Finding Lottery Tickets in LSTM Networks

Rolf Anderson
Electrical & Computer Engineering
Iowa State University
Ames, Iowa, USA
rolf@iastate.edu

***Abstract*—The Lottery Ticket Hypothesis has become a hot topic in machine learning circles. The focus of significant amounts of both private and public research funding, its investigation has largely been restricted to feed-forward networks, specifically convolutional and GAN networks. In this paper, I investigate if finding lottery tickets in LSTM networks is possible, and what differences, if any, there are between it and feed-forward networks.**

***Keywords*—Machine learning, Deep learning, LSTM, Lottery Ticket Hypothesis**

I. Introduction

When it was first proposed by Frankle et al. in 2019 [1], the **Lottery Ticket Hypothesis** (LTH) took the machine learning research field by storm. In the world of dynamic computing, relatively cheap resource prices and low interest rates led to a culture of excess. Deep learning algorithms quickly swelled to unbelievable size, over-parameterized and bulky to no end. In a way, deep learning systems have moved past human comprehension. With complexities so great it would have seemed impossible just 10 years ago, some of the drawbacks of excessive networks began to appear. Not all researchers and engineers were comfortable with the state of affairs, and the field was waiting on a silver bullet to allow complexity reduction without sacrificing too much in performance, training time, or other metrics. The Lottery Ticket Hypothesis and its promise of relatively easy to find spare networks felt like a solution.

II. Background

A. *Drawbacks of Excessive Networks*

Unnecessarily deep and complex networks suffer from a number of drawbacks that make reducing it a priority. First, they suffer from stability issues with regards to their function. Among them includes *vanishing*, *exploding*, and *shattering gradients* [9]. While there are methods to combat gradient stability risks, they are never fully removed and become more likely the deeper and wider a network is. Another drawback is training is inefficient. This may not be as much of an issue for large tech giants with virtually infinite resources, but has kept many universities and small to medium companies out of the machine learning field. What this has led to is an almost feudal state of affairs, where a small number of top-level companies are able to afford to develop, train, and operate the models that have become so intertwined with modern business and technology. Similarly, overly complex networks have made analysis of them very difficult [10]. It may not seem like a pressing concern, however analysis of the form and function of deep networks—especially when they encounter problems—can be very important in their development. With the ever-increasing responsibilities delegated to machine learning systems, the potential for investigation and accountability will be paramount to whether the public accepts or rejects their use. Whether from science fiction or an inherent fear of the uncanny, it's clear the public distrusts machine learning and the vaguely defined "artificial intelligence." To continue to create systems opaque to human understanding is to risk a public backlash the first time an AI system accidentally kills someone. Identifying and explaining how things work and why certain actions occurred goes a long way to easing the concerns of many.

Much of the blame for the excessive complexity of deep networks is the economic factors. For over a decade, tiny interest rates enabled significant investment in computing resources to develop and deploy deep learning systems. This may be changing soon, as a number of factors are combining which will in all likelihood see the era of Moore's law end completely soon. It is out of the scope of this paper, but potential armed conflict over Taiwan (the most important semiconductor producer in the world), supply chain breakdowns highlighted by the Covid-19 epidemic, interest rate increases, and manufacturing hitting an efficiency wall could lead to significant price increases in processing hardware necessary for working with the enormous models used in industry today. Companies like OpenAI won't be able to afford their current operations costs when server CPUs cost five times more than they do at present.

B. *Lottery Ticket Hypothesis*

As stated previously, in 2019 Frankle et al. dropped their bombshell paper "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks" [1]. In it, they propose a method of reducing network complexity by **pruning** parameter counts. According to them, overall parameter counts can be reduced by as much as 90% without significant accuracy decreases. To be specific, the Lottery Ticket theory holds that: "dense, randomly-initialized, feed-forward networks contain subnetworks (*winning tickets*) that—when trained in isolation—reach test accuracy comparable to the original network in a similar number of iterations" [1]. Essentially, a winning lottery ticket is a specific initialization of a model that leads to model with a bare-minimum number of parameters and thus complexity. The implications of this theory were massive, and the waves caused by it quickly reached many other research organizations in the field. As a result, a mini bull-run on the theory led to dozens of papers exploring the theory further in the five years since [2][3][4][5][6][7]. Furthermore, the major players in the machine learning space pumped a fair amount of money into its research. Google and Facebook have both published Github repositories with basic code and scripts related to it, demonstrating that the corporate world also sees potential in it for business reasons. Seeing major corporations dumping resources into a novel research topic suggests they see fiscal opportunity in it. If it was just a research vanity project, they likely would not pursue it to an official degree.

Despite the game-changing potential of it, there is one major complication. How does one identify a lottery ticket without first spending significant resources training a network? It is feared that the Lottery Ticket Hypothesis could turn out to be an NP problem. Like other NP problems, a lottery ticket is easy (relatively) to verify but computationally difficult to answer. Fortunately, there are some (partial) solution strategies for this in the research literature. You et al. identified what they named **Early-Bird (EB)** tickets [5], which are winning tickets identified through simple, low-resource intensive training regimens. What they discovered, was that the core network that comprises the result of a winning ticket, emerges early in training. Thus, allowing the discovery of winning initializations without being forced to use the same resource allocations in training as a non-pruned network. However, their method still requires some level of wasted effort in resources in their ticket discovery regimen. The golden achievement would be the discovery of a method to identify the winning ticket without requiring training. The solution to that problem has proven elusive.

Intertwined with the quest for easily discoverable lottery tickets is the idea that bounds can be put on the minimum complexity required for a given solution. In theoretical math and computer science, bounds are important, and if those can be identified for a certain problem, they could contribute to a better conceptualization of how deep networks solve the problem as well as how to design efficient network architectures. Mehmeti-Göpel et al., in their paper on non-linear pruning, found that the width and depth of their winning ticket networks converged on certain values for a given task [9]. Their experiments showed this convergence even across different network architectures. This could lead to interesting breakthroughs in the future, with problem and domain analysis being the key to finding winning lottery tickets in polynomial time.

The most common method of pruning in the research on lottery tickets is weight pruning, where the network weights are pruned. There are a few methods for determining which weights to prune. In the original paper, Frankle et al. pruned the bottom p% of weights by absolute value [1]. Other methods include pruning all weights below a certain value, or pruning the lowest p% of weights according to their L1-norm or Ln-norm values. This last strategy is officially supported by the PyTorch Python library. Pruning can be performed in single-shot operations, or it can be an iterative process. In iterative pruning, the model is repeatedly initialized (or reinitialized), trained, pruned, and evaluated. This method tends to involve smaller pruning rates than single-shots, and potentially leads to higher performance networks. There are other more niche methods of complexity pruning not involving weights. Mehmeti-Göpel et al. proposed a method whereby ReLU activation functions are methodically replaced by Parametric ReLU (PReLU) activations, thereby *partially linearizing* the network [9]. Their method operates on the premises that linearizing components in a deep network prune it in the same way that weight pruning does.

C. Long Short-Term Memory Networks

The Long Short-Term Memory (LSTM) network is a type of **Recurrent Neural Network (RNN)** created to solve some of the issues with vanilla RNNs. It adds short-term memory to an RNN that, while not indefinite, can last for hundreds or thousands of steps; thus, the curiously dissonant "long short-term" descriptor. LSTM networks have wide applications including image tasks as well as **Natural Language Processing (NLP)** workflows. LSTM's show their strength best in time-series and serialized data flows, which are very common in NLP. An LSTM network's architecture resembles a two-dimensional grid with individual cells feeding up, forward, and backward (if it is bi-directional). Cells retain memory for an arbitrary amount of time. The depth of an LSTM network is defined by how many layers of cells it is configured with.

III. Related Work

A. Previous Work on Other Network Paradigms

For a while, especially in the weeks and months directly following the proposal of the LTH, researchers considered that lottery tickets might be a quirky nuance of feed-forward networks or image tasks. It was very new, and took some time before the new research literature appeared. In the wake of the gold rush, there were a few attempts to find lottery tickets in non-CNN networks. Yu et al. published a paper in 2020 on finding lottery tickets in Reinforcement Learning (RL) networks [3]. They were able to achieve significant but varied pruning rates in the tasks they experimented on. Interestingly, while tasks like LunarLander-v2 and Acrobot-v1 allowed pruning rates beyond 90% with no impact to reward rates, the same was not true for CartPole-v0 [3] (LunarLander-v2, Acrobot-v1, and CartPole-v0 are popular, simple games for testing RL networks). Furthermore, whereas with some tasks like Breakout and LunarLander-v2, reward rates plummeted past a certain amount of weights pruned, other tasks like Assault and Seaquest saw gradual reductions in performance. Residual Neural Networks (ResNets) have also been a target of research into the LTH. García-Arias et al., in a paper published in IEEEAccess in 2023, found winning tickets in ResNets [7]. In fact, they discovered that pruned ResNets "are more accurate and parameter-efficient than both the ones found within feedforward models and than the full [ResNet] models with learned weights" [7]. These results are surprising, the original theory holds that winning tickets produce networks as performant as the full network, but not more performant.

B. Previous Work on LSTM Networks and NLP

In addition to their work on RL networks, Yu et al. also tested for winning tickets in LSTM networks Their research found that the winning tickets outperformed randomly initialized pruned networks, confirming they are winning tickets. Their paper mostly focused on RL networks however, and only experimented on an omnidirectional, 2-layered model.

IV. The Lottery Ticket Hypothesis on LSTM Networks

Most of the research on the LTH has been confined to feed-forward networks trained for image classification tasks, as well as **Reinforcement Learning** networks and tasks. Beyond that, almost all research into lottery tickets has focused on image related tasks, like classification and recognition. Little effort has been spent on studying the LTH in more niche network paradigms like GANs and LSTMs, and what effort has been put into LSTM's has not explored the differences in pruning behavior or winning tickets based on differences in model characteristics, such as its training rate and layer count. This paper explores that. To test it, I will be employing two types of experiments, simple and complex. The simple experiment consists of designing and training a network best for a given task, then pruning it until performance craters. This ideally should prove that most of the LSTM network is redundant, and give an idea as to how much sparsity should be expected in a winning ticket. The second will be training the network, saving the initialization and weight masks, then testing a new model trained with the saved initialization and weight masks. This should confirm that LSTM networks are capable of producing winning tickets, and fit into the Lottery Ticket Hypothesis.

To prove that LSTM networks produce lottery tickets, the experiments must demonstrate a few things. First, they will need to demonstrate that trained networks are over-parametrized. To show this, I will train several models for each of the datasets described. If the trained models hold their performance to a reasonable amount past at least 50% pruned, it will be fairly clear that they produce lottery tickets. To completely prove the paradigm fits, another more specific test must be passed. In this experiment, I will train a network, prune it to a certain amount, then apply that same mask to two newly initialized models. One of the models is initialized with the same parameters as the previously trained one, while the other is initialized with randomly selected parameters. The first one is the 'winning ticket' model, whereas the other is the 'randomly-initialized' model. If the winning ticket holds its performance significantly further than the randomly-initialized one, the experiments will have proved that LSTM networks fit into the Lottery Ticket Hypothesis. To explain a bit, the latter experiment proves the hypothesis in what is basically a statistical proof. The null hypothesis is the winning ticket does not contain a special set of initial parameters, rather the results seen in experiment one are due entirely to some quirk of the network, independent of initial state. If it is shown that the winning ticket produces a uniquely strong result, the null hypothesis is rejected, and we conclude that the winning ticket's initial parameters allow it to be pruned to a significantly greater degree than a randomly created network.

A secondary goal of this paper is to investigate the impacts of hyperparameter variance on lottery ticket behavior in LSTM networks. Of the little research on LSTM lottery tickets, they almost always restrict their testing to one dataset and one model configuration. Even though varying the hyperparameters may reduce the nominal network performance, what is important is investigating the relative figures within each run of a certain set of hyperparameters. It seems logical that with increasing network depth (through increased layers) will come greater tolerance to network pruning, especially since pruning only removes the least important weights.

V. Implementation

A. Software & Hardware Stack

In development and testing, I leveraged the ubiquitous PyTorch Python library for model management, training and testing. I executed all experiments on a 2021 MacBook Pro, which allowed usage of the novel Torch device "MPS" as a Cuda alternative. This was a boon for testing, as it sped up training and testing times—although it was still much slower than what would be expected from a modern Nvidia GPU with Cuda support. It would not be surprising that without PyTorch's support of Apple Silicon GPU's, the production time of this paper would have doubled. The development, testing, and experiments contained in this paper took many hours to execute, and it's difficult to think of how much longer they might've taken were they CPU bottlenecked.

B. Datasets

As I was focusing on NLP tasks, I chose a few solidly representative datasets for testing the LTH. The first one, concerns the sentiment analysis of a set of Tweets about entities in the finance world, and their sentiment. This dataset was used in class. It's not a large set by any means, but large and varied enough to be useful for development of training, testing, and pruning functions. Processing it required using Regex to remove 'http[s]://', '@', and other unnecessary components of a Tweet. I sourced several datasets from a PyTorch related repository. The first is a set of 50,000 reviews and their ratings from IMDB (a website concerning movie and T.V. show reviews). It is a large dataset, with the largest entries containing almost 2,500 individual tokens. Unlike simple Tweets with defined metacharacters, this dataset included all of the HTML tags included in the originally scraped texts. Again, Regex was employed to shave these undesirable tokens off, saving a lot of space in the end. Originally some of the entries included greater than 3,000 individual tokens.

C. Model

For the model primarily used during the experiments for this paper, I selected a bi-directional, multi-layered LSTM design with a dropout and fully connected layer preceding the output. I varied the number of layers during experiments, to test how that affects sparseness scores after pruning.

D. Custom Software

As PyTorch's support for model pruning is still developing—in fact a significant portion of core functions used in network prunin are still in alpha—I implemented functions and classes necessary for this paper. PyTorch supports some pruning methods, however all of them modify the network in place, and do not allow masks to be built. Masks are necessary for testing the Lottery Ticket Hypothesis, as there must be ways to apply the exact pruning of one network to a control. With PyTorch's current methods, this is impossible. Furthermore, rather surprisingly, PyTorch does not include a simple function for pruning the bottom p% of nonzero parameters, rather they implement methods to prune the bottom p% by their L1-norm or Ln-norm values. It's great that this is possible, but unnecessary for the purposes of finding and training lottery tickets. As such, I implemented a few different methods of pruning the bottom p% of weights. The ones I primarily used were split into two categories, mask construction and mask application. I made the functions as generic as possible, with recursion and type checking. This allows masks to be built from and applied to arbitrary networks, not just LSTM ones. To run the experiments, I built out a few test functions automating the management of models, their training and evaluation.

VI. Experiments and Analysis

A. *Pruning After Training*

To begin my experiments, I tested the performance of trained models as weights are pruned. For each task and dataset, I selected a number of task run configs, with varying model and training parameters. The main independent variables were the number of layers of the model, number of training epochs, and the learning rate during training. As both the financial tweet and IMBD reviews problems were classification with two outputs, the floor of the performance should have been 50%. However, curiously the financial analysis models dropped into the low 30% range. I don't think there is any real significance to this, it is probably just a quirk of the dataset having an imbalance between positive and negative sentiment datapoints.

The results of this experiment confirm that LSTM networks are overparametrized, and are capable of seeing more than 90% reductions in activated weights. For each of these runs, the mask and original initializations were saved. A winning ticket in this context is the original initialization with the masked weights set to zero, deactivating them.

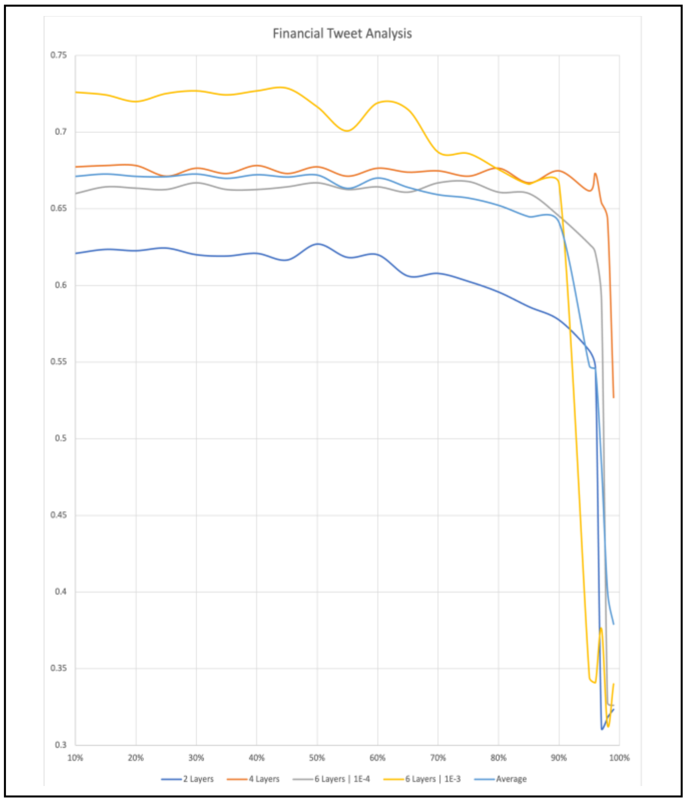

Fig. 1. Financial Tweet Test Accuracy as Pruning Increases

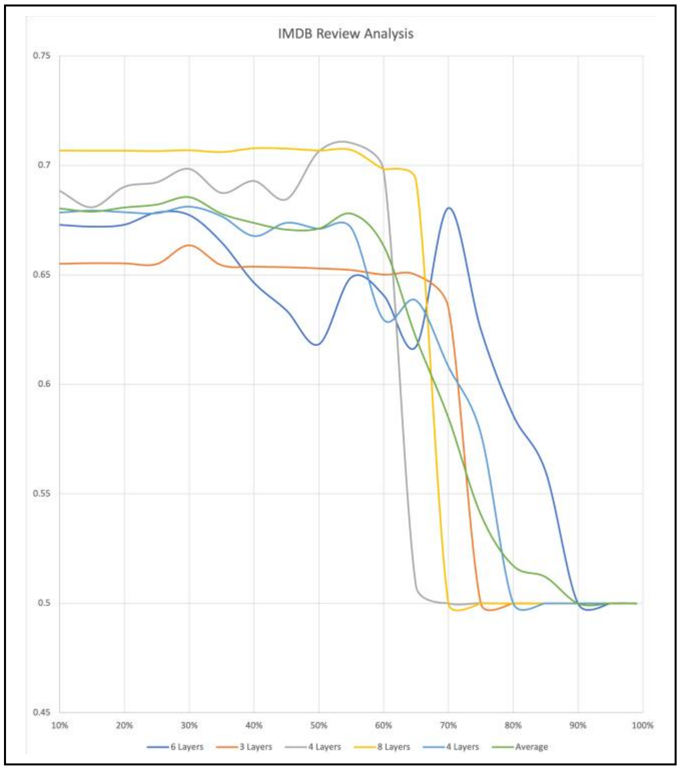

Fig. 2. IMDB Review Test Accuracy as Pruning Increases

Both of these results, while not conclusively proving that LSTM's generate lottery tickets, go a long way to hint at it. Should it be proven in this paper that LSTM networks generate lottery tickets, Figures 1 & 2 show what the expected network performance is for a given amount of pruning. Figure 1 examines the simpler Financial Tweets task. Not surprisingly, models trained on it hang on to their performance much longer. All of the networks' performance crater after 90% pruning, with most only cratering after 95%. Now, this analysis has been in relative terms. When compared to each other, there are clear performance tiers, with most falling in a middle pack. The main causes for variation in absolute performance was layer count. The best performing network had 6 layers, while the worst had 2. Figure 2 displays the experimental results of pruning LSTM networks trained on the IMDB Review task. This task is far more complex, with entries having as many as 2,500 tokens. The difficulty of the problem led to worse overall performance compared to the smaller set, both in absolute performance as well as resistance to pruning. The pruning rate at which each model broke down was far more varied than in the previous task. Some of the models began cratering once the rate reached 60%, others held on past 70%. In a reassuring turn, this time none of the models' accuracy evaluations went below 50%, again I think the larger dataset helped this out. Even though the performance and breakdown points were inferior to the tweet analysis task, they still clearly demonstrated over-parameterization in the networks.

The secondary goal of this paper is to examine the differences, if any, between models of different hyperparameters and their lottery tickets. While I rightfully hypothesized that deeper networks would perform outright better, I was incorrect in thinking they would resist performance breakdown more than the shallower networks. The dataset is not enough to draw any clear conclusions from, but it holds consistent across both experiments. At least it’s clear there is no positive correlation between depth and pruning resistance. It begs further study. I did not perform enough tests on variability in learning rates during training to make any reasonable inferences about what the results were.

B. *Confirming Lottery Tickets*

In order to confirm lottery tickets are special due to their unique initialization (combined with the mask), we have to compare a lottery ticket to a non-lottery ticket with the same mask. I restricted this experiment to the IMDB task. As discovered in experiment one, it gives better resolution in performance when independent variables are changed. To perform it, I followed the steps articulated in section IV: build and train a model, derive a mask from it and apply it to two new networks, one with the same initial weights as the original, the other with new weights. I then trained them for 10 epochs and compared their performance. I repeated this iteratively for pruning rates from 10% to 95%. The results of the experiment can be seen in Figure 3. The winning ticket holds a clear and significant advantage over the losing ticket. Although they remain fairly similar up to 45% pruning, immediately after the losing ticket begins cratering. The winning ticket remarkably holds on past 80% pruning. Thus, it can be confidently concluded that LSTM networks fit into the greater group of deep networks that are privy to the Lottery Ticket Hypothesis.

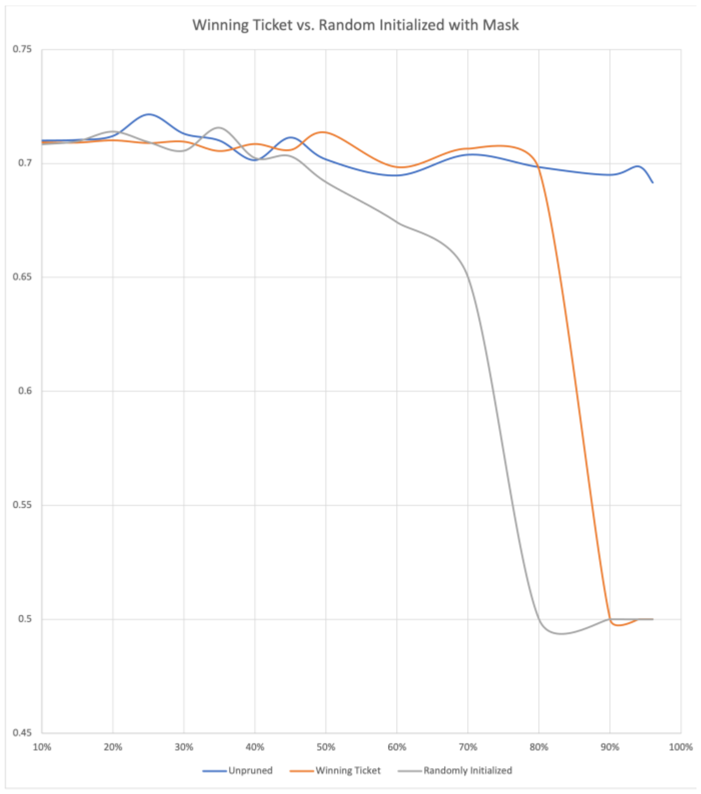

Fig. 3. Performance of Winning vs. Losing Ticket

C. *Post-Pruning Training*

A strategy used by some researchers [7] when investigating novel processes related to lottery tickets is to conduct a brief training burst after pruning the network to refine any unwanted inconsistencies introduced by eliminating large portions of the network. Often this can have significant impacts on the performance of the pruned network. To test this, I created three different LSTM networks. All with 3 layers, trained for 15 epochs before pruning. I measured accuracy before and after a regimen of post-pruning training consisting of 2 epochs with a relatively high learning rate. Figure 4 shows the plotted results, with post-retrained scores represented by dashed lines. Generally, the brief training exposure improved accuracy of the network. Curiously, this effect is stronger in weakly or strongly pruned networks, yet mid-level pruning often resulted with poorer accuracy post-retraining. More research is necessary to determine the viability—or necessity—of it. It appears that strongly pruned networks stand to gain the most from it, as past 60% pruning it provides significant improvements to all three networks. Another curiosity is the varying effect depending on the learning rate of the original training run. The model trained with the lowest learning rate, which performed worst overall, consistently gained between 5 and 10% in its accuracy evaluations. Conclusions are difficult to make, but it seems worthwhile for future investigators of LSTM lottery tickets to consider.

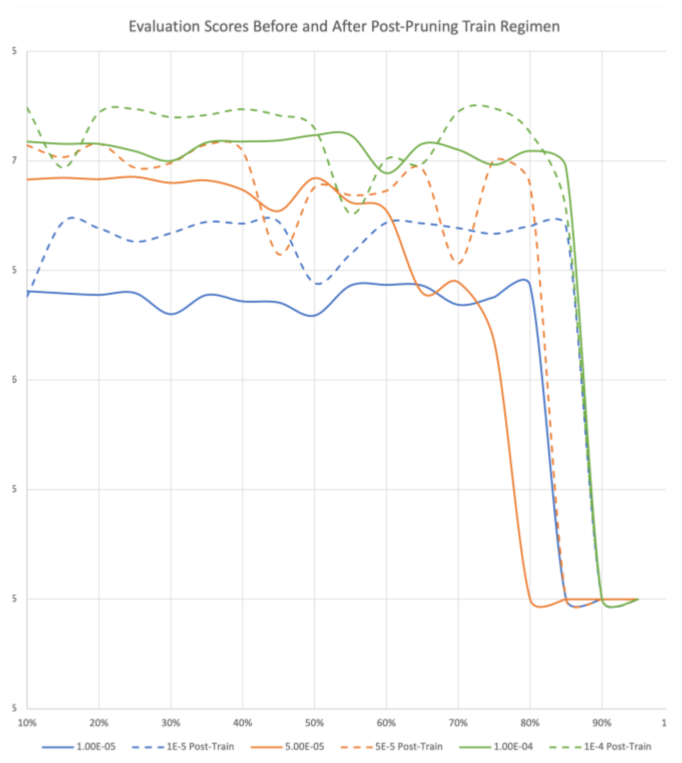

Fig. 4. Testing Scores Before and After Post-Prune Training

VII. Conclusion

At this point, with all of the extent research universally concurring, it is safe to conclude that the lottery tickets are real, and exist within all deep networks. If a simple research paper for a graduate class is able to confirm them in LSTM architectures, then it's certain that they are everywhere. More research is needed, but my results also suggest differences in lottery ticket behavior and network pruning based on the network's hyperparameters. It may have been an artifact of empirical analysis, and the randomness in real life testing. However, similar research literature suggests that problems have abstract constraints on lottery ticket networks, regardless of system configuration or even network architecture.

It's amazing, and a testament to the interest in deep learning in general, that a topic can so quickly arrive and explode in the research literature. The Lottery Ticket Hypothesis offers a solution for the timeless problem of over-complex system design that is by no means exclusive to the field. If it can be harnessed, it could solve many problems. It could help democratize a field that is basically off-limits to all but the most well-funded of organizations. It could eliminate some of the stability issues encountered by larger networks. It could allow easier study of networks, easier diagnosis of issues, easier investigation of faults. It is a fascinating topic that both promises to ease problems while shedding a light on our over-reliance of brute-force excessive machine learning systems to make development easier. Unfortunately, much like other NP problems, it's promises are so compelling, but locked behind a wall of resource intensive solution finding. I experienced that heavily in this project. Finding lottery tickets was about as resource intensive as just training them as-is from the start. Yes, a pruned network can be stored in less memory, but that is frankly secondary to the processing power required, which is hardly conserved employing lottery tickets. Early-Bird tickets probably hold the most promise within the realm of the Lottery Ticket Hypothesis. They're a satisfactory compromise between finding full winning lottery tickets and spending the resources to do so. It appears likely that the future of LTH research will center on that concept as a blueprint for practical applications of it. Not many other NP or NP-ish problems include backdoors like it. Only time and more research will tell if there is a P-time way to find general lottery tickets without training and searching for them. If there is one, and it is found, it will revolutionize deep learning. As such, even with my reservations, it's still worth investing in and I look forward to seeing further developments in it.

VIII. References

[1] Frankle, Jonathan, and Michael Carbin. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. arXiv:1803.03635, arXiv, 4 Mar. 2019. *arXiv.org*, http://arxiv.org/abs/1803.03635.

[2] Vaswani, Ashish, et al. *Attention Is All You Need*. arXiv:1706.03762, arXiv, 1 Aug. 2023. *arXiv.org*, http://arxiv.org/abs/1706.03762.

[3] Yu, Haonan, et al. *Playing the Lottery with Rewards and Multiple Languages: Lottery Tickets in RL and NLP*. arXiv:1906.02768, arXiv, 25 Feb. 2020. *arXiv.org*, http://arxiv.org/abs/1906.02768.

[4] Morcos, Ari S., et al. *One Ticket to Win Them All: Generalizing Lottery Ticket Initializations across Datasets and Optimizers*. arXiv:1906.02773, arXiv, 27 Oct. 2019. *arXiv.org*, http://arxiv.org/abs/1906.02773.

[5] You, Haoran, et al. *Drawing Early-Bird Tickets: Towards More Efficient Training of Deep Networks*. arXiv:1909.11957, arXiv, 16 Feb. 2022. *arXiv.org*, http://arxiv.org/abs/1909.11957.

[6] Chen, Xiaohan, et al. *The Elastic Lottery Ticket Hypothesis*. arXiv:2103.16547, arXiv, 27 Oct. 2021. *arXiv.org*, http://arxiv.org/abs/2103.16547.

[7] García-Arias, Ángel López, et al. "Recurrent Residual Networks Contain Stronger Lottery Tickets." *IEEE Access*, vol. 11, 2023, pp. 16588–604. *DOI.org (Crossref)*, https://doi.org/10.1109/ACCESS.2023.3245808.

[8] Lange, Robert Tjarko. "The Lottery Ticket Hypothesis: A Survey." *Rob's Homepage*, 27 June 2020, https://roberttlange.github.io/posts/2020/06/lottery-ticket-hypothesis/.

[9] Mehmeti-Göpel, Christian H. X. Ali, and Jan Disselhoff. Nonlinear Advantage: Trained Networks Might Not Be As Complex as You Think. arXiv, 1 June 2023. https://doi.org/10.48550/arXiv.2211.17180.

[10] Allen-Zhu, Zeyuan, et al. "Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers." *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. *Neural Information Processing Systems*, https://proceedings.neurips.cc/paper/2019/hash/62dad6e273d32235ae02b7d321578ee8-Abstract.html.

[11] Elliott, Desmond, et al. *Multi30K: Multilingual English-German Image Descriptions*. arXiv:1605.00459, arXiv, 2 May 2016. *arXiv.org*, http://arxiv.org/abs/1605.00459.

Appendix: Source Code

https://github.com/asaprolfy/lstm-lottery-tickets