CS230

Evolutionary strategies for finding sparse randomly weighted subnetworks

Alexander Conklin
Department of Applied Physics
Stanford University
*conklin1@stanford.edu

Abstract

Recent trends in deep learning suggest backpropagation based training techniques scale poorly and will suffer from computational barriers in the near future. Separately, on the inference side, a growing body of literature has demonstrated neural networks can be made sparse without losing accuracy. Sparse models can be represented more compactly and accelerate computation. In this project we revisit evolutionary learning as a method to find high performing sparse subnetworks in randomly initialized neural networks. We show these sparse neural networks can be found for a subset of control/locomotive problems from OpenAI's gym with a simple evolutionary algorithm that is highly parallelizable.

1 Introduction

In the modern deep learning era, the use of GPUs along with hyperscaling computing has enabled training of massive neural networks. Some of the world's largest models, such as GPT-3, exceed 175 billion parameters and cost an estimated 12 million USD in compute resources to train. While GPT-3 may be an outlier in NLP models, it is reminiscent of a broader trend in neural networks - larger networks and parameter counts win out. Recent studies have shown that across most areas of deep learning, performance and training is more dependent on computational resources than architectural changes [7]. A priori, this may not be particularly concerning given the consistent computational improvements we've seen over the last 50 years due to Moore's law or the observation that transistor density (and therefore compute) roughly doubles every two years due to manufacturing improvements. However, a McKinsey study found the compute required to train the largest AI models was doubling every 3.4 months, far outpacing the two year doubling period of Moore's law. These observations motivate us to examine alternative training paradigms that are less compute intensive, can create compact parameter representations, and offer increased parallelizability. This last point is important, backpropagation methods have poor resource utilization when the workload is split over multiple GPUs [6].

In this work we examine evolutionary algorithms as a method to find sparse subnetworks for locomotive learning tasks. Instead of calculating gradients, evolutionary strategies (ES) learn neural network weights by comparing the fitness of different initialized parameter sets, comparing them and then recombining them according to a user defined heuristic. As a result, their "training" is composed of inference-heavy operations. In particular we focus on locomotive tasks from the OpenAI gym, where our inputs are states in a specified locomotive environments. Although small, these are tractable examples of an important class of control problems which robots deployed in the real world may encounter under constrained RAM and processing power. Sparse representations, when paired with proper hardware, can drastically reduce computation overhead and may hold the key to enabling

CS230: Deep Learning, Spring 2021,

greater robotic operation on the edge. Instead of training the weights of a network, we use a modified evolutionary algorithm to train a mask which when applied over the weights of a randomly initialized neural network identifies an appropriate subnetwork that is the correct policy for the task at hand.

2 Related works

2.1 Sparsity in neural networks

In recent years sparse representations of neural networks have received renewed interest. Seminal works by Yan LeCun in the 1990s showed that pruning (i.e. removing weights) of weights in already trained neural networks can yield up to 90 percent more compact representations without harming accuracy[3]. While diverse methodologies for pruning already trained neural network have emerged, finding sparse representations during training has shown to be challenging. An new approach has been to search large randomly weighted neural networks. In the lottery ticket hypothesis, Fankle and Carbin postulate that randomly weighted dense neural networks contained winning tickets, or sparse subnetworks, which can be trained to high accuracy[2]. They then demonstrate a gradient based method to find these winning tickets. Subsequent work has extended this hypothesis, suggesting that large randomly weighted neural networks contain sparse subnetworks that achieve high accuracy without requiring any retraining[4]. Current methods to find these networks focus on using "masking", or learning a mask (usually binary), which is applied on the random network (via element by element multiplication) to discern a hidden sub network[8]. Underlying all these methods is a form of explicit gradient calculations to identify the sub network.

2.2 Evolutionary strategies for reinforcement learning

Motivated by these works we look towards evolutionary learning as a scalable way to find these subnetworks. Evolutionary strategies are black-box optimization methods that iterate towards a solution by perturbing populations of parameters and evaluating their fitness. OpenAI demonstrated evolutionary strategies using stochastic gradient ascent were surprisingly effective in direct policy learning for robotic control tasks [5]. Due to the iterative observation-action process in reinforcement learning, it can be challenging to get a complete, continuous picture of the policy derivatives. Evolutionary strategies seem to offer more gadgets that can overcome the challenges of these non-smooth policy landscapes compared to their reinforcement learning counterparts [5]. Lastly, OpenAI showed that evolutionary strategies used 3x less compute than propagation based methods at the cost of increased memory compared to equivalent state-of-the-art backpropagation solutions for Atari.

3 Dataset and Features

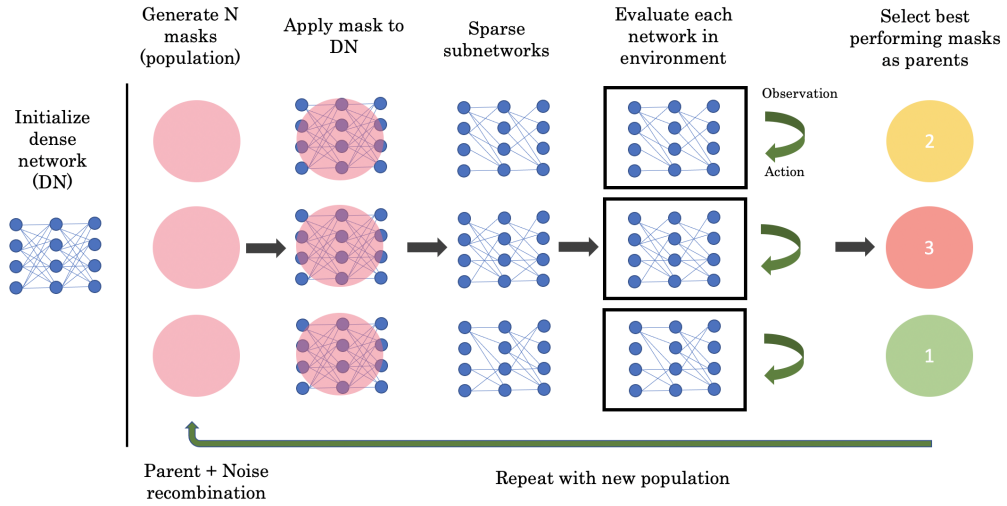
For robotic-like tasks we utilize OpenAI's gym environments, specifically from Box2D and ClassicalControl [1]. Since the nature of our project is to explore a new algorithmic strategy, these well developed environments provide a good test bed to compare against traditional reinforcement learning. As per OpenAI's intent, each environment comes with a threshold reward for which researchers can compare the iterations, steps, or time required to train a network that can achieve the threshold. Additionally, the environments vary in complexity allowing evaluation on reinforcement tasks requiring various parameter counts.

4 Methods

4.1 Algorithm Overview

We employ a modified evolutionary learning strategy that identifies sparse subnetworks through masks. These masks are best thought of as filters that are applied to a randomly initialized neural network to identify a network within the original network. Figure 1 depicts a visualization of the training process. First, a randomly initialized dense neural network is generated and saved - this network will remain unchanged throughout the training process. Next, a population of N individuals is generated. Each individual has a randomly generated "mask" which is a set of weights that mirrors the shape of the neural network.

Figure 1: Visualization subnetwork search and training process.



The mask is then turned into binary by a thresholding function which determines the sparsity enforced on the network. A sparse subnetwork is generated by applying an element by element product of the binary mask with the original dense network. The population of sparse networks is independently evaluated in random instance of the problem environment for a specified number of runs (3 - 10 depending on the environment) to obtain several different training rewards. These rewards are weighted and scored. The top K performing networks are chosen as "elite".

Subsequent iterations follow a similar procedure with one change. The populations ($IndividualMask_n$) are now generated by selecting a random "elite" member ($ParentMask_j$) and injecting gaussian noise parameterized by sigma:

$$IndividualMask_n = ParentMask_j + \sigma * \mathcal{N} \tag{1}$$

The relationship dictates our recombination strategy, allowing successful parameter sets to be passed from generation to generation. It is worth emphasizing this method directly learns the policy mapping.

4.2 Model

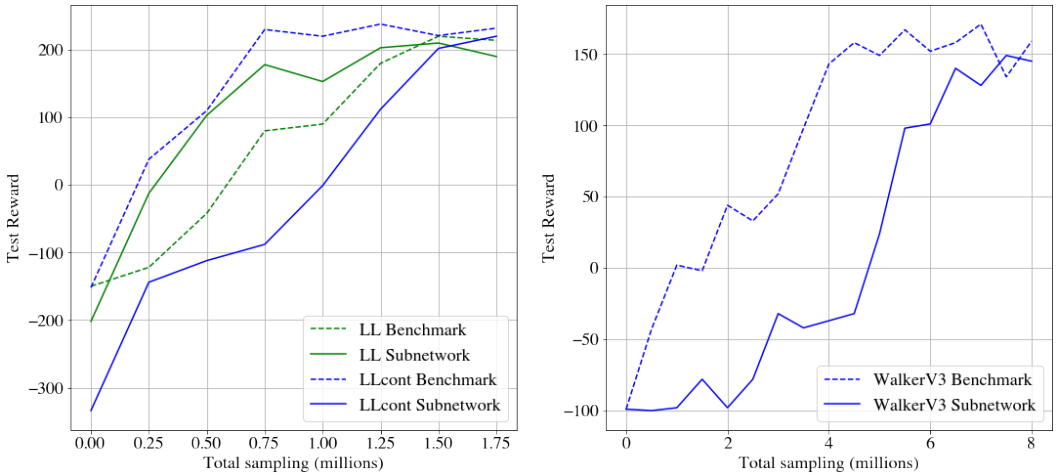
Few layer, fully connected neural networks with tanh activations are used as the initial networks. These small, simple networks were chosen deliberately to allow for easy diagnosis and faster experiment iterations. Total model parameters ranged from 5k - 100k weights.

5 Results/Discussion

Broadly our results demonstrate simple evolutionary algorithms applied to binary masks can effectively find sparse subnetworks. The primary metrics for evaluation used were the total number of steps taken in the environment to reach a target reward and the target reward itself. For evolutionary learning, step number corresponds to the total number of forward passes and provides an excellent capture of computational overhead for models of similar size. We will refer to our algorithm through out as "subnetwork".

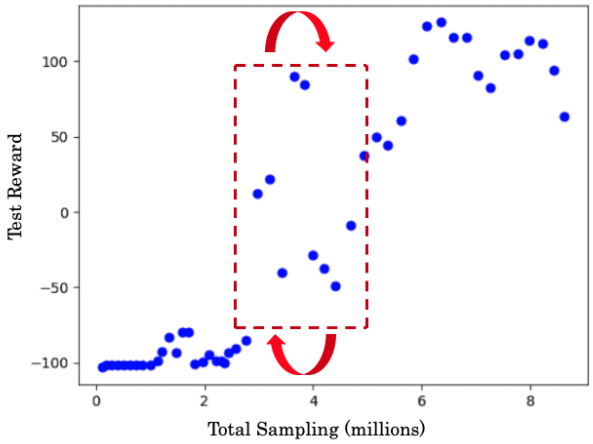
We calibrated our model against a traditional evolutionary strategy (weights could be changed) in order to get a baseline of hyperparameters. Figure 2 displays the weighted average of our algorithms test reward over 5 random seeds compared to a standard ES implementation. We found the best hyperparameters were population sizes of 32 - 64, selecting < 6 "elite" parents and keeping sigma between .01 and .2 . There a few features of note. First, on average our subnetwork algorithm only slightly lags the ES benchmark in samples required to achieve comparable rewards. However, on a run by run basis the subnetwork algorithm is highly sensitive to the initialization seed and exhibits

Figure 2: Test reward for OpenAI's LunarLander, LunarLanderContinuous and BipedalWalkerV3 tasks on our subnetwork model and an ES benchmark. Test reward is averaged over 5 random seeds. A reward above 200 for each of the tasks is considered solved.



a highly stochastic path towards the solution. Figure 3 isolates a single training run for the sparse subnetwork algorithm in the BipedalWalkerV3 environment and highlights the magnitude of this stochasticity in the outlined box. Our intuition is the model may have exhausted a certain search pathway and is unwinding previously important connections that were inherited from the parents. This is possible because the "elite" parents are never fully cloned into the new population which means subsequent populations can decrease their solution quality. Additionally, the poor test reward early in training suggests finding the first network that beats random performance is particularly difficult, but once found training accelerates.

Figure 3: Test reward fluctuations during subnetwork training.



To elucidate how model shapes influences the difficulty of the sparse network search, we fixed parameter counts and varied the model shape. Table 1 displays the results for the LunarLander and LunarLanderContinuous environments. Here, we left sparsity unconstrained and found the resulting subnetworks have sparsity's between 45 - 55 percent indicating the normal noise keeps the sparsity well centered at 50 percent for thresholding above 0. A notable result is that the sparse network search becomes more challenging and networks get deeper. This is particularly evident for the LunarLander task which was only able to be solved for with a single hidden layer. On average, even when high performing solutions are found for the deeper neural network it takes 3x as many samples to to identify these networks.

Task	Model	Hidden Layer Dimensions	Number of Parameters	Intialization	Test Reward	Samples (x1M)
LunarLanderContinuous	FC	(64,64,64)	8448	Uniform	254	0.91
LunarLander	FC	(64,64,64)	8448	Uniform	194	-
LunarLanderContinuous	FC	(87,87)	8448	Uniform	267	0.17
LunarLander	FC	(87,87)	8448	Uniform	77	-
LunarLanderContinuous	FC	(704,)	8448	Uniform	272	0.36
LunarLander	FC	(704,)	8448	Uniform	241	1.21

Table 1: Highest test reward achieved and samples required to achieve solved reward of 200 for various model sizes.

Finally we explored methods to control sparsity by tuning the thresholding value. Our results suggest that sparsity up to 90 percent can be enforced reliably. Qualitatively enforcing sparsity has unclear effect on the number of training samples required to hit a solution. In extreme cases the more compact search space seems to accelerate time to solution. One of the most difficult challenges we encountered during debugging was determining the proper size of the overparameterized network we begin with.

6 Conclusion/Future Work

We have demonstrated that in small network contexts, evolution provides an excellent method to find sparse neural networks within randomly weighted neural networks. To our knowledge this is the first demonstration of its kind. We make qualitative arguments suggesting the nature of the search is stochastic and may involve the unwinding of key connections in order to overcome dead-end networks. Although the algorithm displays high sensitivity to initialization seed as we would expect, this increased computational overhead must be evaluated fairly. This algorithm tackles two problems, namely finding a solution and ensuring the solution has a degree of sparsity. Perhaps even more important is that this algorithm only requires inference passes (i.e. matrix multiplication) and is truly parallelizable - each environment can be run in isolation.

Future work should start at studying methods for scaling the network search process and improving the initial network initialization. While it may be feasible to run through several seeds for the small networks studied in this paper, this overhead quickly scales for larger problem spaces. The ability to search in deeper networks may allow this method to scale to support larger image models.

7 Contributions

This was a solo project but the author would like to thank and acknowledge former Stanford CS229 TA and current Apple employee Mario Srouji for his mentorship. This includes introducing the author to genetic algorithms and providing code to work with.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[2] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[3] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[4] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network?, 2020.

[5] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

[6] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. On parallelizability of stochastic gradient descent for speech dnns. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 235–239. IEEE, 2014.

[7] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.

[8] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask, 2020.