Discovering Control-Intensive State Regions With Deeper Policy Networks

Juho Ylä-Jääski*
Department of Computer Science
Aalto Univeristy
Espoo, Finland
juho.yla-jaaski@aalto.fi

Otto Solatie*
Department of Computer Science
Aalto University
Espoo, Finland
otto.solatie@aalto.fi

Sanna Laitinen*
Department of Computer Science
Aalto University
Espoo, Finland
sanna.m.laitinen@aalto.fi

Nam Hee Kim
Department of Computer Science
Aalto University
Espoo, Finland
namhee.kim@aalto.fi

Abstract

A key knowledge gap in deep reinforcement learning (DRL) is the role of depth in policy networks. While the conventional 2-layered fully-connected networks (FCNs) tend to work sufficiently well for most tasks, we hypothesize that additional layers contribute to the expressivity of FCN policies by introducing finer-grained mapping from the state space to actions. Our fine-tuning experiments isolate the effects of additional layers for a DRL agent deployed on a simple 1D toy car problem. Both qualitative and quantitative analyses are presented to examine the evolution of added layers as fine-tuning progresses. Our results indicate that deeper layers afford more granularity in state-action mappings to control-intensive regions in the state space.

1 Introduction

Although deep reinforcement learning (DRL) has become the current state-of-the-art method for intelligent control, the relationship between the constituent deep neural networks' architectures and DRL policies' performance is surprisingly poorly understood. This is an especially jarring gap, given the successful research on deep learning in recent years in the context of supervised and unsupervised learning. Shallower (often 2-layer) fully-connected networks (FCNs) are often preferred as a policy architecture within the DRL community for the sake of simplicity, as added depths do not necessarily yield performance gains. Incidentally, the question of *which benefits exactly are offered by a deeper policy to DRL agents and algorithms* remains an under-explored topic. In this exploratory work, we aim to gain visual insights into the role of added depth in DRL policy networks. Our main research question can be summarized as follows: *how does adding depth to a policy network affect the expressivity of the policy?*

In DRL, a policy's expressivity corresponds to how adeptly the agent can change its action output as a function of state input. In other words, an expressive policy means that the agent possess a high *granularity* of control. It is a well-established fact that adding depth to neural networks increases the overall expressivity, but in the context of DRL, little attention is paid to *where the additional expressivity is spent*. While DRL policies are often treated as black-box function approximators,

*These authors contributed equally to this work.

Preprint. Under review.

we aim to pay closer attention to the inner workings of a policy network. We examine networks of different sizes and incorporate transfer learning to better isolate the effects of network depth. In so doing, we establish an incremental groundwork towards understanding how a policy's expressivity gain due to added depth relates to the control task being solved.

Our contributions can be summarized as follows:

- We provide an incremental follow-up to existing investigations of linear regions in DRL policy networks. We offer qualitative and quantitative insights regarding how an added depth may affect a policy's control granularity. Namely, we discover that additional layers may help discover decision boundaries that indicate regions within the state space that are more control-intensive.
- We examine used cell ratio (UCR) and visited cell counting as quantitative measures of efficiency in state space division patterns. We discover that, although UCR in itself does not scale well to larger policies, visited cell counting helps contextualize UCR values.

2 Related Work

The expressivity of neural networks in supervised learning has been actively studied in recent years. One proposed tool (Arora et al., 2016) for measuring the expressivity is the number of so-called *linear regions* of the input space emerging in neural networks that use rectified linear (ReLU) activation functions (Nair and Hinton, 2010). Previous studies on the effect of network depth (Pascanu et al., 2013; Montufar et al., 2014) on the number of linear regions have suggested that the number of linear regions can grow exponentially with network depth. Tight upper and lower bounds for the number of linear regions for networks using ReLU activation functions have been proposed by Serra et al. (2018). However, recent works (Hanin and Rolnick, 2019a,b) on the effect of network depth on the number of linear regions suggest that, in practice, the depth of a neural network has far less importance on the number of linear regions and that the number of linear regions is determined primary by the number of neurons in the network.

While most effort in studying the expressivity of neural networks has been focused on supervised learning tasks, for deep reinforcement learning (DRL) algorithms, the topic is under-explored. Some attempts have been made to visualize and analyze the agent behavior. Rupprecht et al. (2019) developed a methodology where they learn a generative model of the environment as an input to the agent. This methodology is used to visualize and understand states where agent behavior is below optimal.

Another study from Luo et al. (2018) developed visualization techniques for evaluating convolutional neural networks (CNN)-based DRL algorithms. They address the problem that, in many DRL algorithms, the action is selected based on probabilities and this should be taken into account when applying visual diagnostics. These visualization techniques are used to better understand agents' behaviours and understand their weaknesses.

We follow the recent successful investigation of Cohan et al. (2022) in examining the relationship between a DRL algorithm's progress and the evolution of linear regions in the RL environment's state space. Our work adds transfer learning and fine-tuning in order to isolate the effects of added depth in terms of the granularity of the decision boundaries in deeper policy networks.

3 Problem Statement

For the toy DRL environment, we elect to use a 1D car problem as employed in Cohan et al. (2022). We modify the environment such that we have a continuous state space and a discrete action space. In this environment (Figure 2), a car in a 1-dimensional world, whose state is parameterized by its x-position (x) and x-velocity ($\dot{x}$), starts from the neighbourhood of $(x = 0, \dot{x} = 0)$. The goal of the agent is to park this car at $x = 100$, given a finite horizon of 200 simulation steps. In each step, the agent chooses among three actions: `{accelerate (+1), neutral (0), decelerate (-1)}`. The car's peak speed is limited to 10 units/step.

We consider examining the division of the state space for a toy DRL environment. Divisions with linear regions (or *cells*) emerging from ReLU-based networks are promising visualization tools

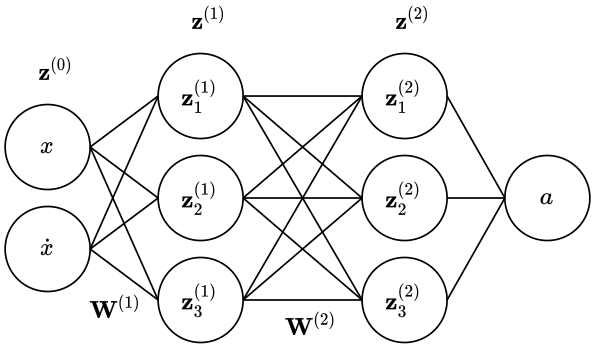

Figure 1: Policy neural network with the inputs x and $\dot{x}$, layer activations $\mathbf{z}$, weigths $\mathbf{W}$, and the output action a.

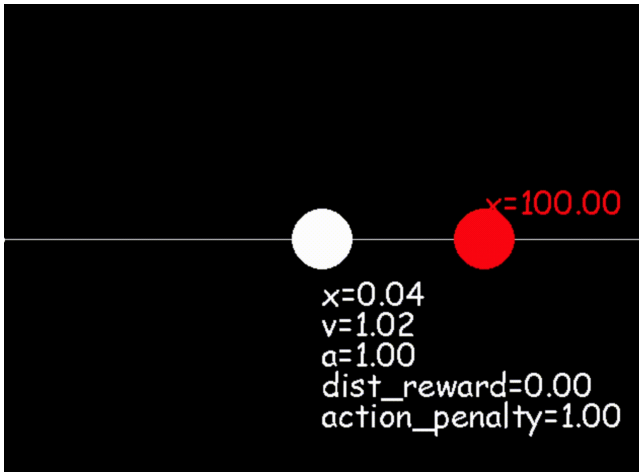

Figure 2: The environment can be visualized with rendering. Here the white circle represents the car and the red circle denotes the goal.

(Hanin and Rolnick, 2019a). Intuitively, regions in the state space, as perceived by the policy network, indicate neighbourhoods of states that have overlapping information content when it comes to choosing appropriate action.

More formally, we populate a grid of states within a fixed window $x \in \{-20, 130\}$ and $\dot{x} \in \{-10, 10\}$. We then compute a *binary embedding* for each state. The binary embeddings are computed based on the output value of the hidden neurons of the neural network, which are variously activated in different points in that space. Neurons that output positive values are considered to be 1's and others are 0's. These values are concatenated together to create binary embeddings. Mathematically, a layer's activation can be represented as

$$\mathbf{z}^{(i)} = \text{ReLU}\left(\mathbf{W}^{(i)}\mathbf{z}^{(i-1)}\right), \quad i > 0, \tag{1}$$

where i is the layer index and $\mathbf{W}^{(i)}$ represents the weights and biases corresponding to layer i. The binary embedding of a neuron j of layer i can be formally written as follows:

$$\text{BinEmb}\left(\mathbf{z}_j^{(i)}\right) = \begin{cases} 1 & \text{if } \mathbf{z}_j^{(i)} > 0 \\ 0 & \text{if } \mathbf{z}_j^{(i)} \leq 0. \end{cases} \tag{2}$$

The binary embeddings across the hidden neurons and across layers are then concatenated together. In addition to the hidden neuron activations, we attach the one-hot embeddings of the environment's discrete actions. The binary embeddings are then converted via colour hashing to create RGB colour values for visualization. As a result, the state space is divided into differently-coloured regions as

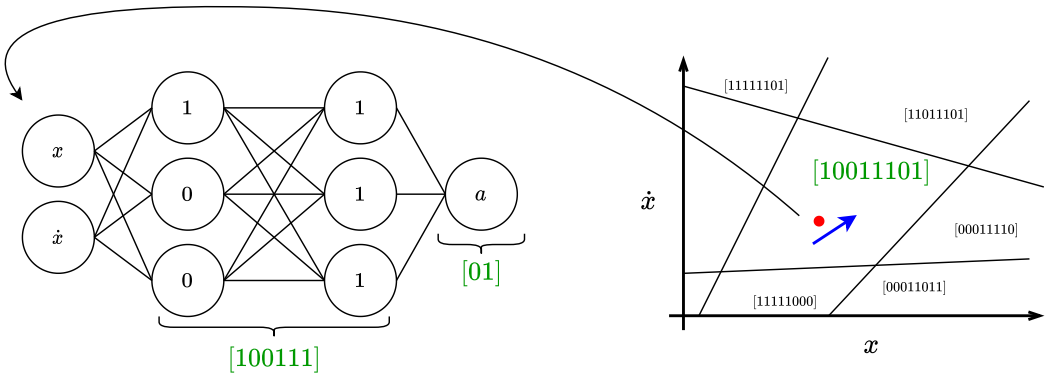

Figure 3: Constructing the binary embedding. The activations at hidden neurons of the policy network are turned into a binary code according to their values. The state space is divided based on these codes. The action dictates where the agent moves in the state space. The action is added as a binary string to the end of the binary embedding from the layers.

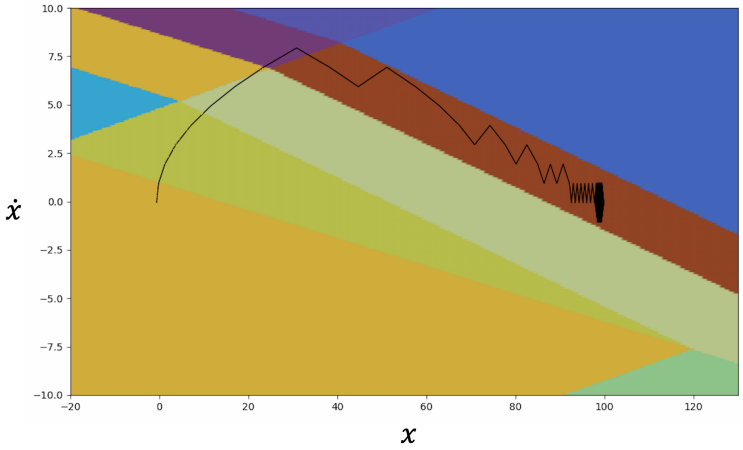

Figure 4: Example state space division into binary embedding cells. The black line is the trajectory in the state space taken by the agent.

shown in Figure 4. Notably, the emerging cells are constrained to be linear regions according to ReLU activations, which consequentially make each cell a convex polytope, effectively forming a Voronoi diagram of the state space.

Our analysis is heavily based on the number of visited cells that the agent's trajectory encounters. We also derive another quantity called used cell ratio (UCR), which simply divides the number of visited cells by the total number of cells in the studied window of state space. UCR is meant to quantify how efficiently the policy divides the state space into cells. Intuitively, a high UCR indicates that many of the cells in the state space are used actively for the agent's control. An efficient cell division would correspond to a large number of cells dedicated to regions where more effort or maneuverability is required for the given task. In our toy car example, such a control-intensive region could be located where the agent has to start slowing down after accelerating initially. We hypothesize that added depth would result in an increase in UCR as high-density divisions would occur along the optimal trajectory.

Given these proposed tools, we aim to answer the following research question: *how does the effect of added depth for a policy network show up in state space divisions?* To dive into this question, we use a transfer learning technique between a shallower teacher policy and a deeper student policy, while only allowing for added layers to learn. In our teacher-student learning paradigm (Figure 5), we copy the weights and biases of a teacher's layers to a student. The teacher's sample trajectories

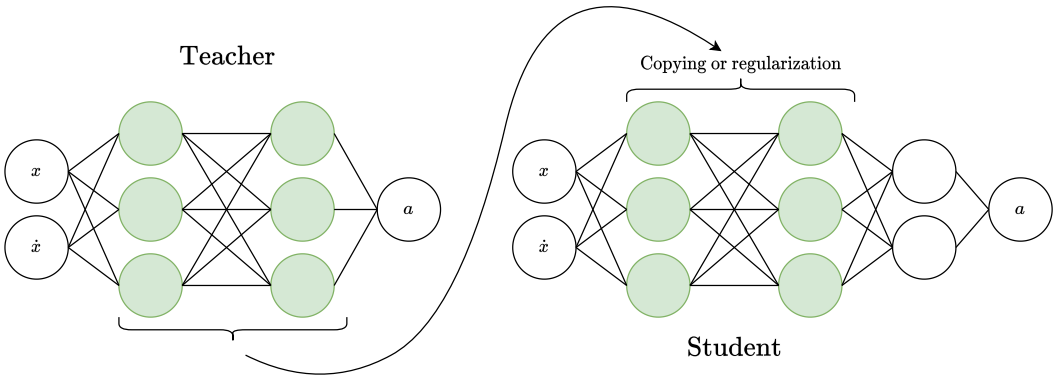

Figure 5: Our fine-tuning scenario. A teacher is trained and its layers are used in the student policy network. The student has extra trainable layers that are randomly initialized. The teacher's influence can be applied by copying and freezing the layers or regularizing the student. Here, we focus on copying.

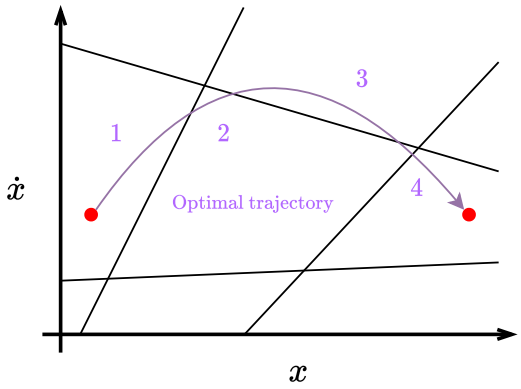

Figure 6: Illustration of visited state cells. The agent follows the trajectory and goes through some of the state cells. UCR is computed as the number of visited cells divided by the total number of cells.

are then used as expert guidance for the student, whose last layers are trained to imitate the teacher's actions. We hypothesize that the evolution of state divisions of the trainable layers over time will show distinct patterns, allowing us to examine the effects of additional layers and understand the role of depth in policy networks.

During the training of both teacher and student policies, we generate the binary embeddings after each episodic iteration of PPO, hence tracking the evolution and movement of cells along the training progress. Based on the resulting binary embeddings, we compute UCR after each iteration to evaluate whether the fine-tuning converges the student policies towards improving the efficiency in state space divisions. The UCR computation is based on the controlled trajectory of the agent according to a policy snapshot at the corresponding PPO iteration.

4 Implementation

We implement two fully-connected networks (FCNs) for teacher policies and two for student policies. The sizes of the teachers are 2×2 and $64 \times 64 \times 2$ whereas the students have dimensions of $2 \times 2 \times 2 \times 2$ and $64 \times 64 \times 2 \times 2$. The notation indicates the number of hidden neurons per layer; for example, $64 \times 64 \times 2 \times 2$ indicates hidden layers of sizes 64, 64, 2, and 2. These sizes are chosen so that the numbers of parameters would be similar while varying depth, i.e. the students have added depth without excessively many extra weights in the added layers. In particular, adding a layer of 2

hidden neurons on top of a previous layer of 2 hidden neurons introduces 4 weights and 2 biases, which we consider to be the minimum number of parameter increase from adding a layer.

The teacher networks are first trained from scratch using Proximal Policy Optimization (PPO) implementation provided by Stable Baselines3 (SB3) (Raffin et al., 2021). A teacher network's weights and biases are then used as the early layers of a student network, whose additional layers are randomly initialized. Then the students' trainable layers are trained via PPO, with the weights and biases copied from the teacher networks fixed as constants. Both teachers and students are trained for 240,000 iterations, for which a 2×2 teacher of every seed converged to a reasonable performance level in terms of cumulative reward. All of our experiments were conducted on Aalto University's Triton cluster using CPUs.

5 Results

Our qualitative and quantitative analyses offer insights as to how adding depth to a policy network affects its behaviour and perception. In both cases, as mentioned in Section 3, we examine the division of the state space according to the activations in the policy's hidden layers as well as actions. Quantitatively, we determine whether the deeper policies' state space divisions are meaningful by examining the evolution of UCR and the characteristics of the resulting controlled trajectory. We note that the analyses offer insight about the policy's behaviour, although the performance in terms of the observed cumulative reward is not significantly affected by the added depth, as even the simplest 2×2 policy network can solve the environment.

5.1 Qualitative analysis

Every snapshot of teacher and student policy networks has a corresponding state space division as perceived by the policy. We are interested in seeing how the trainable layers of the student networks can edit the original division given by the teacher. In Figure 7, we compare how the state space divisions differ across teachers, trainable student layers, and full student layers. The teacher's hidden dimensions are 2×2. As shown in Figure 7, the student's trainable layers (middle column) induce their own state space division patterns, which are then collated to create more complex state divisions in the full layers (third column). The pattern of the trainable layers are particularly noteworthy, as they reveal how the students' trainable layers identify places where an active edit to the embeddings and actions might be required. Remarkably, the trainable layer visualizations maintain coherent patterns across different random seeds, where additional embeddings roughly divide the space into top-right and bottom-left regions.

The evolution of these state space division plots are also notable, as they show the relationship between a policy snapshot's controlled trajectory and the corresponding state space division after each iteration of PPO. The animations can be found on our project website: `https://ylajaaski.github.io/state_space_page/`. In Figure 8, we compare the state space divisions according to the last student layers, hence visualizing the rough course of their evolutions. We note that different initializations lead gradually to the qualitatively similar pattern, as also seen in Figure 7.

5.2 Quantitative analysis

In Section 5.1, we unsurprisingly observed that added layers induce more divisions of the state space according to the full network activations. Then, *does the increase in the number of overall cells show any meaningful pattern?* As discussed in Section 3, we propose to use the number of visited cells and UCR as the measure of occupancy. Intuitively, if the finer divisions induced by added layers are not useful, the cells would be placed randomly, and the agent may not traverse these new cells. In this section, we examine how those two quantities evolve over training iterations. In Figure 9, the cumulative reward and UCR are plotted as functions of training iterations for our teacher-student learning scenarios. The cumulative reward curves clearly mark the iterations at which policies across all runs converge to a reasonable performance. We can see that in the case of a 2×2 teacher policy, UCR tends to increase slightly in the beginning of training, whereas the opposite happens when the teacher is of size 64×64. Additionally, there's no significant difference between the UCRs of the teacher and the student.

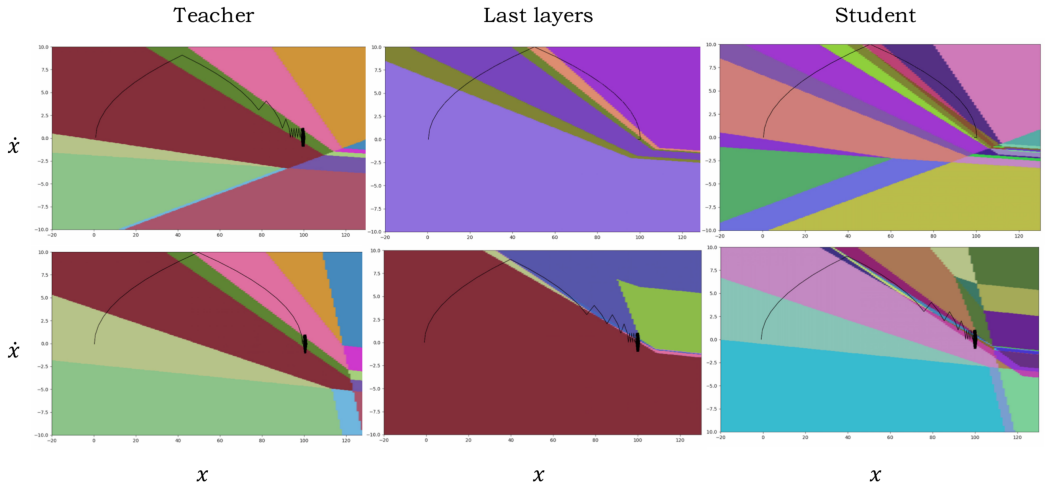

Figure 7: Cell divisions of a 2×2 teacher, the last layers of a $2 \times 2 \times 2 \times 2$ student, and the student itself. The dark line in each plot indicates the mean trajectory produced by the policy snapshot starting at $(0, 0)$. The rows correspond to different random seeds.

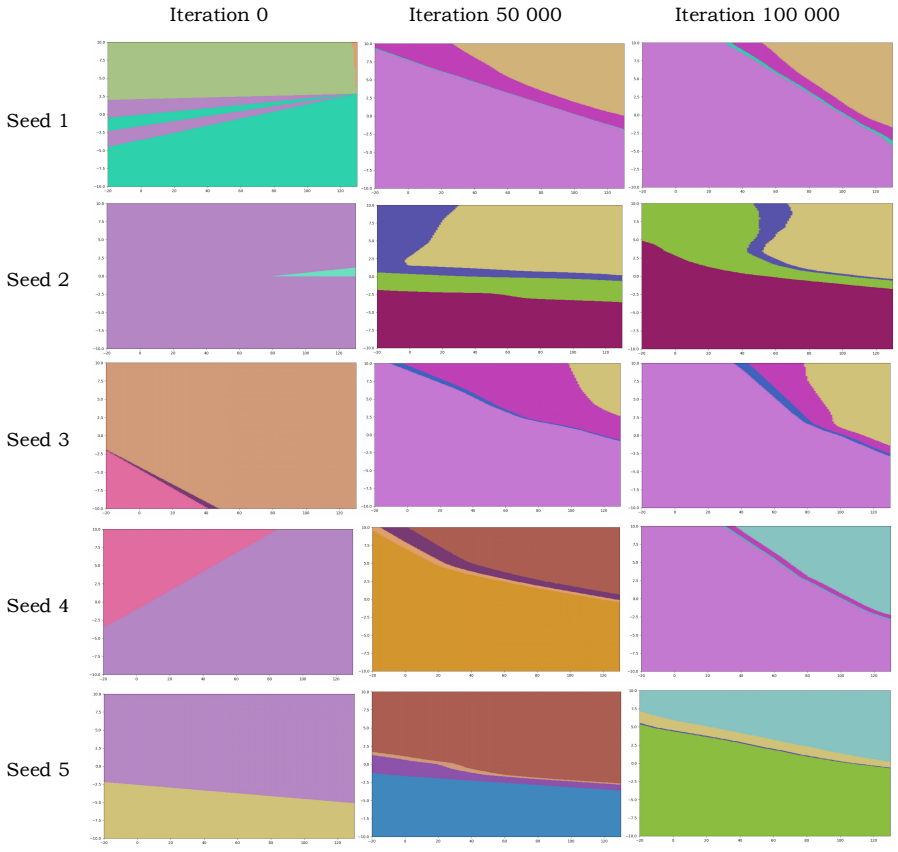

Figure 8: State space divisions according to the trainable layers of $64 \times 64 \times 2 \times 2$ students over training iterations. Across different random seeds, the trainable layer's embeddings are organized from random patterns to a cohesive division between upper-left and lower-right regions.

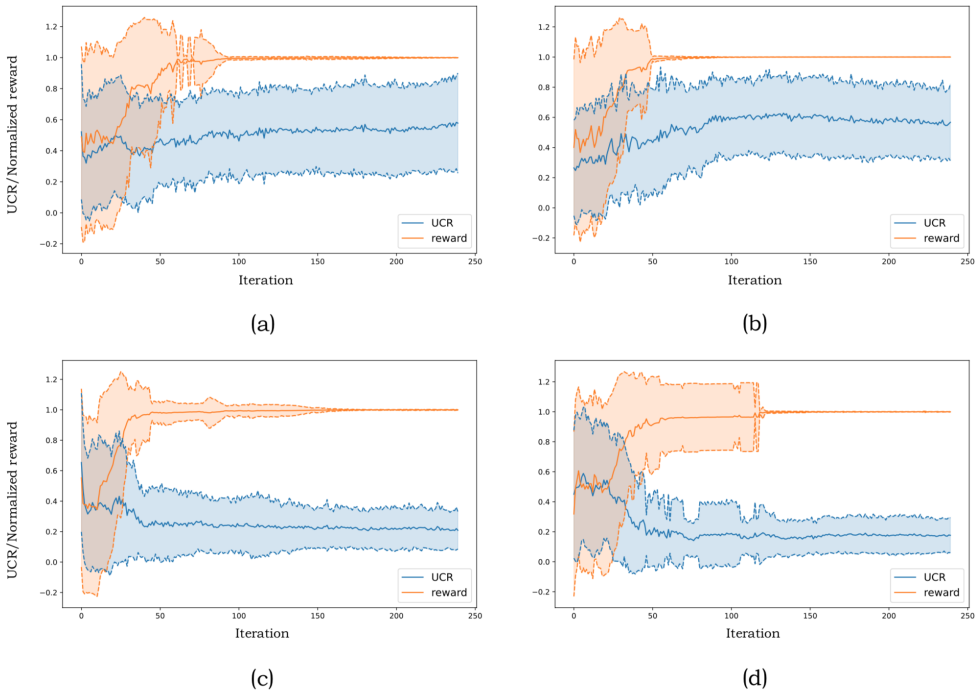

Figure 9: UCR and cumulative reward over training iterations. (a) corresponds to the 2×2 teacher, (b) is the $2 \times 2 \times 2 \times 2$ student, whereas (c) is the $64 \times 64 \times 2$ teacher, and (d) is the $64 \times 64 \times 2 \times 2$ student. Converging cumulative rewards indicate a successful training. The mean UCR, on the other hand, stays relatively constant in all scenarios.

While examining UCR in isolation does not seem to lead us to a concrete pattern, we also look into the counts of visited cells during training, as shown in Figure 10. The opaque line in each plot of Figure 10 represents the mean of several random seeds. It can be noted that in all four cases, in both teachers and students, the mean count increases in the beginning but converges to a relatively stable value. Individual models have a considerable amount of oscillations in their visited cell counts, i.e. some random seeds are considerably above and some considerably below the mean curve. However, visited cell counts for student layers show a coherent pattern of increasing sharply in the beginning and then showing some oscillations before staying mostly converged afterwards. In the case of the larger policy, the total number of cells, on average, goes up sharply in the beginning causing UCR to decrease. Unsurprisingly, the smallest the model, in (a), has the most noise as there are inherently fewer cells.

We further assess whether the additional embeddings provided by students' trainable layers are meaningful compared to a random baseline. Intuitively, a smart student would improve its teacher's existing cell division by introducing new cells where they are most needed, instead of introducing them in completely random ways. In Figure 11, we use a $64 \times 64 \times 2$ teacher policy and spawn multiple students with random initialization across multiple seeds. The visited cell counts are measured first after initialization and then after a fixed number of iterations of PPO training. The trained students clearly visit more cells than the initialized ones as the histogram of visited cell counts shifts to the right as a result of training.

Finally, we provide a visual explanation for the observed behaviour in UCR and number of visited cells. In Figure 12(b), one can see how the trajectory actually tends to go through areas that are sparsely populated by cells whereas the the densest regions are elsewhere. The same conclusions does not apply in the smaller network as shown in Figure 12 (b). Figure 12 demonstrates visually why UCR is lower for the larger policy. The number of total cells in the full student layers increase drastically in larger networks, although the last layers of the student policy are still organized in

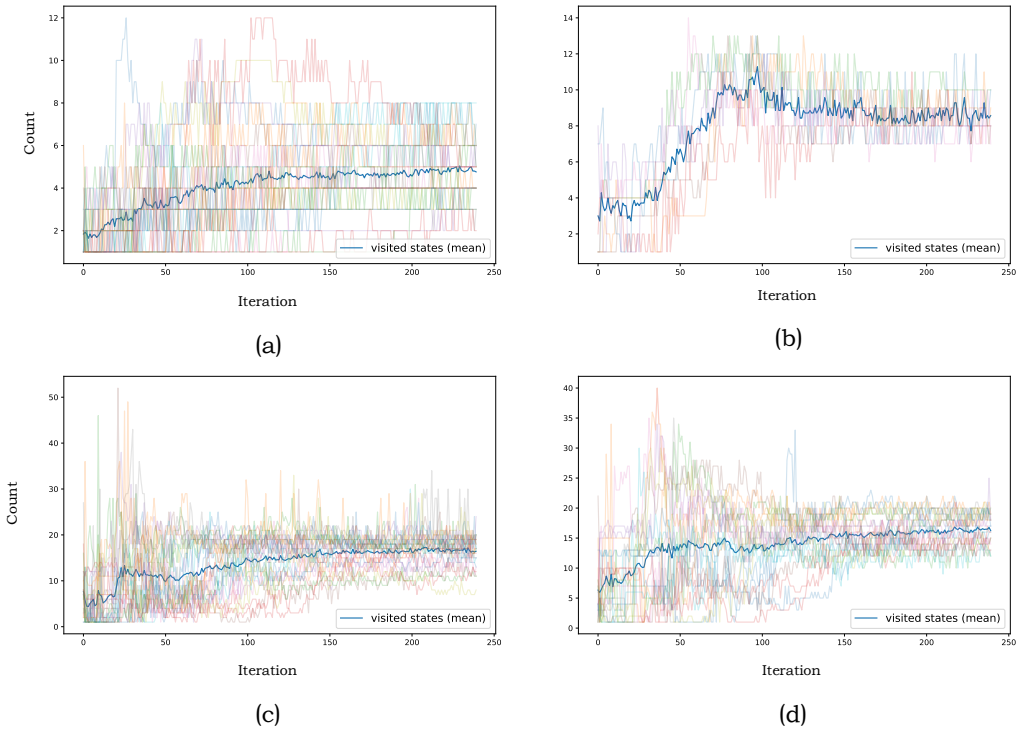

Figure 10: Count of visited states. The opaque line in each plot represents the mean of the transparent ones. Here, the students were trained. The mean is going up quite consistently over the early iterations. (a) corresponds to the 2×2 teacher, (b) is the $2 \times 2 \times 2 \times 2$ student, whereas (c) is the $64 \times 64 \times 2$ teacher, and (d) is the $64 \times 64 \times 2 \times 2$ student.

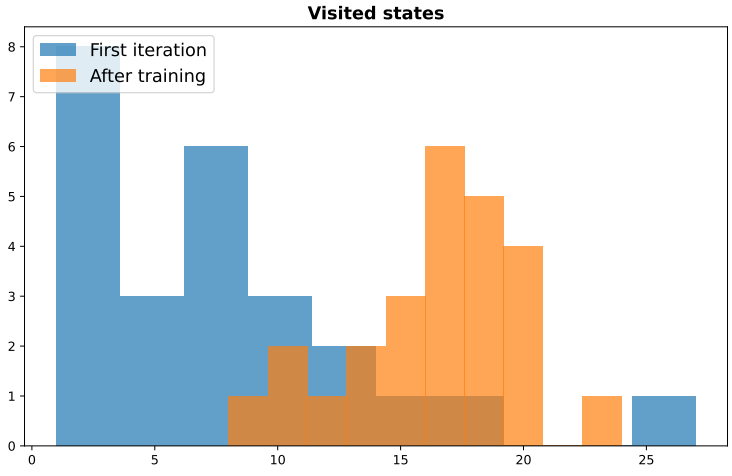

Figure 11: Baseline experiment showing how the count of visited cells evolves. Here, one teacher is used to generate multiple students and their counts are measured before and after training. The distribution of visited cells clearly shifts to the right due to training.

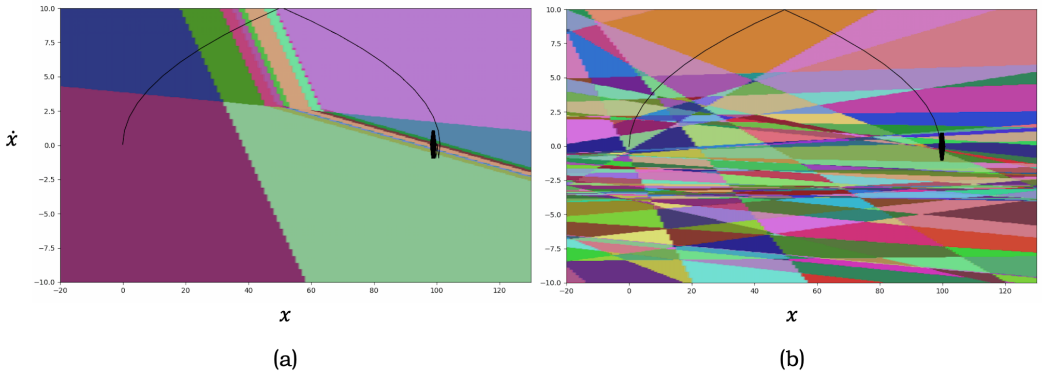

Figure 12: State space divisions after training. (a) corresponds to the $2 \times 2 \times 2 \times 2$ student and (b) is the $64 \times 64 \times 2 \times 2$ student.

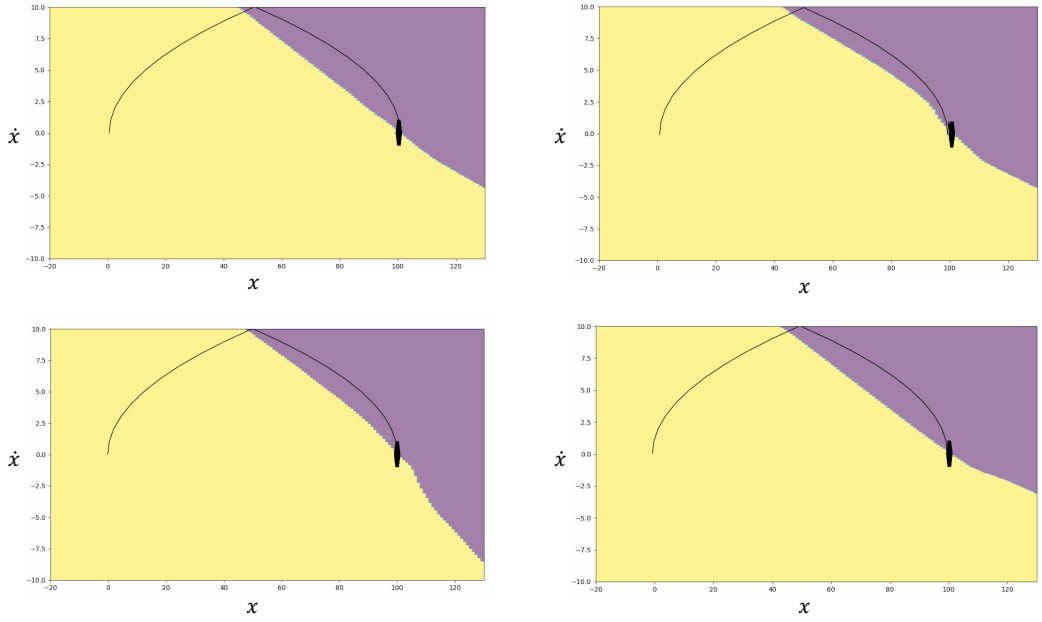

Figure 13: Examples of action boundary in the state space with three different random seeds. Here, yellow refers to positive action and purple to negative action. On the left column we have 64×64 teachers and on the right column we have $64 \times 64 \times 2 \times 2$ students. The teachers and students on the same row correspond to each other. Qualitatively, the action boundaries look very similar across seeds, network sizes, students, and teachers.

such a way the resulting control trajectory still traverses the new cells. Consequentially, it is evident that UCR is only an appropriate heuristic in smaller policies. A robust and scalable measure of cell occupancy will be an interesting direction in future work. However, having examined visited cell counts and UCR together, we are able to conjecture that the students' added layers are introducing their new cells in a logically coherent manner that helps the policy optimize its behaviour, hence increasing cell occupancy, albeit less effectively in a larger network.

6 Discussion

In the previous section, we discovered that the trainable layers of student policies, with the minimum parameter increase possible, show consistent patterns in editing the state divisions of the teacher policies. Given the role of depth in neural networks in general, this is an unsurprising finding; added depth is bound to increase the number of linear regions and therefore the capacity of a network. However, we take particular note of the allocation of state division boundaries as determined by the iterations of a DRL algorithm. Namely, we detected a pattern where a particular upper-right and lower-left regions according to students' trainable layers emerge across different random seeds. Intuitively, given the limited capacity increase, the student policy must attempt to edit the teacher's decision boundaries as efficiently as possible. Then, where the student policy chooses to allocate the additional boundary may indicate where in the state space the minimal amount of edit makes the most difference in performance, albeit negligible gains overall.

Visualizing the state space division as a decision boundary, as in Figure 13, helps us understand the role of the trainable layers of the student policies. These decision boundaries are created by coloring each linear region with their output action. Comparing the difference between teacher and student policy boundaries, we see that student policies change the shapes of the decision boundaries in subtle ways. Primarily, the additional boundaries are allocated close to teachers' existing boundaries between positive and negative actions, while abiding by the division between the upper-right and lower-left regions. The state space divisions in rightmost column in Figure 8 corroborates this behaviour. Intuitively, sample trajectories encountered during training inform the students' trainable layers to pay closer attention to when to stop accelerating and start decelerating. In terms of control, this boundary could signify where intensive control is needed in this environment. Our experiments hint at the possibility that the extra layers for deeper policies help afford more fine-grained control in such regions.

It is known that a deeper policy network does not necessarily improve RL agents performance. We demonstrated that the added depth into a policy network, however, has a non-null effect. Possibly, added layers could be introducing more fine-grained state space cells divisions which imply more granular control. We also presented UCR as a potential measure of quantifying the usefulness of the cell divisions. However, we observed that UCR does not robustly increase in larger networks, due to the overwhelming number of irrelevant cells in the state space. Counting the visited cells, meanwhile, seems to provide a reliable measure of cell occupancy, although this does not take the overall size of the policy network into account. We hope to see further efforts into quantifying the efficiency of the state space division in DRL.

As we conclude this paper, we remark that understanding the inner working of DRL policies and algorithms is far from solved. We hope to see further investigations in this direction, which could bring benefits especially in improving sample efficiency in DRL. For example, if the difficult control regions were known (e.g., by the density of state space cells) state sampling could be focused on those regions. Understanding the control landscape could have consequences in human-assisted training and environment design as well. In the former case, a human could steer the training by providing better signals. In the latter case, the designer could engineer the environment difficulty in a more rigorous manner, with the additional insight provided by a policy's perceived difficulties or control intensities present in given tasks.

References

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. 2016. Understanding Deep Neural Networks with Rectified Linear Units. (2016). `https://doi.org/10.48550/ARXIV.1611.01491`

Setareh Cohan, Nam Hee Kim, David Rolnick, and Michiel van de Panne. 2022. Understanding the Evolution of Linear Regions in Deep Reinforcement Learning. (2022).

Boris Hanin and David Rolnick. 2019a. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*. PMLR, 2596–2604.

Boris Hanin and David Rolnick. 2019b. Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems* 32 (2019).

Jieliang Luo, Sam Green, Peter Feghali, George Legrady, and Cetin Kaya Koç. 2018. Visual diagnostics for deep reinforcement learning policy development. *arXiv preprint arXiv:1809.06781* (2018).

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. 2014. On the number of linear regions of deep neural networks. *Advances in neural information processing systems* 27 (2014).

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Icml*.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. 2013. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098* (2013).

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. `http://jmlr.org/papers/v22/20-1364.html`

Christian Rupprecht, Cyril Ibrahim, and Christopher J Pal. 2019. Finding and visualizing weaknesses of deep reinforcement learning agents. *arXiv preprint arXiv:1904.01318* (2019).

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. 2018. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*. PMLR, 4558–4566.