27/bengio12a/bengio12a.pdf`.

[2] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. 2006. `http://science.sciencemag.org/content/313/5786/504`.

[3] Volodymyr Mnih, Adri Puigdomnech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *arXiv*, 2016. `https://arxiv.org/abs/1602.01783`.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv*, 2013. `https://arxiv.org/abs/1312.5602`.

[5] Mark B. Ring. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv*, 2016. `https://arxiv.org/abs/1511.06342`.

[6] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv*, 2016. `https://arxiv.org/abs/1606.04671`.

[7] Paul Ruvolo and Eric Eaton. An efficient lifelong learning algorithm. 2013. `http://proceedings.mlr.press/v28/ruvolo13.pdf`.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017. `https://arxiv.org/abs/1707.06347`.

[9] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *arXiv*, 2017. `https://arxiv.org/abs/1708.05144`.