Structured Control Nets for Deep Reinforcement Learning

Mario Srouji* [1] **Jian Zhang*** [2] **Ruslan Salakhutdinov** [1] [2]

Abstract

In recent years, Deep Reinforcement Learning has made impressive advances in solving several important benchmark problems for sequential decision making. Many control applications use a generic multilayer perceptron (MLP) for nonvision parts of the policy network. In this work, we propose a new neural network architecture for the policy network representation that is simple yet effective. The proposed Structured Control Net (SCN) splits the generic MLP into two separate sub-modules: a nonlinear control module and a linear control module. Intuitively, the nonlinear control is for forward-looking and global control, while the linear control stabilizes the local dynamics around the residual of global control. We hypothesize that this will bring together the benefits of both linear and nonlinear policies: improve training sample efficiency, final episodic reward, and generalization of learned policy, while requiring a smaller network and being generally applicable to different training methods. We validated our hypothesis with competitive results on simulations from OpenAI MuJoCo, Roboschool, Atari, and a custom urban driving environment, with various ablation and generalization tests, trained with multiple black-box and policy gradient training methods. The proposed architecture has the potential to improve upon broader control tasks by incorporating problem specific priors into the architecture. As a case study, we demonstrate much improved performance for locomotion tasks by emulating the biological central pattern generators (CPGs) as the nonlinear part of the architecture.
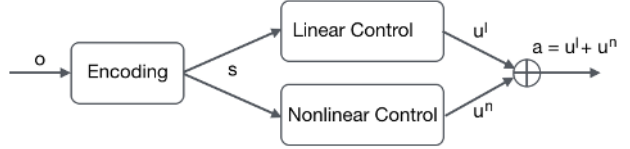
---

*Equal contribution [1]Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213. [2]Apple Inc., 1 Infinite Loop, Cupertino, CA 95014. This work is done during Mario's internship at Apple. Correspondence to: Jian Zhang, Ruslan Salakhutdinov <jianz@apple.com, rsalakhutdinov@apple.com>.

Figure 1. The proposed Structured Control Net (SCN) for policy network that incorporates a nonlinear control module, u^n, and a linear control module, u^l. o is the observation, s is encoded state, and a is the action output of the policy network. Here time t is dropped for notation compactness.

1. Introduction

Sequential decision making is crucial for intelligent system to interact with the world successfully and optimally. Deep Reinforcement Learning (DRL) recently has made significant progress on solving several important benchmark problems, such as Atari (Mnih et al., 2015), the game of Go (Silver et al., 2017), high-dimensional continuous control simulations (Schulman et al., 2017; Lillicrap et al., 2015), and robotics (Levine et al., 2016). Many successful applications, especially control problems, still use generic multilayer perceptron (MLP) for non-vision part of the policy network. There have been only few efforts exploring adding specific structures to the policy network as an inductive bias for improving training sampling efficiency, episodic reward, generalization, and robustness (Wang et al., 2016).

In this work, we focus on an alternative but complementary area by introducing a policy network architecture that is simple yet effective for control problems (Figure 1). Specifically, to improve the policy network architecture, we introduce control-specific priors on the structure of the policy network. The proposed Structured Control Net (SCN) splits the generic multilayer perceptron (MLP) into a nonlinear control stream and a linear control stream that combine additively into the final action. This approach has the benefit of being easily combined with many existing DRL algorithms and complementary to the methods with structures for planning (Tamar et al., 2016; Oh et al., 2017).

Without any special treatment, we extensively tested SCN across 3 types of training method (Evolutionary Strategies (Salimans et al., 2017), PPO (Schulman et al., 2017), ACKTR (Wu et al., 2017)) and 4 types of environment (MuJoCo (Brockman et al., 2016; Todorov et al., 2012), Roboschool (OpenAI, 2017), Atari (Bellemare et al., 2013),

simulated urban driving). Our results demonstrate that this architecture brings together the benefits of both linear and nonlinear policies by improving training sampling efficiency, final episodic reward, and generalization of learned policy, while using a smaller network compared to standard MLP baselines. Our ablation tests also show that sub-modules of the architecture are effectively learned. This structured policy network has the potential to be expanded to broader sequential decision making tasks, by incorporating problem specific priors into the architecture.

2. Related Work

Discrete control problems, such as Atari games, and high-dimensional continuous control problems are some of the most popular and important benchmarks for sequential decision making. They have applications in video game intelligence (Mnih et al., 2015), simulations (Schulman et al., 2017), robotics (Levine et al., 2016), and self-driving cars (Shalev-Shwartz et al., 2016). Those problems are challenging for traditional methods, due to delayed rewards, unknown transition models, and high dimensionality of the state space.

Recent advances in Deep Reinforcement Learning (DRL) hold great promise for learning to solve challenging continuous control problems. Popular approaches are Evolutionary Strategy (ES) (Salimans et al., 2017; Such et al., 2017; Conti et al., 2017) and various policy gradient methods, such as TRPO (Schulman et al., 2015), A3C (Mnih et al., 2016), DDPG (Lillicrap et al., 2015; Plappert et al., 2017), PPO (Schulman et al., 2017), and ACKTR (Wu et al., 2017). Those algorithms demonstrated learning effective policies in Atari games and physics-based simulations without using any control specific priors.

Most existing work focuses on training or optimization algorithms, while the policy networks are typically represented as generic multilayer perceptrons (MLPs) for the non-vision part (Schulman et al., 2015; Mnih et al., 2016; Salimans et al., 2017; Schulman et al., 2017; Wu et al., 2017). There are only few efforts exploring adding specific structures to DRL policy networks, such as inductive bias for improving sampling efficiency, episodic reward, and generalization (Wang et al., 2016).

Some of the recent work also focused on studying network architectures for DRL. The Dueling network of (Wang et al., 2016) splits the Q-network into two separate estimators: one for the state value function and the other one for the state-dependent action advantage function. They demonstrated state-of-the-art performance on discrete control of the Atari 2600 domain. However, the dueling Q-network is not easily applicable to continuous control problems. Here, instead of value or Q network, we directly add continuous control-specific structure into the policy network. (Tu & Recht,

2017) studied the sampling efficiency of least-squares temporal difference learning for Linear Quadratic Regulator. Consistent with some of our findings, (Rajeswaran et al., 2017; Mania et al., 2018) showed that a linear policy can still achieve reasonable performance on some of the MuJoCo continuous control problems. This also supports the idea of finding a way to effectively incorporate a linear policy into the architecture.

The idea of splitting nonlinear and linear components can also be found in traditional feedback control liturature (Khalil, 1996), with successful applications in control of aircrafts (Camacho & Alba, 2013), robotics, and vehicles (Thrun et al., 2006); albeit, those control methods are not learned and are typically mathematically designed through control and stability analysis. Our intuition is inspired by the physical interpretations of those traditional feedback control approaches.

Similar ideas of using both nonlinear and linear networks have been explored in the vision domain, e.g. ResNet (He et al., 2016), adding linear transformations in perceptrons (Raiko et al., 2012), and Highway networks (Srivastava et al., 2015). Our architecture resembles those and probably shares the benefits of easing signal passing and optimization. In addition, we experimentally show the learned sub-modules of our architecture are functional control policies and are robust against both state and action noise.

3. Background

We formulate the problem in a standard reinforcement learning setting, where an agent interacts with an infinite-horizon, discounted Markov Decision Process $(\mathbb{O}, \mathbb{S}, \mathbb{A}, \gamma, P, r)$. $\mathbb{S} \subseteq \mathbb{R}^n$, $\mathbb{A} \subseteq \mathbb{R}^m$. At time t, the agent chooses an action $a_t \in \mathbb{A}$ according to its policy $\pi_\theta(a_t|s_t)$ given its current observation $o_t \in \mathbb{O}$ or state $s_t \in \mathbb{S}$, where policy network $\pi_\theta(a_t|s_t)$ is parameterized by θ. For problems with visual input as observation o_t, state s_t is considered to be the encoded state after visual processing using a ConvNet, i.e. $s_t = \mu(o_t)$. The environment returns a reward $r(s_t, a_t)$ and transitions to the next state s_{t+1} according to the transition probability $P(s_{t+1}|s_t, a_t)$. The goal of the agent is to maximize the expected γ-discounted cumulative return: $J(\theta) = E_{\pi_\theta}[R_t] = E_{\pi_\theta}[\sum_{i=0}^{\infty} \gamma^i r(s_{t+i}, a_{t+i})]$, with respect to the policy network parameters θ.

4. Structured Control Network Architecture

In this section, we develop an architecture for policy network $\pi_\theta(a_t|s_t)$ by introducing control-specific priors in its structure. The proposed Structured Control Net (SCN) splits the generic multilayer perceptron (MLP) into two separate streams: a nonlinear control and a linear control. The two streams combine additively into the final action. Specifi-
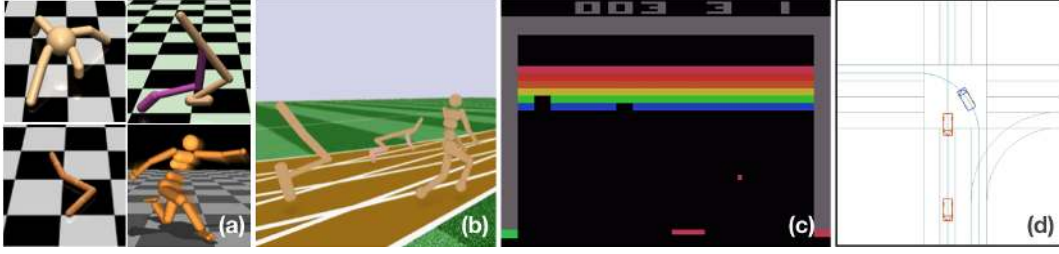
Figure 2. Various environments: (a) MuJoCo, (b) Roboschool, (c) Atari games, (d) Urban driving environments

cally, the resulting action at time t is

$$a_t = u_t^n + u_t^l, \qquad (1)$$

where u_t^n is a nonlinear control module, and u_t^l is a linear control module. Intuitively, the nonlinear control is for forward-looking and global control, while the linear control stabilizes the local dynamics around the residual of global control. The proposed network architecture is shown in Figure 1.

This decoupling of control modules is inspired by traditional nonlinear control theory (Khalil, 1996). To illustrate, let us consider the example of the trajectory tracking problem. In this setting, we typically have the 'desired' optimal trajectory of the state, denoted as s_t^d, provided by the planning module. The goal of the control is to track the desired trajectory as close as possible, i.e. the tracking error, $e_t = s_t - s_t^d$, should be small. To this end, in the nonlinear control setting, the action is given by:

$$a_t = u_t^s + u_t^e = u_t^s(s_t, s_t^d) + K \cdot (s_t - s_t^d), \qquad (2)$$

where $u_t^s(s_t, s_t^d)$ is a nonlinear control term, defined as a function of s_t and s_t^d, while $u_t^e = K \cdot (s_t - s_t^d)$ is a linear control term, with K being the linear control gain matrix for the tracking error e_t. Control theory tells us that the nonlinear term, u_t^s, is for global feedback control and also feed-forward compensation based on the predicted system behavior, s_t^d, while the linear control, u_t^e, is for reactively maintaining the local stability of the residual dynamics of e_t. At the first glance, Eq. 2 looks different from Eq. 1. However, if we apply the following transformation, $u_t^n(s_t, s_t^d) = u_t^s(s_t, s_t^d) - K \cdot s_t^d$, we obtain:

$$\begin{aligned} a_t &= u_t^s(s_t, s_t^d) + K \cdot (s_t - s_t^d) \\ &= u_t^s(s_t, s_t^d) + K \cdot s_t - K \cdot s_t^d = u_t^n(s_t, s_t^d) + K \cdot s_t, \end{aligned} \qquad (3)$$

where $u_t^n(s_t, s_t^d)$ can be viewed as the lumped nonlinear control term, and $u_t^l = K \cdot s_t$ is the corresponding linear control. This formulation shows that the solution for the tracking problem can be converted into the same form as SCN, providing insights into the intuition behind SCN. Moreover, learning the nonlinear module $u_t^n(s_t, s_t^d)$ would imply learning the desired trajectory, s_t^d, (planning) implicitly.
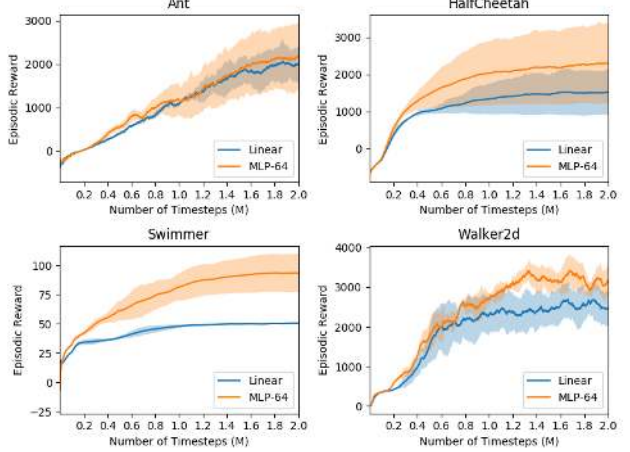

Figure 3. Example learning curves of MuJoCo environments using Linear policies vs. MLP policies averaged across three trials.

For the linear control module of SCN, the linear term is $u_t^l = K \cdot s_t + b$, where K is the linear control gain matrix and b is the bias term, both of which are learned. To further motivate the use of linear control for DRL, we empirically observed that a simple linear policy can perform reasonably well on some of the MuJoCo tasks, as shown in Figure 3.

For the nonlinear control module of SCN, we use a standard fully-connected MLP, but remove the last bias term from the output layer, as the bias is provided by the linear control module. In the next section, we show the size of our nonlinear module can be much smaller than standard Deep RL MLP networks due to our split architecture. We also show that both control modules of the architecture are essential to improving model performance and robustness.

5. Experimental Setup

We design and conduct all of our experiments following the guidelines introduced by the recent study on DRL reproduciblity (Henderson et al., 2017). To demonstrate the general applicability and effectiveness of our approach, we experiment across different training algorithms and a diverse set of environments.
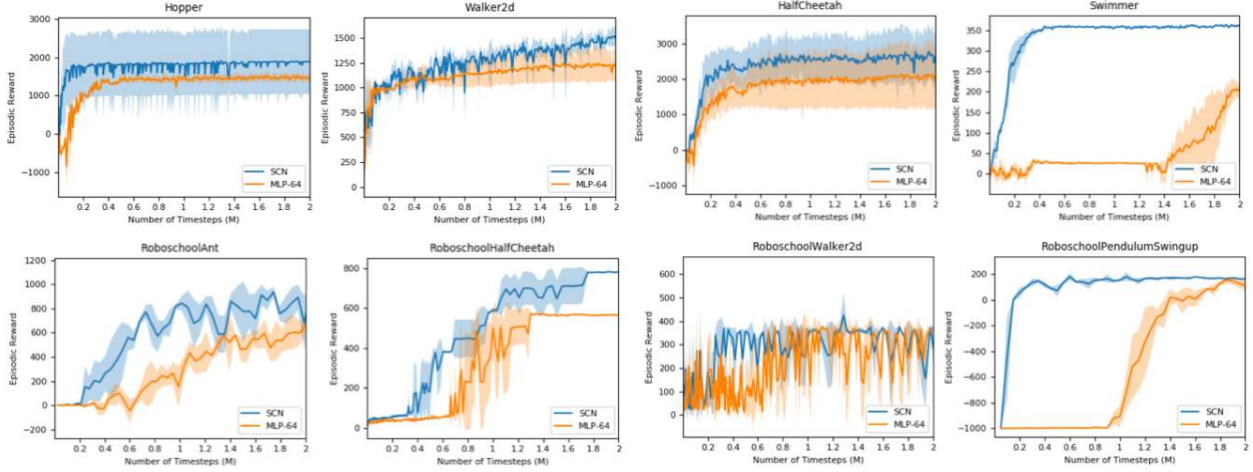
Figure 4. Learning curves of SCN-16 in blue, and baseline MLP-64 in orange, for ES on MuJoCo and Roboschool environments.
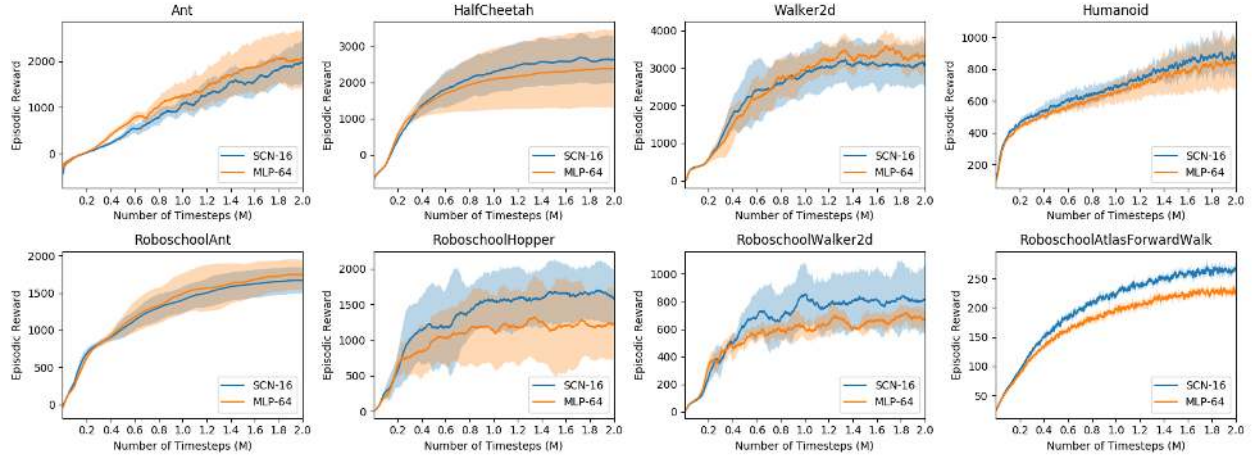

Figure 5. Learning curves of SCN-16 (blue), and baseline MLP-64 (orange), for PPO on MuJoCo and Roboschool environments.

5.1. Environments

We conduct experiments on several benchmarks, shown in Figure 2, including OpenAI MuJoCo v1 (Todorov et al., 2012), OpenAI Roboschool v1 (OpenAI, 2017), and Atari Games (Bellemare et al., 2013). OpenAI Roboschool has several environments similar to those of MuJoCo, but the physics engine and model parameters differ (OpenAI, 2017). In addition, we test our method on a custom urban driving simulation that requires precise control and driving negotiations, e.g. yielding and merging in dense traffic.

5.2. Training Methods

We train the proposed SCN using several state-of-the-art training methods, i.e., Evolutionary Strategies (ES) (Salimans et al., 2017), PPO (Schulman et al., 2017), and ACKTR (Wu et al., 2017).

For our ES implementation, we use an efficient shared-memory implementation on a single machine with 48 cores. We set the noise standard deviation and learning rate as 0.1

and 0.01, respectively, and the number of workers to 30. For PPO and ACKTR, we use the same hyper-parameters and algorithm implementation from OpenAI Baselines (Dhariwal et al., 2017).

We avoid any environment specific hyper-parameter tuning and fixed all hyper-parameters throughout the training session and across experiments. Favoring random seeds has been shown to introduce large bias on the model performance (Henderson et al., 2017). For fair comparisons, we also avoid any specific random seed selections. As a result, we only varied network architectures based on the experimental need.

6. Results

Our primary goal is to empirically investigate if the proposed architecture can bring together the benefits of both linear and nonlinear policies in terms of improving training sampling efficiency, final episodic reward, and generalization of learned policy, while using a smaller network.
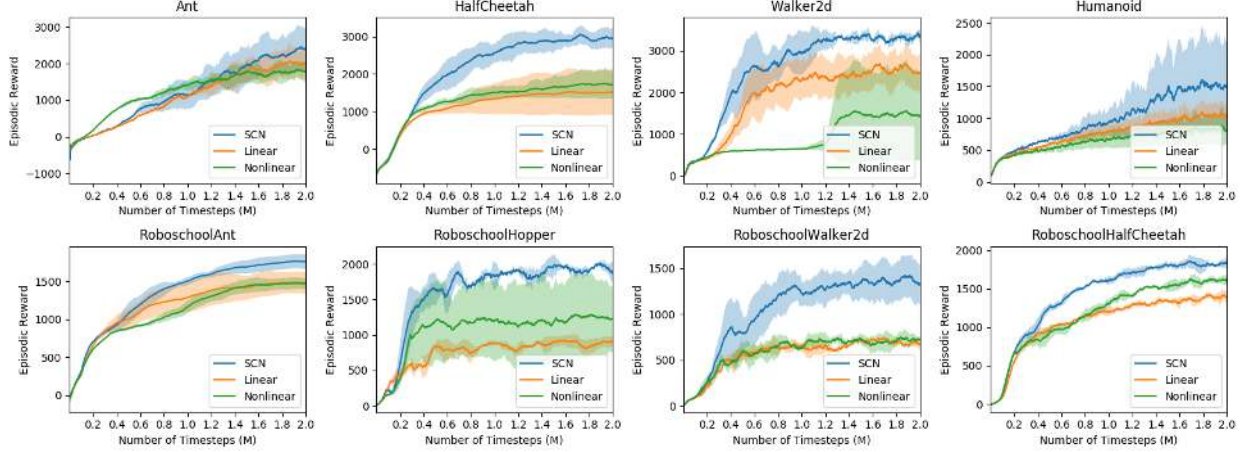
Figure 6. Ablation study on training performance: SCN in blue, Linear policy in orange, and MLP policy in green (same size as nonlinear module of SCN), trained with PPO on MuJoCo and Roboschool tasks.

Following the general experimental setup, we conducted seven sets of experiments:

(1) **Performance of SCN vs. baseline MLP**: we compare our SCN with the baseline MLP architecture (MLP-64) in terms of sampling efficiency, final episodic reward, and network size.

(2) **Generalization and Robustness**: we compare our SCN with the baseline MLP architecture (MLP-64) in terms of robustness and generalization by injecting action and observation noise at test time.

(3) **Ablation Study of the SCN Performance**: we show how performance of the Linear policy and Nonlinear MLP, used inside of the SCN, compares.

(4) **Ablation Study of Learned Structures**: we test if SCN has effectively learned functioning linear and nonlinear modules by testing each learned modules in isolation.

(5) **Performance of Environment-specific SCN vs. MLP**: we compare the best SCN and the best MLP architecture for each environment.

(6) **Vehicle Driving Domain**: we test the effectiveness of SCN on solving driving negotiation problems from the urban self-driving domain.

(7) **Atari Domain**: we show the ability of SCN to effectively solve Atari environments.

6.1. Performance of SCN vs. baseline MLP

The baseline MLP architecture (MLP-64) used by most previous algorithms (Schulman et al., 2017; Wu et al., 2017), is a fully-connected MLP with two hidden layers, each consisting of 64 units, and tanh nonlinearities. For each action dimension, the network outputs the mean of a Gaussian distribution, with variable standard deviation. For PPO and

ACKTR, the nonlinear module of the SCN is an MLP with two hidden layers, each containing 16 units. For ES, the nonlinear module of SCN is an MLP with a single hidden layer, containing 16 units, with the SCN outputting the actions directly due to the inherent stochasticity of the parameter space exploration.

For each experiment, we trained each network for 2M timesteps and averaged over 5 training runs with random seeds from 1 to 5 to obtain each learning curve. The training results of ES and PPO for MuJoCo/Roboschool are shown in Figure 4 and Figure 5 respectively. The ACKTR plots are not shown here due to their similarity to PPO. From the results, we can see that the proposed SCN generally performs on par or better compared to baseline MLP-64, in terms of training sampling efficiency and final episodic reward, while using only a fraction of the weights. As an example, for ES, the size of SCN-16 is only 15.6% of the size of baseline MLP-64 averaged across 6 MuJoCo environments.

We calculated the training performance improvement to be the percentage improvement for average episodic reward. The average episodic reward is calculated over all 2M timesteps of the corresponding learning curve. This metric indicates the sampling efficiency of training, i.e. how quickly the training progresses in addition to the achieved final reward. Even with the same hidden-layer size, across all the environments, SCN-16 achieved an averaged improvement of 13.2% for PPO and 17.5% for ES, compared to baseline MLP-64.

6.2. Generalization and Robustness

We next tested generalization and robustness of the SCN policy by injecting Gaussian noise in action and observation spaces at test time, and then comparing with the baseline MLP. We injected varying levels of noise by adjusting the

Task	Final SCN	Final MLP	Average SCN	Average MLP
HalfCheetah	3310	**3386**	**2148**	1693
Hopper	2479	**2619**	1936	**1977**
Humanoid	**2222**	1078	**979**	674
Walker2d	**3761**	3520	**2404**	2349
Swimmer	81	**109**	58	**74**
Ant	**2977**	2948	**1195**	1155
Roboschool HalfCheetah	1901	**2259**	1419	**1693**
Roboschool Hopper	**2027**	1411	**1608**	914
Roboschool Humanoid	**187**	175	**131**	115
Roboschool Walker2d	**1547**	774	**1048**	584
Roboschool Ant	1968	**2018**	**1481**	1394
Roboschool AtlasForward-Walk	**273**	236	**202**	176

Table 1. Results of final episodic reward and averaged episodic reward for best SCN vs. best MLP per environment.

standard deviation of a normal distribution. The episodic reward for each level of noise is averaged over 10 episodes. Figure 7 shows the superior robustness of the proposed SCN against noise unseen during training compared to the baseline MLP-64.

6.3. Ablation Study of the SCN Performance

To demonstrate the synergy between the linear and the nonlinear control modules for SCN architecture, we trained the different sub-modules of SCN separately and conducted ablation comparison, i.e. linear policy and nonlinear MLP policy with the same size as the nonlinear module of SCN. The results on the MuJoCo and OpenAI Roboschool with PPO are shown in Figure 6. We make following observations: (1) the linear policy alone can be trained to solve the task to a reasonable degree despite its simplicity; (2) SCN outperforms both the linear and nonlinear MLP policy alone by a large margin for most of the environments.

6.4. Ablation Study of Learned Structures

To investigate whether SCN has learned effective linear and nonlinear modules, we conducted three tests after training the model for 2M timesteps. Here, we denote the linear module inside SCN as SCN-L and nonlinear module inside SCN as SCN-N to distinguish from the separately trained Linear policy and ablation MLP policy. We run SCN-L or SCN-N by simply not using the other stream.

The first test compares the performance of the separately trained Linear policy, with the linear control module learned inside SCN (SCN-L). In simpler environments, where a Linear policy performs well, SCN-L performs similarly. Thus SCN appears to have learned an effective linear policy.

However, in more complex environments, like Humanoid, SCN-L does not perform well on its own emphasizing the fact that the nonlinear module is very important. Across MuJoCo environments, SCN-L is able to achieve 68% of the performance of the stand alone Linear policy when trained with ES, and 65% with PPO. Hence for most environments, the linear control of SCN is effectively learned and functional.

The second test compares the performance of the MLP versus the nonlinear module inside SCN (SCN-N). Unlike the linear module test, the performance of the two identical MLPs are drastically different. Across all environments, SCN-N is not able to perform well without the addition of the linear module. We found that SCN-N is only able to achieve about 9% of the performance of the stand-alone MLP when trained with ES, and 8% with PPO. These tests verify the hypothesis that the linear and nonlinear modules learn very different behaviors when trained in unison as SCN and rely on the synergy between each other to have good overall performance.

The third test compares the performance of SCN versus a pseudo SCN, which is assembled post-training by a pretrained MLP and a pre-trained Linear policy. For tested environments, the naive combination of the two already-trained policies does not perform well. By combining the separate MLP and linear model, we were able to achieve only 18% of the performance of SCN when using ES, and 21% with PPO. This demonstrates the importance of training both components of the structure in the same network, like SCN.

6.5. Performance of Environment-specific SCN

In general, different environments have different complexities. To study the SCN for the most efficient size for each environment, we sweep the hidden-layer size of the nonlinear module across the set of model sizes 64, 32, 16, 8,
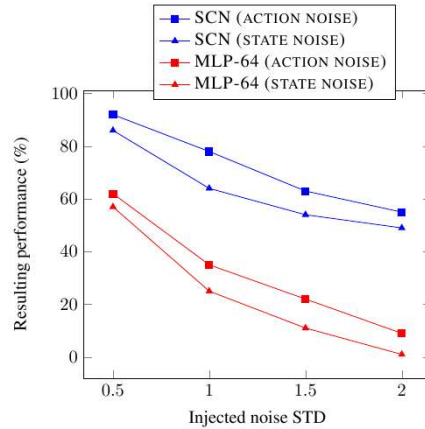

Figure 7. Performance degradation of SCN compared to the baseline MLP-64 when varying levels of noise are injected into the action and state space.
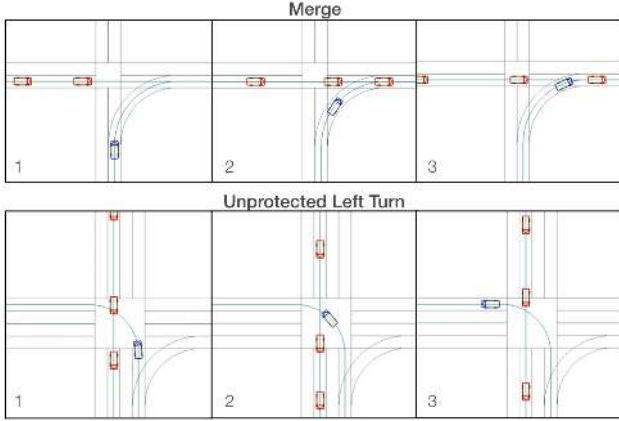
Figure 8. Two sequences of 3 frames showing learned agent (blue) via SCN, making a merge (top 3) and an unprotected left turn (bottom 3) with oncoming traffic (red).

Urban Driving Results

Task	ES SCN/MLP	PPO SCN/MLP
UnprotectedLeftTurn	93/76	138/102
Merge	95/81	101/88

Table 2. Final episodic rewards on the driving scenarios using the SCN vs. MLP with ES and PPO.

and 4. As a comparison, we also sweep the hidden-layer size of the baseline MLP to get the best MLP size for each environment for the same size set. We keep the number of hidden layers fixed at two.

For each environment, we compared the environment-wise best SCN and best MLP from the model size set. We calculated the average episodic reward as episodic rewards averaged over the whole 2M timesteps of the corresponding learning curve. This metric indicates the sampling efficiency of training, i.e. how quickly the training progresses. Final episodic reward is the averaged rewards of the last 100 episodes. We illustrate the results with data trained with PPO. From Table 1, we can see SCN shows equal or better performance, compared to the environment-wise best performing MLP.

6.6. Vehicle Driving Domain

We next validate the effectiveness of SCN on solving negotiation problems in the urban self-driving domain. Sequential decisions in dense traffic are difficult for human drivers. We picked two difficult driving scenarios: completing an unprotected left turn and learning to merge in dense traffic.

For the simulation, we used a bicycle model as the vehicle dynamics (Thrun et al., 2006). Simulation updates at 10Hz. The other traffic agents are driven by an intelligent driver model with capabilities of adaptive cruise control and lane keeping. Both learned agents and other traffic agents are initialized randomly within a region and a range of starting

Atari Results

Metric	SCN	MLP
Average Reward	**34**	26
Final Reward	**35**	25

Table 3. Number of Atari games won by SCN-8 vs. MLP-512 when trained with PPO.

speeds. The other traffic agents have noise injected into their distance keeping and actions for realism. The state observed by the agent consists of ego vehicle state, states of other traffic agents, and the track on which it is traveling (e.g. center lane). The reward is defined to be -200 for a crash, 200 for reaching the goal, small penalties for being too close to other agents, small penalties for going too slow or too fast, and small incentives for making progress on the track. An episode reward larger than 50 is considered to be solved. A Stanley steering controller (Thrun et al., 2006) is used to steer the vehicle to follow the track. The action space for the learned agent is a continuous acceleration value.

Table 2 shows the final episodic reward achieved, comparing SCN and MLP-64, trained with ES and PPO. In Figure 8, we visualize the learned SCN policies controlling the learned agent (blue) while successfully making a merge and an unprotected left turn through a traffic opening.

6.7. Atari Domain

In this section, we show that SCN is able to learn effective policies in Atari environments. The SCN with visual inputs uses the same convolutional layers and critic model as (Schulman et al., 2017), but the learned visual features are flattened and fed into the linear and nonlinear modules of SCN. The baseline Atari policy from PPO (Schulman et al., 2017) is a fully-connected MLP with one hidden layer containing 512 units, chosen by cross-validation, on the flattened visual features, which we denote as MLP-512. The nonlinear module of SCN is a MLP with one hidden layer that has 8 units (SCN-8).

Each learning curve and metric is averaged across three 10M-timestep trials (random seeds: 0,1,2). Learning curves for all 60 games are provided in the Appendix. We summarize the results in Table 3, where we show that the SCN-8 can perform competitively in comparison to Atari baseline policy, MLP-512 (much larger in size), across 60 Atari games. If the metric is similar, we consider SCN wins since it is smaller in size. Figure 9 displays learning curves for 6 randomly chosen games. We can see that SCN-8, achieves equal or better learning performance compared to the baseline policy, MLP-512, and the ablation policy, MLP-8. We further observed that even with 4 hidden units, SCN-4, which is smaller in size than SCN-8, performs similarly well on many games tested.
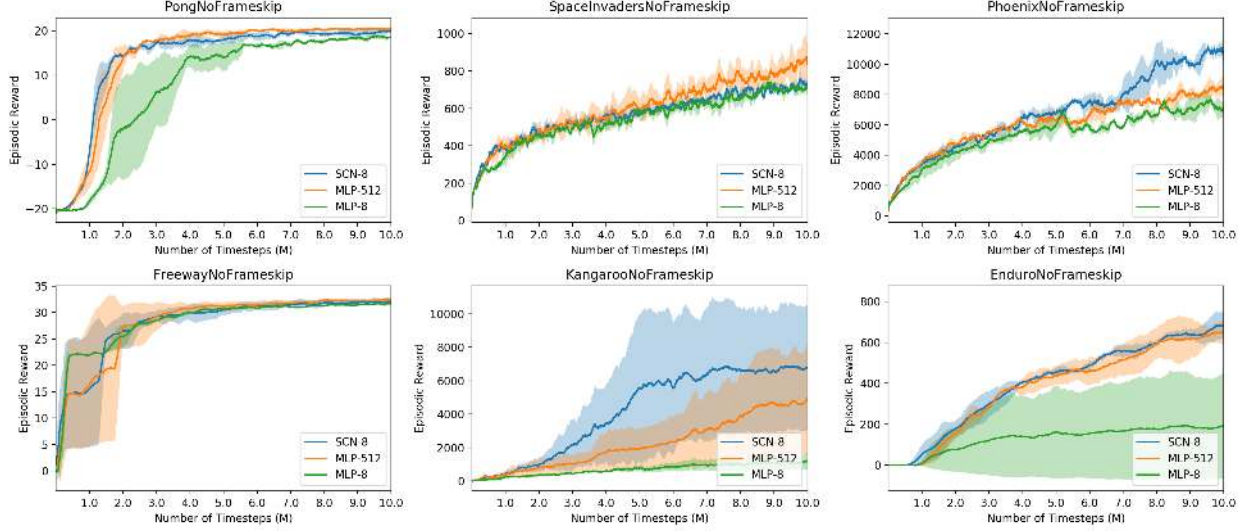
Figure 9. Atari environments: SCN-8 in blue, baseline Atari MLP (MLP-512) in orange, and ablation MLP (MLP-8) in green. 10M timesteps equal 40M frames.
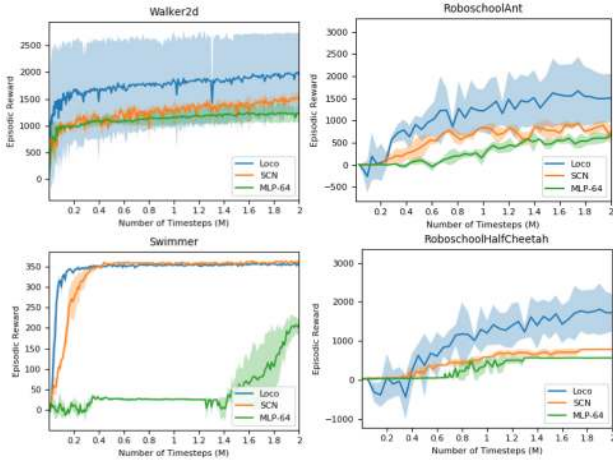

Figure 10. Locomotive tasks from MuJoCo and Roboschool: locomotor net (Loco) in blue, SCN in orange, and baseline MLP (MLP-64) in green. Results achieved using ES.

7. Case Study: Locomotion-specific SCN

In our final set of experiments, we use dynamic legged locomotion as a case study to demonstrate how to tailor SCN to specific tasks using the task-specific priors.

In nature, neural controllers for locomotion have specific structures, termed central pattern generators (CPGs), which are neural circuits capable of producing coordinated rhythmic patterns (Ijspeert, 2008). While the rhythmic motions are typically difficult to learn with general feedforward networks, by emulating biological CPGs using Fourier series and training the Fourier coefficients, we are able to show the power of adding this inductive bias when learning cyclic movements in the locomotive control domain. The nonlinear module of SCN becomes

$$u_t^n = \sum_{i=1}^{c} A_i sin(\omega_i t + \phi_i), \quad (4)$$

where for each action dimension, A_i, ω_i, ϕ_i are the amplitude, frequency and phase, respectively, of the component i, that will be learned, and c is set to 16 sinusoids. In our experimental results, we find that the linear module of SCN is necessary for achieving better performance, by stabilizing the system around the residual of CPGs outputs. We name this specific instantiation of SCN "Locomotor Net". By replacing the MLP in the nonlinear module of SCN with a locomotive-specific implementation, we were able to further improve sampling efficiency and the final reward on those environments. Example results on locomotive tasks from MuJoCo and Roboschool are shown in Figure 10.

8. Conclusion

In this paper we developed a novel policy network architecture that is simple, yet effective. The proposed Structured Control Net (SCN) has the benefits of both linear and nonlinear control: improving training sampling efficiency, final episodic reward, and generalization of learned policy, in addition to using a smaller network and being general and applicable to different training methods. By incorporating problem specific priors into the architecture, the proposed architecture has the potential to improve upon broader control tasks. Our case study demonstrated further improved performance for locomotion tasks, by emulating the biological central pattern generators (CPGs) as the nonlinear part of the architecture. For future work, we plan to extend the SCN to incorporate a planning module. The planning module will be responsible for long-term planning and high-level abstracted decision making.

References

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279, 2013.

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

Camacho, E. F. and Alba, C. B. *Model predictive control*. Springer Science & Business Media, 2013.

Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K. O., and Clune, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *arXiv preprint arXiv:1712.06560*, 2017.

Dhariwal, P., Hesse, C., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Openai baselines. https://github.com/openai/baselines, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Ijspeert, A. J. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

Khalil, H. K. Noninear systems. *Prentice-Hall, New Jersey*, 2(5):5–1, 1996.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Mania, H., Guy, A., and Recht, B. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Oh, J., Singh, S., and Lee, H. Value prediction network. In *Advances in Neural Information Processing Systems*, pp. 6120–6130, 2017.

OpenAI. Roboschool. https://github.com/openai/roboschool, 2017.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Raiko, T., Valpola, H., and LeCun, Y. Deep learning made easier by linear transformations in perceptrons. In *Artificial Intelligence and Statistics*, pp. 924–932, 2012.

Rajeswaran, A., Lowrey, K., Todorov, E. V., and Kakade, S. M. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pp. 6553–6564, 2017.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shalev-Shwartz, S., Ben-Zrihem, N., Cohen, A., and Shashua, A. Long-term planning by short-term prediction. *arXiv preprint arXiv:1602.01580*, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162, 2016.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9): 661–692, 2006.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Tu, S. and Recht, B. Least-squares temporal difference learning for the linear quadratic regulator. *arXiv preprint arXiv:1712.08642*, 2017.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1995–2003, 2016.

Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pp. 5285–5294, 2017.