LEARNING WITH AUXILIARY TASKS: A STUDY IN REPLICATION AND PREDICTION

Cody Rosevear
Department of Computing Science
University Of Alberta
Edmonton, AB T6G 2E8, Canada
rosevear@ualberta.ca

ABSTRACT

Recent work in deep reinforcement learning has demonstrated that learning with auxiliary tasks in complex visual environments with highly parallelized agent architectures can yield significant performance improvements with respect to a number of metrics, relative to agents whose training focuses exclusively on a single task (Jaderberg et al., 2017). However, the efficacy of auxiliary tasks within the context of simpler, non-visual environments with more traditional serial agent architectures remains understudied. The following work explores the conditions necessary for efficacious learning with auxiliary tasks by studying the effects of a number of such tasks in a variety of simple gridworld maze environments with differing state transition dynamics and reward schemes. Results on these preliminary experiments demonstrate no significant improvement over single task learning in terms of the speed of learning, and no systematic relationship between specific auxiliary tasks and particular environment characteristics.

1 INTRODUCTION & RELATED WORK

Recent work in reinforcement learning (RL) has suggested that training one's agent alongside an additional number of suitable auxiliary tasks can lead to an improvement in performance across a variety of metrics, relative to agents whose training focuses exclusively on a single task (Jaderberg et al., 2017). In particular, these results demonstrated significant improvements over the state-of-the-art Asynchronous Advantage Actor-Critic (A3C) agent architecture (Mnih et al., 2016) on a testbed of environments from the arcade learning environment (Bellemare et al., 2013).

Auxiliary task based learning (ATBL) is similar to both multi-task learning within the context of traditional supervised learning, (Caruana, 1997) and transfer learning within RL (Taylor & Stone, 2009). In all three cases one is attempting to leverage agent skill gained from training on one task to improve performance along some dimension of another task. However, ATBL differs with respect to the traditional multi-task learning setting in that it embeds a neural network (or other function approximator) within an online setting without any explicitly labeled data, relying instead on a reward signal to shape agent performance, in the vein of the traditional RL paradigm. Moreover, ATBL is focused on leveraging improvements from other tasks occurring simultaneously within the context of the same markov decision process (MDP), whereas traditional transfer learning tends to focus on utilizing representations acquired from prior task learning sessions in a sequential way, so that agent performance can be improved across different MDP's (or variants of the same MDP), although there has been some work on leveraging multiple source tasks simultaneously to facilitate transfer learning as well (Parisotto et al., 2016).

While there are few, if any, theoretical analysis of ATBL per se, there is some prior work that analyzes multi-task learning more generally. Baxter (2000) characterizes multi-task learning as a form of meta-learning, wherein one is actively seeking to bias the hypothesis space of one's network in a given environment (problem domain) by exposing it to multiple tasks within that domain so that it can more efficiently generalize to novel tasks within that same environment, where 'efficiently' in this case refers to the sample complexity of one's tasks. Ben-David & Schuller (2003) attempt to better characterize when two or more tasks can be said to be part of the same environment,

and provide a definition based on a data generation model of task similarity, and proceed to prove generalization bounds in light of this more limited definition.

More recent empirical work has also been done within the field of medical diagnostics and computational lingusitics. With respect to the latter, Alonso & Plank (2017) provide empirical evidence of the favourability of certain data sets over others for use in supervised multi-task learning. They use a variety of information-theoretic measures to search for correlations in improvements in performance with certain data sets over others. Bingel & Søgaard (2017) provide further empirical support for the data distribution hypothesis of Alonso & Plank (2017), and furthermore develop a methodology of using the induced representations of single task learning to predict the effectiveness of auxiliary tasks. Finally, Situ et al. (2011) demonstrate the advantage of auxiliary tasks in a non-linguistic binary classification problem domain by treating the main task classifier as the weighted average of the output of a number of auxiliary classifiers.

Much of the above work focuses on examining the issue of which tasks in a multi-task learning setting are likely to be helpful with respect to the main task in virtue of some property or similarity of the auxiliary tasks in question. However, less attention has been devoted to exploring why, precisely, auxiliary tasks should be helpful at all in the reinforcement learning setting, especially given the latter's relative lack of propensity for overfitting due to its emphasis on continual interaction with the environment (and thus the collection of new data from which to generalize). In light of this, the following work proposes two contributions to the study of learning with auxiliary tasks.

Firstly, it seeks to more thoroughly examine the effect of auxiliary tasks within the context of a simpler environment and agent architecture compared to that which has been done so far (Jaderberg et al., 2017). In particular, it seeks to more clearly delineate the scope and impact of a set of auxiliary tasks on agent learning wthout the confounding variability of complicated agent architectures such as A3C that already feature large degrees of parallelism and other factors that may be boosting the effect of auxiliary tasks in ways that simpler environments and/or agent architectures are incapable of doing. Many problem domains in computer science admit of clever algorithms that allow problem instances to be solved efficiently at scale, but often these solutions are only useful for suitably large and/or complex problem instances, whether due to overhead of the proposed algorithms or other factors. Similarly, it may be the case that auxiliary tasks are only useful when deployed within the context of suitably complicated environments/agents, and it is therefore worth exploring whether simpler environments also admit of significant performance improvements via the addition of auxiliary tasks.

Secondly, it seeks to explore the impact of specific auxiliary tasks given the relationship between the current task and characteristics of the environment in question, such as its reward structure and state transition dynamics, so as to ascertain when certain types of tasks are more likely to be helpful with respect to certain environments.

2 BACKGROUND

A traditional RL problem as described in Sutton & Barto (2017) is composed of two parts: an agent, and an environment. The environment defines a set of states over which the agent can be said to inhabit at a given time step t, and for each such state there is a set of actions the agent can perform. Each state action pair (s_t, a_t) yields a corresponding real valued scalar reward, r_t, while transitioning to a new state s_{t+1} according to a (usually unknown) state transition function, which can be either deterministic or stochastic. The agent takes a new action a_{t+1}, yielding a new reward r_{t+1}, and the cycle continues, either until termination in the case of an episodic task with terminal states, or indefinitely, in the case of a continuing task.

The agent chooses actions according to a policy $\pi(a \mid s)$, which defines the probability of selecting action a given state s. The return, denoted G_t, is defined as the sum of the discounted rewards since time t. That is, $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_t$, where $0 \leq \gamma \leq 1$ is the discount factor and T is the final time step. T may be a natural number or, in the case of continuing tasks, be assigned the value of ∞, in which case γ must be < 1 to ensure that no returns diverge. The discount factor is used to specify how far-sighted the agent is, where 0 indicates a myopic agent concerned with maximizing the immediate reward and 1 specifies an agent that takes into account the full value of future states when deciding its actions.

The state-action value function $Q(s, a)$ is defined with respect to a particular policy π, such that it is equal to the expected return if one were to take action a in state s and follow policy π therafter. The goal of the agent is, roughly, to find a policy that maximizes the agent's expected return within the current environment.

3 AUXILIARY TASKS

The auxiliary task setting extends the traditional RL problem domain by adding one or more auxiliary tasks that are learned alongside the main task. These can be either additional control or prediction tasks. The focus of the present work is on auxiliary prediction tasks; specifically, on the tasks of reward and state prediction.

Reward prediction is a task of the following form: given a historical sequence of states and actions $(s_t, a_t, s_{t+1}, a_{t+1}, ...s_{t+n-1}, a_{t+n-1})$, starting at time t, predict the subsequent reward r_{t+n-1}, for a fixed value of n, where n is a hyperparameter that determines the size of the historical sequence used in predicting the subsequent reward.

Likewise, the following form comprises a state prediction task: given a historical sequence of states and actions $(s_t, a_t, s_{t+1}, a_{t+1}, ...s_{t+n-1}, a_{t+n-1})$, starting at time t, predict the subsequent state s_{t+n}.

In addition to the auxiliary tasks defined above, we also define two additional tasks intended to explore the impact of redundant and noisy auxiliary tasks on agent performance, so as to rule out the addition of noise and/or redundancy as explanations for any observed performance improvments.

With respect to the former, a redundant task is simply an auxiliary task that has the same set of inputs and target outputs as the main task (See section 4 for a definition of the main task).

With respect to the latter, we define a noise prediction task as having the following form: given a historical sequence of states and actions $(s_t, a_t, s_{t+1}, a_{t+1}, ...s_{t+n-1}, a_{t+n-1})$, starting at time t, predict the result of a pseudo-random number generator queried at the current time step.

Finally, it should be noted that all of these tasks make use of a type of experience replay. The sequences are stored in a buffer and then sampled in a skewed manner, rather than uniformly at random, so as to more evenly expose to the agent certain sequences during training than would otherwise be the case if it had to rely entirely on its current experiences. In this way, the experiential trajectories of auxiliary tasks can be biased to focus on various facets of the enviornment, such as rewards or dynamics, in a more equitable way, so as to learn about that facet of the enviornment more quickly.

4 HYPOTHESES

Beyond the general question of whether auxiliary tasks will help improve learning speed in simple environments, one can make more specific predictions regarding the relative performance of state and reward prediction tasks with respect to learning speed, given different environmental conditions such as the reward scheme and transition dynamics.

4.1 THESIS

The central hypothesis concerning the auxiliary-task-environment relationship is that reward prediction should be beneficial in sparse reward environments and neutral with respect to rich ones, while state prediction should be beneficial in stochastic environments and neutral with respect to deterministic ones, such that each auxiliary task agent should outperform the other when deployed in those environments congenial to it and the other is not.

The rationale concerning the reward prediction claim is that when using reward prediction as an auxiliary task in a sparse environment, the agent can sample from the replay buffer those historical sequences that yield zero and non-zero rewards more evenly, instead of having to wait to encounter rarely occuring non-zero reward values solely through its current interaction with the environment; in this way the features that recognize those states preceding the high value goal state will be shaped faster through reward prediction, yielding faster learning overall.

Similarly, the intuition with respect to state prediction is that by sampling from the replay buffer evenly between historical contexts with and without stochastic state transitions, and trying to predict the subsequent state given that historical context, the agent will come to recognize certain states as being of lower value, due to the uncertainty of transitioning from those states irrespective of its actions, more quickly than if it relied solely on collecting many samples from such stochastic states over time via direct experience.

4.2 PREDICTIONS

4.2.1 SPARSE DETERMINISTIC ENVIRONMENTS

In light of the above rationale it is reasonable to suppose that for environments with completely deterministic transition dynamics and sparse rewards, that reward prediction will be better at facilitating faster learning than state prediction.

4.2.2 RICH DETERMINISTIC ENVIRONMENTS

For the case of rich deterministic settings, it is plausible that neither state nor reward prediction will have much of a salutary effect on learning speed, and therefore that the relative performance difference between agents with state and reward prediction tasks will not be significant. This is because in such cases there is not much in the way of information in the environment that is missing or variable, such that focusing an auxiliary task on that aspect of the environment is likely to have much of an impact beyond what the main task is already acheiving.

4.2.3 RICH STOCHASTIC ENVIRONMENTS

In the case of rich stochastic environments it seems likely that the state prediction task will outperform reward prediction, since the added value of more quickly learning to recognize those states that are of low value in virtue of poor transition dynamics should provide it a greater advantage than that afforded by reward prediction, since the reward signal is already strong in such an environment and learning to predict it is unlikely to yield non-redundant information.

4.2.4 SPARSE STOCHASTIC ENVIRONMENTS

Finally, environments with stochastic transitions and sparse rewards would seem to be good candidates for either reward or state predictions tasks to have a significant positive effect on performance, relative to single task learning, for the same reasons as described above; however, it is not intuitively obvious which auxiliary task should perform better relative to the other, as this is likely to depend on the exact details of the reward scheme and transition dynamics of the particular environment in question.

5 EXPERIMENTS

5.1 GRIDWORLD MAZE

5.1.1 ENVIRONMENT

The gridworld maze environment as described in Sutton & Barto (2017) is a 6×9 grid of discrete states, with a single start and goal state. Intermediate states can be of two types: blocks, or open spaces. Blocks are spaces that the agent cannot inhabit and are denoted by grey squares, while open spaces, which the agent can inhabit, are white (See figure 1). The gridworld maze is an episodic task, where the agent's goal is to navigate from the start state to the goal state during each episode. A reward of 1 is provided everytime the agent reaches the goal state, and 0 otherwise. The action space is discrete, and composed of 4 possible movements: up, down, left, or right.

The experimental results discussed in section 6 include both the original formulation of gridworld, as described above, in addition to a number of variations. One variant simply involves changing the reward scheme from rich to sparse, such that all states return a reward of -1, except for the goal state, which returns a reward of 0.

Two other variations were also explored in which the obstacle states were altered to be inhabitable. Instead of explicitly blocking the agent from entering, the agent is allowed to enter the state, but whenever it takes an action from that state the results are no longer deterministic; rather, the action's results are decided according to a probability distribution, such that there is an 0.80 probability that the agent will go down or left, and a 0.20 probability that the agent will go up or right. The intuition is that these particlar dynamics are unfavourable for reaching the goal state in a way that knowledge of the state's dynamics would reveal, such that learning about them through state predicition tasks might facilitate more accurate approximations of the state's value earlier on, and thus faster learning overall. As in the deterministic case, results were obtained for both a rich reward and sparse reward version of the environment.

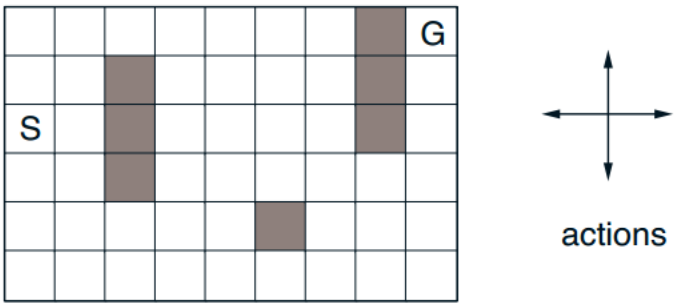

Figure 1: The girdworld maze environment. Source: (Sutton & Barto, 2017)

5.1.2 AGENTS

A simple agent that selected actions uniformly at random, with no learning, was used as a sanity check against which to compare all other agents. Additionally, two single task agents were implemented by which to compare the auxiliary task based agents. One of these was a tabular agent that learned off-policy via traditional Q-learning (Watkins, 1989). The other was also trained off policy, but used a two-layer feedforward neural network to approximate the state-action values in place of a table. All agents were trained using ϵ-greedy action selection, with the value of ϵ set to 1.0 at the start of each run, and then subsequently decreased after each successful episode until it reached a minimum value of 0.1.

The neural network weights for all agents, denoted by θ, were optimized via RMSProp. The main task loss used is the same as that described in (Mnih et al., 2015), and denoted by equation 1.

$$L_t = (r_t + \gamma \max_a Q(s_{t+1}, a, \theta_t) - Q(s_t, a_t, \theta_t))^2 \tag{1}$$

The reward prediction task loss R_t, defined by equation 2, is the binary cross entropy between the predicted reward p_x, and the subsequent actual reward r_{t+n-1}, given the historical sequence $x = (s_t, a_t, s_{t+1}, a_{t+1}, ...s_{t+n-1}, a_{t+n-1})$, starting at time t and ending at time $t + n$. For all experiments the set of zero and non-zero rewards for the given environment compose the 2 classes over which the loss is defined.

$$R_t = -(r_{t+n-1} * log(p_x) + (1.0 - r_{t+n-1}) * log(1.0 - p_x)) \tag{2}$$

The state prediction task loss S_t, defined by equation 3, is the multinomial cross entropy between the predicted state vector p_x, and the subsequent actual state vector s_{t+n}, given the historical sequence $x = (s_t, a_t, s_{t+1}, a_{t+1}, ...s_{t+n-1}, a_{t+n-1})$, starting at time t and ending at time $t+n-1$, and where m is the total number of states (classes), and i indexes the current state.

$$S_t = -\sum_{i=1}^{m} s_{t+n} i * log(p_x i) \tag{3}$$

Finally, the noise prediction task loss N_t, described by equation 4, seeks to minimize the squared error between the output of a pseudo-random number generator $rand_x$ and the auxiliary network's output o_x, given a historical sequence $x = (s_t, a_t, s_{t+1}, a_{t+1}, ...s_{t+n-1}, a_{t+n-1})$.

$$N_t = (rand_t - o_x)^2 \tag{4}$$

For each main-auxiliary task pair we define a single combined loss function, described by equation 5, that the agent seeks to optimize, and which is composed of the sum of the losses of the main task loss L_t and the current auxiliary task loss A_t, weighted by a set of hyperparameters $\lambda_i \in [0, 1]$.

$$C_t = \lambda_1 L_t + \lambda_2 A_t \tag{5}$$

5.2 Architecture & Implementation

The main task network is composed of an input layer with 54 units that takes a single 1-hot encoded state vector as input, which is derived from a 2-dimensional vector denoting the agent's current row and column position. There are 2 additional hidden layers, the first of which contains 164 units, and the second of which contains 150. The final output layer is structured with 4 nodes, so as to emit the value for each possible action given a single state vector as input, as in Mnih et al. (2013). This is done to speed up the maximization step in the Q-learning update so that one does not have to rerun the network for each state-action pair in order to find the maximum Q-value for the current state. Both hidden layers employ a rectilinear activation function, while the final layer is linear.

The auxiliary task network input layer has 216 units that takes as input the concatenation of 3 1-hot encoded state-action vectors, each of which are derived from 3 dimensional state-action vectors containing the agent's row and column position as well as the action from that state. This is followed by a single hidden layer of 128 units and an output layer that varies according to the current auxiliary task. The number of units in the final layer range from a single output node, in the case of reward prediction, to 56 units for the case of state prediction via classification. The redundant reward task uses 20 output nodes, 4 for each instantiaiton of the main task, while the pseudo-random prediction task uses 10 output nodes, the number of which was chosen arbitrarily. The hidden layer of the auxiliary network uses a rectilinear activation function, and is merged with the second hidden layer of the main task network to ensure a shared representation between the 2 networks. Sigmoid, softmax, and linear activation functions are used for the final layer of the auxiliary network for the reward, state, and redundant/noise prediction tasks, respectively. All networks use He normal initialization (He et al., 2015), and all auxilliary tasks use a replay buffer that stores the 10 most recent historical n-length contexts. See figure 2 for a depiction of the generic network architecture used for learning with auxiliary tasks.

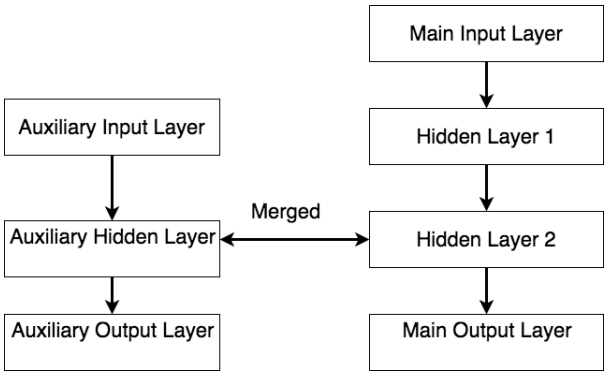

Figure 2: Generic Neural Network Architecture for Auxiliary Tasks. Source: The author

The experiment code implements the RL glue interface (Tanner & White, 2009). All neural networks were implemented using the Keras deep learning framework (Chollet et al., 2015) with Theano (Al-Rfou et al., 2016) serving as the backend. Source code for all of the experiments

can be found at: `https://github.com/ZerkTheMighty/CMPUT659-Project/tree/actions/Gridworld`

6 Results And Discussion

The average number of time steps until episode termination, as a function of the number of episodes, for all variations of the gridworld maze environment, are included below. Each experiment was run for 50 episodes with a maximum limit of 1000 time steps per episode. Results are averaged over 50 runs, each set with a random seed corresponding to that specific run, to induce identical trajectories of experience and random action selection across different agents, as well as to ensure the reproducibility of results. The learning rate and discount factor were set to 0.001 and 0.90, respectively, across all runs. The lambda hyperparamters responsible for weighting the component costs of the combined loss functions were set to 1 for all tasks and across all runs, and the historical sequence length used for the state, reward, and noise prediction tasks was set to $n = 3$ for all runs.

6.1 Rich Determinisitc Gridworld

None of the auxiliary task agents outperformed the single task agent in the rich deterministic setting; and, in particular, all but the state prediction task actually performed worse (see figure 3). This is not particularly surprising with respect to the redundant and noise prediction tasks, given that they are intended to act as essentially a form of interference, but even the reward prediction task hampered the agent's ability to learn to a non-trivial degree. The state prediction task, while showing no improvement, nevertheless suceeded in remaining competitive with the single task learning agent. This suggests that the state prediction task is having little effect on the main task, in contrast to the others.

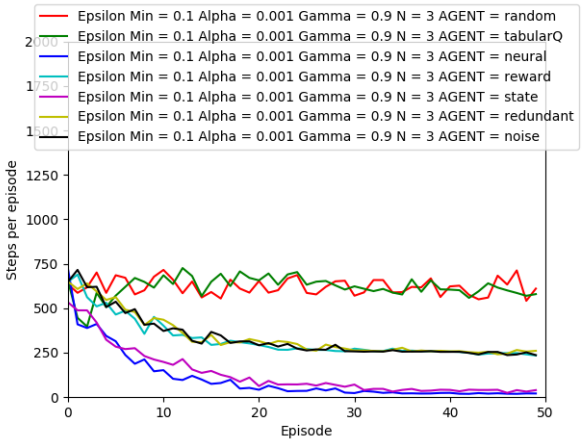

Figure 3: Rich Deterministic Gridworld Results

6.2 Sparse Deterministic Gridworld

None of the auxiliary task agents outperformed the single task agent within the sparse reward setting, although, prima facie, the variation across tasks in terms of their impact was greater in this case (See figure 4). The noise prediction task had the largest negative effect, followed by the redundant task, both of which reached a higher final episode termination time than the single task agent. This is generally in line with expectations insofar as both tasks were not intended to improve performance, and it is reasonable that the noise task would perform worse than simple redundancy, as the former is arguaby more likely to distort the network weights much more than simple redundancy due to the complete randomness of its target outputs.

Interestingly, even the reward task negatively impacted performance, albeit to a lesser degree than in the rich reward setting, as well as relative to the redundant and noise prediction tasks. This is some-

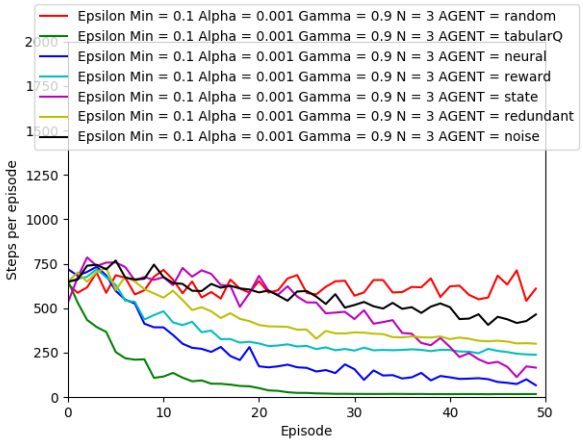

Figure 4: Sparse Deterministic Gridworld Results

what in line with the hypothesis that reward prediction should perform better in sparse environments rather than rich ones, but this is true only in the limited sense that it seems to have a smaller negative impact on performance in the sparse case, and should therefore not be taken as a true confirmation of the hypothesis.

Moreover, while reward prediction did outperform state prediction for roughly the first $4/5$ of the total number of episodes, the learning curve for state prediction declined precipitously during the last $1/5$ of episodes, and began to approach the final performance level of the single task agent. This behaviour is rather unexpected, but one possible explanation for it is that since state prediction seems to have less of an impact on the network's weights than the other tasks do, as attested by the similarity of performance between the state prediction and single task agents in the rich reward setting, it may be that once the agent learns enough so that it is actually improving with each episode, it reaches a 'tipping point', where the impact of the auxiliary task is overtaken by the continual improvements made by the main task; but in the sparse reward setting this tipping point is reached much more slowly, which causes the auxiliary task to have greater influence than it otherwise would on the network's weights early on, until the tipping point is reached towards the end of the run and the main task learning starts to dominate the weight update procedure.

6.3 Rich Stochastic Gridworld

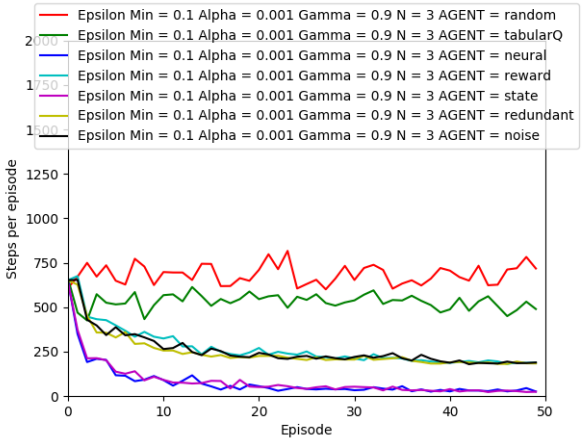

Figure 5: Rich Stochastic Gridworld Results

Results for the rich stochastic gridworld differ little from the rich deterministic case (See figure 5). The fact that the state prediction task clearly outperforms the reward prediction task in both the stochastic and deterministic setting suggests that the superior performance of the former may have less to do with correctly anticipating low value stochastic states and more to due with interfering less with the main task in some sense.

6.4 Sparse Stochastic Gridworld

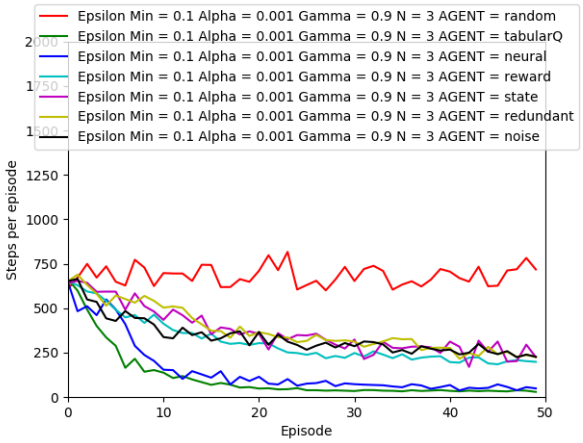

Figure 6: Sparse Stochastic Gridworld Results

Finally, as in the previous cases, none of the auxilary tasks served to improve agent performance. Moreover, in contrast to the sparse deterministic case there is little in the way of any possibly significant variation between tasks. All of them appear to perform rather poorly throughout the entirety of the run, including state prediction. In fact, this lackluster performance on the part of state prediction is somewhat anomalous in light of the previous results, and serves to demonstrate that of all the auxiliary tasks state prediction seems to be the most sensitive to changes in the environment specification.

7 Conclusion And Future Work

In sum, none of the auxiliary tasks demonstrated significant improvement over the single task agent; nor did the patterns of effect of the tasks elucidate a straightforward systematic relationship between any of the tasks and specific environment characteristics. The state prediction task did demonstrate a relatively consistent prima facie superiority relative to other tasks, and it may be profitable to further explore why this is the case by studying the actual policies and/or representations that the agent is learning given different auxiliary tasks and environment combinations, so as to better understand the manner in which these tasks are effecting the network weight update procedure.

Moreover, gridworld is neither feature rich nor particularly suitable for function approximation, given its finite and discete nature, and was chosen primarily for its conceptual and implementational simplicity as a starting point. As such, it is plausible that the failure of auxiliary tasks to prove beneficial is due to the lack of predictive features that there are to train on in the environment, and that in order for auxiliary tasks to be useful a certain threshold of complexity in terms of state representation and/or transition dynamics must be present with respect to the current environment. It would therefore be prudent to further explore auxiliary tasks within a number of other environments, progressively increasing their complexity along a number of dimensions in order to identify at which point, and under what conditions, auxiliary tasks start to become useful. Two environment variations that immediately suggest themselves for future study are that of (non-visual) continuous state spaces, due to their appropriateness for function approximation methods, as well as environments with raw visual state representations that are more likely to possess many features for the auxiliary tasks to learn from. Both of these environmental variations are good directions for future work.

8 ACKNOWLEDGMENTS

The author would like to thank Martha White for their commentary and constructive conversation concerning the present work.

REFERENCES

Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

Héctor Martínez Alonso and Barbara Plank. Multitask learning for semantic sequence prediction under varying data conditions. In *Association for Computational Linguistics European Chapter*, April 2017.

Jonathan Baxter. A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12(149-198):3, 2000.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pp. 567–580. Springer, 2003.

Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Association for Computational Linguistics European Chapter*, April 2017.

Rich Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, July 1997. ISSN 0885-6125. doi: 10.1023/A:1007379606734. URL `https://doi.org/10.1023/A:1007379606734`.

François Chollet et al. Keras. `https://github.com/keras-team/keras`, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, April 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *International Conference on Learning Representations*, May 2016.

Ning Situ, Xiaojing Yuan, and George Zouridakis. Assisting main task learning by heterogeneous auxiliary tasks with applications to skin cancer screening. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 688–697, 2011.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning : An Introduction*. MIT Press, 2017.

Brian Tanner and Adam White. Rl-glue: Language-independent software for reinforcement-learning experiments. In *Journal Of Machine Learning Research*, pp. 2133–2136, 2009.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.