Learning Dynamic Networks

Kale-ab Tessera, Chiratidzo Matowe, Arnu Pretorius, Benjamin Rosman, Sara Hooker

Problem Statement

Overparameterized models have led to many **breakthroughs** in machine learning.

Challenges:
- Larger Models:
  - Efficient **storage** and inference.
  - **Efficient training** of large models.
  - **Overfitting**, regularization and generalization.
- Train Longer:
  - Handle **temporal** dynamics of training.

Overparameterization - Sparsity/Pruning

Common method to handle challenges of overparatermization - **Pruning**.

Benefits - similar performance, with a **fraction of the weights** , **faster training** and more **robust** to noise.

Our Approach - Can we learn these schedules using RL

Algorithm 1 Learning Sparsity Schedules using Reinforcement Learning

Input: train dataset X_{train_set}, test dataset X_{test_set}, train network f_{train}, eval network f_{eval}, agent a, number of episodes N, minimum density per layer min_d and maximum density per layer max_d.

$a \leftarrow init(min_d, max_d)$ ▷ Initialize agent a.

$X_{train\_split}, X_{val\_split} \leftarrow split(X_{train\_set})$ ▷ Split train dataset.

for episode=1,N **do**

  $a \leftarrow train\_loop(a, X_{train\_split}, X_{val\_split}, f_{train})$ ▷ Run train loop and retrieve trained agent a.

  eval_loop$(a, X_{train_set}, X_{val_set}, f_{eval})$ ▷ Run evaluation loop on unseen network f_{eval} using trained agent a.

end for

- Agent - PPO.
- Dataset - Cifar10.
- Sparsity:
  - Random Pruning, with Random Regrowth (**RP-RR**)
  - Magnitude Pruning, with Random Regrowth (**MP-RR**)

Train Longer - Schedules

When we train longer -> more **critical temporal decisions** to make.

Temporal Decisions (examples include):
- Learning Rate - Initial Learning Rat & **LR Schedule**.
- Sparsity - Initial Sparsity & **Sparsity Schedule**.

What about these choices per layer?! - **Layerwise Schedules.**
Standard approach - choose these schedules through **trial-and-error**.

Question

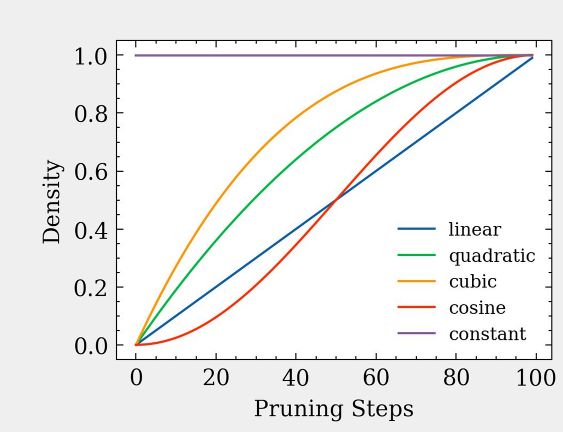
Can we learn **temporal** (possibly **layerwise**) **schedules** in a principled manner?

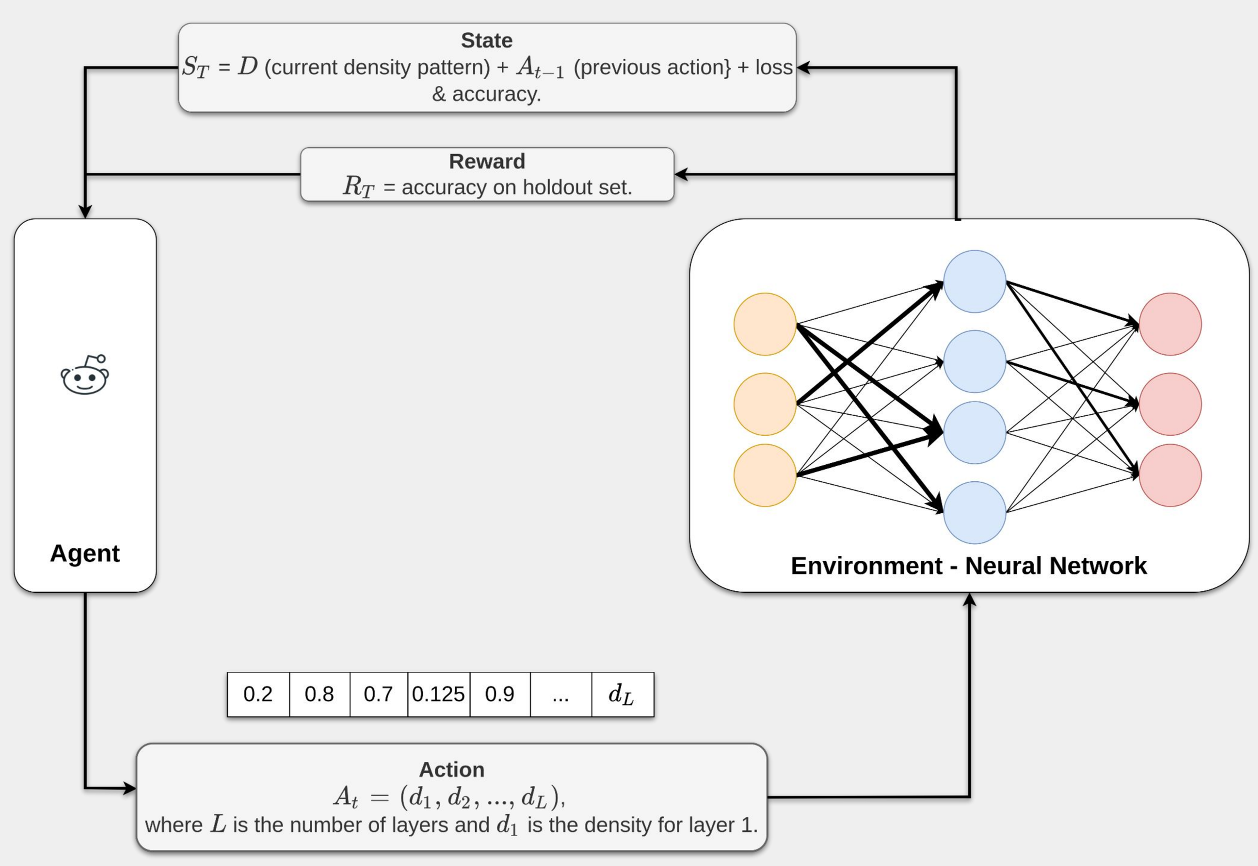
Related Work

Handcrafted schedules.

- ***Constant*** - SET [1], Deep Rewiring (DeepR) [2] and Neural Network Synthesis Tool (NEST) [3].
- ***Cosine*** - RigL [4] and Sparse Network From Scratch (SNFS) [5]
- ***Cubic*** - [6],[7].

Examples of Simple Schedules:





Results

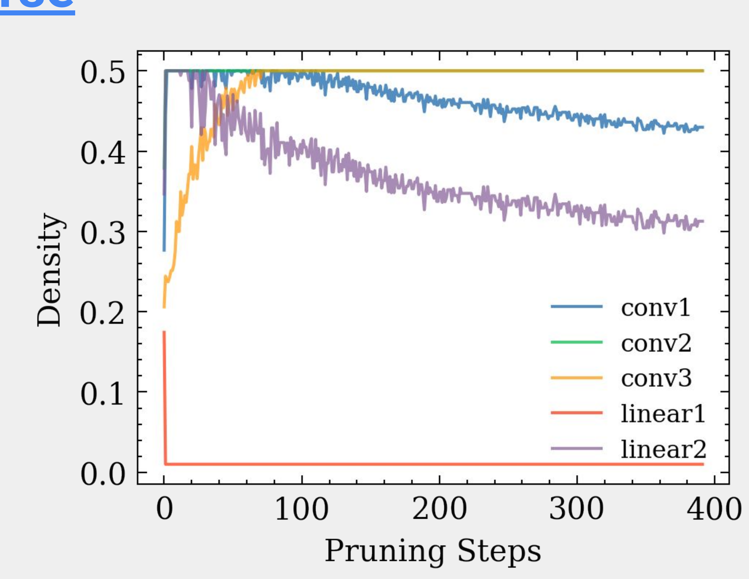
Simple CNN - 5 Layers

1. Learned Schedules are Competitive

Table 1: Test Accuracy (mean and standard deviation) of different schedules on CIFAR-10, using Simple-CNN.

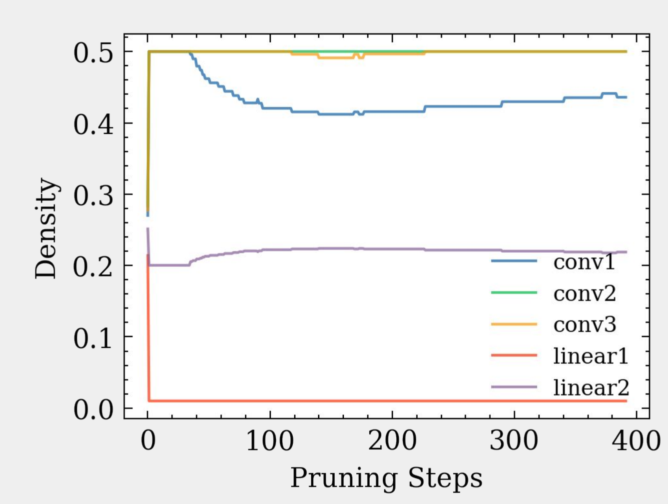
Target Density (%)	Schedule	Random Pruning with Random Regrowth (RP-RR)	Magnitude Pruning with Random Regrowth (MP-RR)
10	Linear	20.365 +- 17.952	60.418 +- 1.362
	Quadratic	23.15 +- 20.043	61.259 +- 1.485
	Cubic	33.721 +- 21.693	60.396 +- 0.832
	Cosine	18.302 +- 14.379	59.807 +- 0.384
	Constant	**61.475 +- 0.731**	62.536 +- 0.314
	Learned (Ours)	61.071 +- 1.574	**63.191 +- 0.810**
50	Linear	64.54 +- 0.477	64.78 +- 0.464
	Quadratic	64.987 +- 0.86	63.933 +- 0.431
	Cubic	65.31 +- 0.49	64.315 +- 0.437
	Cosine	64.672 +- 0.771	64.737 +- 0.345
	Constant	65.1 +- 0.283	65.388 +- 0.375
	Learned (Ours)	**65.655 +- 0.515**	**65.686 +- 0.284**
100	Linear	66.228 +- 0.691	66.711 +- 0.423
	Quadratic	66.947 +- 0.749	67.25 +- 0.578
	Cubic	66.857 +- 0.627	67.395 +- 0.547
	Cosine	66.074 +- 0.282	66.18 +- 1.027
	Full Dense	**67.815 +- 0.146**	67.878 +- 0.482
	Learned (Ours)	67.534 +- 0.174	**67.908 +- 0.162**

2. Learned Schedules are Layerwise Diverse



3. Learned a Handcrafted Schedule!

Piecewise Schedule for Random Pruning- [8].



ResNet-18

Schedule	Test Accuracy
Linear	93.019 +- 0.024
Quadratic	93.106 +- 0.107
Cubic	**93.148 +- 0.156**
Cosine	92.916 +- 0.105
Constant (Fully Dense)	92.481 +- 0.641
Learned (Ours)	92.818 +- 0.048

Challenges:
1. **Non-stationarity** environment.
   a. Our environment (the network we are learning a schedule for) is learning and adapting while our agent is learning to model the environment.
   b. Worse for challenging networks - use techniques like **data augmentation** and **learning rate decay** (e.g. ResNet-18).
2. **High dimension** action and (possibly) state space.
3. **Slow convergence** - 25-50 episodes.

Conclusion:

In this work, we demonstrate that it is **possible** to learn **well performing dynamic sparsity schedules** using reinforcement learning. The schedules learned are not arbitrary and are distinct per layer and pruning method.

References

1. Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nature communications, 9 (1):1–12, 2018.
2. Bellec, G., Kappel, D., Maass, W., and Legenstein, R. Deep rewiring: Training very sparse deep networks. arXiv preprint arXiv:1711.05136, 2017.
3. Dai, X., Yin, H., and Jha, N. K. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. IEEE Transactions on Computers, 68(10):1487–1497, 2019.
4. Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In International Conference on Machine Learning, pp. 2943– 2952. PMLR, 2020.
5. Dettmers, T. and Zettlemoyer, L. Sparse networks from scratch: Faster training without losing performance. arXiv preprint arXiv:1907.04840, 2019.
6. Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878, 2017.
7. Mostafa, H. and Wang, X. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In International Conference on Machine Learning, pp. 4646–4655. PMLR, 2019.
8. Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks, 2019. URL https://arxiv. org/abs/1902.09574.

ICML Workshop Paper- Workshop on Dynamic Neural Networks.