# Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP

**Haonan Yu**
Facebook AI Research
haonanu@gmail.com

**Sergey Edunov**
Facebook AI Research
edunov@fb.com

**Yuandong Tian**
Facebook AI Research
yuandong@fb.com

**Ari S. Morcos***
Facebook AI Research
arimorcos@fb.com

## Abstract

The lottery ticket hypothesis proposes that over-parameterization of deep neural networks (DNNs) aids training by increasing the probability of a "lucky" sub-network initialization being present rather than by helping the optimization process [8]. This phenomenon is intriguing and suggests that initialization strategies for DNNs can be improved substantially, but the lottery ticket hypothesis has only previously been tested in the context of supervised learning for natural image tasks. Here, we evaluate whether "winning ticket" initializations exist in two different domains: reinforcement learning (RL) and in natural language processing (NLP). For RL, we analyzed a number of discrete-action space tasks, including both classic control and pixel control. For NLP, we examined both recurrent LSTM models and large-scale Transformer models [30]. Consistent with work in supervised image classification, we confirm that winning ticket initializations generally outperform parameter-matched random initializations, even at extreme pruning rates. Together, these results suggest that the lottery ticket hypothesis is not restricted to supervised learning of natural images, but rather represents a broader phenomenon in DNNs.

## 1 Introduction

The lottery ticket hypothesis [8, 9, 32] is an interesting and surprising phenomenon in which small, sparse sub-networks can be found in over-parameterized deep neural networks (DNNs) which, when trained in isolation, can achieve similar or even greater performance than the original, highly over-parameterized network. This phenomenon suggests that over-parameterization in DNN training is beneficial primarily due to proper initialization rather than regularization during the training process itself [1, 2, 6, 7, 20, 21].

However, despite extensive experiments in [8, 9], it remains unclear whether the lottery ticket phenomenon is an intrinsic feature of DNN behavior, or whether it is dependent on other factors such as supervised learning, network architecture, specific tasks (e.g., image classification), the bias in the dataset, or artifacts from the optimization algorithm itself. As discussed in [8, 15], large learning rates severely damage the lottery ticket effect, and for larger models (such as VGG and ResNets) and datasets (e.g., ImageNet), heuristics like learning rate warmup [8] and late resetting [9] are needed to induce high performance and reliable winning tickets. Recent work has also questioned the effectiveness of the lottery ticket hypothesis, raising concerns about the generality of this phenomenon [10, 15].

*To whom correspondence should be addressed

Preprint. Under review.

In this work, we address the question of whether the lottery ticket phenomenon is merely an artifact of supervised image classification with feed-forward convolutional neural networks, or whether this phenomenon generalizes to other domains, architectural paradigms, and learning regimes (e.g., environments with reward signals). This question is particularly pressing given the concerns raised regarding the brittleness of the lottery ticket phenomenon as well as 1) the substantial differences between supervised learning and other learning paradigms, such as reinforcement learning (RL), and 2) the substantial differences between learning independent and identically distributed (i.i.d.) data and learning sequential data. For example, in RL the data distribution shifts as the agent learns from often reward signals, significantly modifying the optimization process and the resultant networks. Pre-trained feature extractors have proven successful in computer vision [13, 26, 31], but in RL, agents often fail to generalize even to extremely similar situations [5, 14, 25, 27]. In most NLP tasks, DNNs need to model temporal dynamics which is absent in supervised image classification.

To answer this question, we evaluate the lottery ticket hypothesis on two broad tasks that are drastically different from traditional supervised image classification: natural language processing (NLP) on Wikitext-2 [16], machine translation on WMT'14 English-German translation task, and reinforcement learning applied to classic control problems and Atari games [4]. In NLP, network architectures are typically much more diverse than in supervised image classification (e.g., embedding-based models [17], recurrent models [28], self-attention [30], etc.). In contrast, though network architectures in RL are generally quite similar, the learning settings and data distributions are highly different: RL models often train on rewards and their own outputs, datasets are dynamic with regular distributional shifts induced by current behavior policies, and finally, tasks vary dramatically from game to game.

Perhaps surprisingly, we found that lottery tickets are present in both RL and NLP tasks. In RL, we observed winning tickets in both classic control problems and Atari games, while in NLP, winning tickets were present both in recurrent LSTMs trained on language modeling and in Transformers [30] trained on a machine translation task. Together, these results demonstrate that the lottery ticket phenomenon is a general property of deep neural networks.

## 2 Related work

Our work is primarily inspired by the lottery ticket hypothesis, first introduced in [8], which argues that over-parameterized neural networks contain small, sparse sub-networks (with as few as 0.1% of the original network's parameters) which can achieve high performance when trained in isolation. [9] extended the lottery ticket hypothesis to large-scale image classification datasets and introduced the notion of late resetting, which was found to significantly improve performance for large-scale models. However, both of these works solely focused on supervised image classification, leaving it unclear whether the lottery ticket phenomenon is present in other domains and learning paradigms.

Recent work [15] challenged the lottery ticket hypothesis, demonstrated that for structured pruning settings, random sub-networks were able to match winning ticket performance. [10] also explored the lottery ticket hypothesis in the context of ResNets and Transformers. Notably, they found that random sub-networks could achieve similar performance to that of winning ticket networks. However, they did not use iterative pruning or late resetting, both of which have been found to significantly improve winning ticket performance [8, 9].

More broadly, pruning methods for deep neural networks have been explored extensively [11]. Following [8, 9], we use magnitude pruning in this work, in which the smallest magnitude weights are pruned first [11]. To determine optimal pruning performance, [19] greedily prune weights to determine an oracle ranking. Also, [3, 24] have attempted to rank channels by redundancy and preferentially prune redundant filters, and [18] used variational methods to prune models. However, all of these works were performed in the supervised image classification.

## 3 Approach

### 3.1 Generating lottery tickets

**Pruning methods** In our experiments, we use both one-shot and iterative pruning to find winning ticket initializations. In one-shot pruning, the full network is trained to convergence, and then a given

| Type | Name | Network specs | Algorithm | $N$ | $M$ | $L$ |
|---|---|---|---|---|---|---|
| Classic | CartPole-v0 | MLP(128-128-128-out) | A2C | 20 | 160 (games) | 100 |
|  | Acrobot-v1 | MLP(256-256-256-out) | A2C | 20 | 320 (games) | 100 |
|  | LunarLander-v2 | MLP(256-256-256-out) | A2C | 20 | 640 (games) | 100 |
| Pixel | Assault, Berzerk,<br>Breakout, Centipede,<br>Kangaroo, Krull,<br>Qbert, Seaquest,<br>Space Invaders | Conv(5,64,1,2)-MaxPool(2)<br>-Conv(5,64,1,2)-MaxPool(2)<br>-Conv(3,64,1,1)-MaxPool(2)<br>-Conv(3,64,1,1)-MaxPool(2)<br>-MLP(1920-512-512-out) | A2C<br>(with<br>importance<br>factor<br>correction) | 25 | 1000<br>(batches) | 1024 |

Table 1: A summary of the games in our RL experiments. Conv($w$, $x$,$y$,$z$) represents a convolution layer of filter size $w$, channel number $x$, stride $y$, and padding $z$, respectively. All the layer activations are ReLUs. See Sec. 3.2.2 for the meaning of $M$, $N$ and $L$.

fraction of parameters are pruned, with lower magnitude parameters pruned first. To evaluate winning ticket performance, the remaining weights are reset to their initial values, and the sparsified model is retrained to convergence. However, one-shot pruning is very susceptible to noise in the pruning process, and as a result, it has widely been observed that one-shot pruning under-performs iterative pruning methods [8, 11].

In iterative pruning [8, 11], alternating cycles of training models from scratch and pruning are performed. At each pruning iteration, a fixed, small fraction of the remaining weights are pruned, followed by re-initialization to a winning ticket and another cycle of training and pruning. More formally, the pruning at iteration $k + 1$ is performed on the trained weights of the winning ticket found at iteration $k$. At iteration $k$ with an iterative pruning rate $p$, the fraction of weights pruned is:

$$r_k = 1 - (1 - p)^k, \;\; 1 \leq k \leq 20$$

We therefore perform iterative pruning for all our experiments unless otherwise noted, with an iterative pruning rate $p = 0.2$. For our RL experiments, we perform 20 pruning iterations. We also confirm the result that iterative pruning outperforms one-shot pruning in comparison experiments on 6 representative games in section 4.1.3. Pruning was always performed globally, such that all weights (including biases) of different layers were pooled and their magnitudes ordered for pruning. As a result, the fraction of parameters pruned across layers may vary given a total pruning fraction.

**Random tickets (RT)** To determine whether winning ticket initializations perform better than we'd expect by chance, we compare winning tickets (WT) to random tickets (RT), in which the same sub-network is used, but the weights are randomly re-initialized from the initialization distribution.

**Late resetting** In the original incarnation of the lottery ticket hypothesis [8], winning tickets were reset to their values at initialization. However, [9] found that resetting winning tickets to their values to an iteration early in training resulted in dramatic improvements in winning ticket performance on large-scale image datasets, even when weights were reset to their values only a few iterations into training. Late resetting can therefore be defined as resetting winning tickets to their weights at iteration $j$ in training, with the original lottery ticket paradigm taking $j = 0$. Unless otherwise noted, we use late resetting throughout, with a late resetting value of 1 epoch used for all RL experiments. We also compare late resetting with normal resetting on several representative RL games in section 4.1.3 and for NLP in section 4.2.

## 3.2 Reinforcement learning

### 3.2.1 Simulated Environments

For our RL experiments, we use two types of discrete-action games: classic control and pixel control. For classic control, we evaluated 3 OpenAI Gym[2] environments that have vectors of real numbers as network inputs. These simple experiments mainly serve to verify whether winning tickets exist in networks that solely consist of fully-connected (FC) layers for RL problems. For pixel control, we evaluated 9 Atari [4] games that have raw image pixels as network inputs. These experiments

[2] `https://gym.openai.com/`

test whether winning tickets exist in convolutional networks trained on RL problems. A summary of all the games along with the corresponding networks and methods to solve them is provided in Table 1. Classic control games were trained using the FLARE framework[3] and Atari games were trained using the ELF framework[4] [29].

### 3.2.2 Ticket evaluation

To evaluate a ticket (pruned sub-network), we train the ticket to play a game with its corresponding initial weights for $N$ epochs. Here, an epoch is defined as every $M$ game episodes or every $M$ training batches depending on the game type. At the end of training, we compute the averaged episodic reward over the last $L$ game episodes. This average reward, defined as *ticket reward*, indicates the final performance of the ticket playing a game by sampling actions from its trained policy. For each game, we plot ticket reward curves for both winning and random tickets as the fraction of weights pruned increases. To evaluate the impact of random seed on our results, we repeated the iterative pruning process three times on every game, and plot (mean $\pm$ standard deviation) for all results.

### 3.2.3 Hyperparameters

For most hyperparameters, we used standard values which were shared across games. Below we briefly review the most important ones. Note that our aim is not to get the best scores on the games, but rather to evaluate whether the lottery ticket phenomenon is present in the context of RL. As such, we did not extensively tune hyperparameters to maximize performance.

**Classic control** All three games were trained in the FLARE framework with 32 game threads running in parallel, and each thread gets blocked every 4 time steps for training. Thus a training batch contains $32 \times 4 = 128$ time steps. Immediate rewards are divided by 100. For optimization, we use RMSprop with a learning rate of $10^{-4}$ and $\alpha = 0.99, \epsilon = 10^{-8}$.

**Pixel control** All 9 Atari games are trained using a common ELF configuration with all RL hyperparameters being shared within a reasonable amount of training time (see Table 1 for our choices of $N$ and $M$). Specifically, each game has 1024 game threads running in parallel, and each thread gets blocked every 6 time steps for training. For each training batch, the trainer samples 128 time steps from the common pool. The policy entropy cost for exploration is weighted by $0.01$. We clip both immediate rewards and advantages to $[-1, +1]$. Because the training is asynchronous and off-policy, we impose an importance factor which is the ratio of action probability given by the current policy to that from the old policy. This ratio is clamped at $1.5$ to stabilize training. For optimization, we use Adam with a learning rate of $10^{-3}$ and $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-3}$.

## 3.3 Natural language processing

To test the lottery ticket hypothesis in NLP, we use two different setups: two-layer LSTMs for language modeling on Wikitext-2 [16] and Transformer-base [30] on the WMT'14 En2De News Translation task.

**Language modeling using LSTMs** For the language modeling task, we trained an LSTM model with a hidden state size of 650. It contained a dropout layer between the two RNN layers with a dropout probability of $0.5$. The LSTM received word embeddings of size 650. For training, we used truncated Backpropagation Through Time (truncated BPTT) with a sequence length of 50. The training batch size was set to 30, and models were optimized using Adam with a learning rate of $10^{-3}$ and $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-3}$.

As in the RL experiments, we use global iterative pruning with an iterative pruning rate of 0.2 and 20 total pruning iterations. We also employ late resetting where the initial weights of a winning ticket were set to the weights after first epoch of training the full network. For ticket evaluation, we trained the model for 10 epochs on the training set, after which we computed the log perplexity on the test set. We also perform two ablation studies without late resetting and using one-shot pruning, respectively.

[3] `https://github.com/idlrl/flare`

[4] `https://github.com/pytorch/ELF`

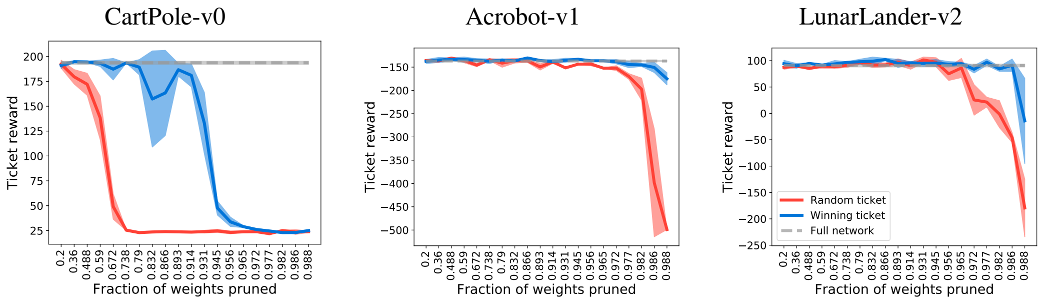

**Figure 1:** Winning ticket performance on classic control games. Each curve plots the mean ± standard deviation of three independent iterative pruning processes for each game.

**Machine translation using transformers** For the machine translation task, we use the FAIRSEQ framework[5] [22], following the setup described in [23] to train Transformer-base model on the pre-processed dataset from [30]. We train models for 50,000 updates on 128 Volta GPUs and apply checkpoint averaging. We report case-sensitive tokenized BLEU with `multi-bleu.pl`[6] on the newstest 2014 [7].

# 4 Results

## 4.1 Reinforcement learning

### 4.1.1 Classic control

To begin our investigation of the lottery ticket hypothesis in reinforcement learning, we evaluated three simple classic control games: Cartpole-v0, Acrobot-v1, and LunarLander-v2. As discussed in Section 3.2.1 and Table 1, we used very simple fully-connected models with three hidden layers. Encouragingly, and consistent with supervised image classification results, we found that winning tickets successfully outperformed random tickets in classic control environments (Figure 1). This result suggests that the lottery ticket phenomenon is not merely an artifact of supervised image classification, but occurs in RL paradigms as well.

### 4.1.2 Atari games

However, in the original lottery ticket study [8], winning tickets were substantially easier to find in simple, fully-connected models trained on simple datasets (e.g., MNIST) than in more complex models trained on larger datasets (e.g. ResNets on CIFAR and ImageNet). We therefore asked whether winning tickets exist in convolutional networks trained on Atari games as well. We found that the impact of winning tickets varied substantially across Atari games (Figure 2), with some games, such as Assault, Seaquest, and Berzerk benefiting significantly from winning ticket initializations, while other games, such as Breakout and Centipede only benefitted slightly. Notably, winning ticket initializations *increased* reward for both Berzerk and Qbert. Interestingly, one game, Krull, saw no such benefit, and both winning and random tickets performed well even at the most extreme pruning fractions, suggesting that Krull may be so over-parameterized that we were unable to get into the regime in which winning ticket differences emerge.

One particularly interesting case is that of Kangaroo. Because we used the same hyperparameter settings for all games, the initial, unpruned Kangaroo models failed to converge to high rewards (typical reward on Kangaroo for converged models is in the several thousands). Surprisingly, however, winning ticket initializations substantially improved performance (though these models were still very far from optimal performance on this task) over random tickets, enabling some learning where no learning at all was previously possible. Together, these results suggest that while beneficial winning

[5] `https://github.com/pytorch/fairseq`

[6] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

[7] `https://www.statmt.org/wmt14/translation-task.html`

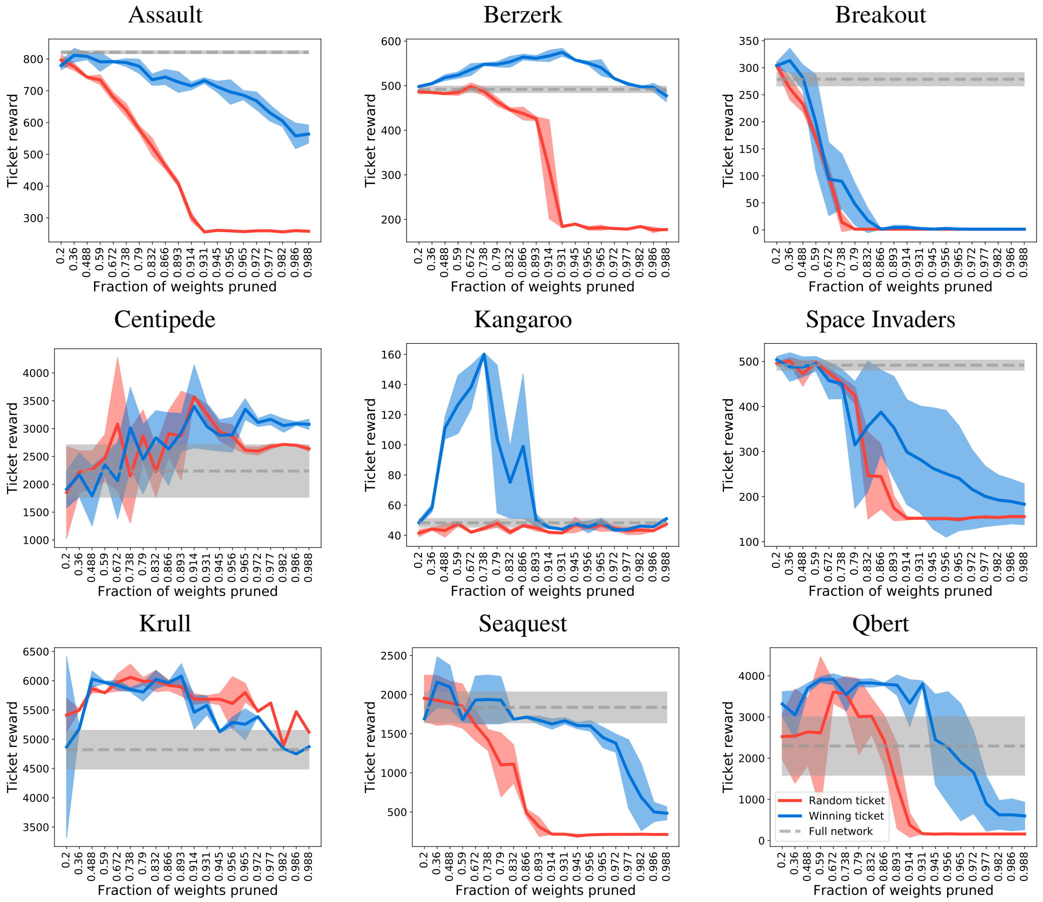

**Figure 2:** Reward curves of WTs (blue) and RTs (red) on Atari. Shaded error bars represent mean ± standard deviation across runs and the gray curve represents performance of the unpruned network.

ticket initializations can be found for some Atari games, winning ticket initializations for other games may not exist or be more difficult to find.

We also observed that the shape of pruning curves for random tickets on Atari games also varied substantially based on the game. For example, some games, such as Breakout and Space Invaders, were extremely sensitive to pruning, with performance dropping almost immediately, while other games, such as Berzerk, Centipede, and Krull actually saw performance steadily *increase* in early pruning iterations. This result suggests that the level of over-parameterization varies dramatically across Atari games and that "one size fits all" models may have subtle impacts on performance based on their level of over-parameterization.

### 4.1.3 Ablation studies

To measure the impact of late resetting and iterative pruning on the performance of winning ticket initializations in RL, we performed ablation studies on six representative games both from classic control and Atari: CartPole-v0, Acrobot-v1, LunarLander-v2, Assault, Breakout, and Seaquest. For all ablation experiments, we leave all training parameters fixed (configuration, hyperparameter, optimizer, etc.) except for those specified. For both classic control (Figure 3) and Atari (Figure 4), we observed that, consistent with previous results in supervised image classification [8, 9], both late resetting and iterative pruning improve winning ticket performance, though interestingly, the degree to which these modifications improved performance varied significantly across games.

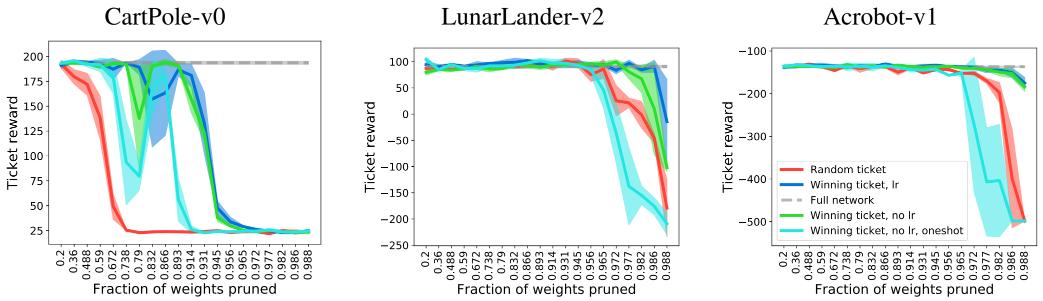

**Figure 3:** Ablation studies of several classic control games on the effects of late resetting and iterative pruning. Shaded error bars represent mean ± standard deviation across runs and the gray curve represents performance of the unpruned network.

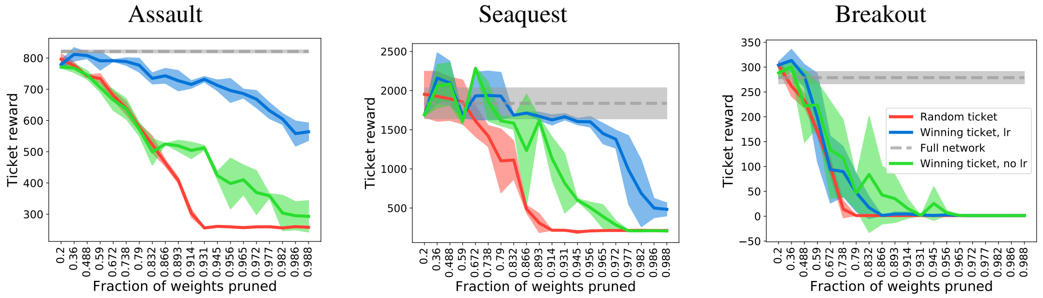

**Figure 4:** Ablation studies of several pixel control games on the effects of iterative pruning. Shaded error bars represent mean ± standard deviation across runs and the gray curve represents performance of the unpruned network. “lr” means late-resetting.

#### 4.1.4 Summary

Overall, our results demonstrate that winning tickets can be found in reinforcement learning tasks, confirming that the lottery ticket phenomenon is not merely limited to supervised image classification. However, as with RL more broadly [12], our results exhibited much higher variance than that observed previously, both with respect to the variance across model runs as well as the variance across RL tasks. The ability to prune models for many tasks with no change, or even an increase, in reward, suggests that further study of the lottery ticket phenomenon in RL may prove fruitful, but care will have to be taken to better understand the task properties which encourage successful winning tickets.

### 4.2 Natural language processing

In Section 4.1, we explored whether winning tickets exist in the context of reinforcement learning. Here, we ask this same question for natural language processing tasks, both using recurrent models and recently popularized Transformer architectures [30].

#### 4.2.1 Language modeling with LSTMs

We first investigate whether winning tickets exist in a two-layer LSTM model trained to perform a language modeling task on the Wikitext-2 dataset. Encouragingly, we again found that winning tickets significantly outperformed random tickets in this task, with as many as 90% of parameters capable of being removed without a noticeable increase in log perplexity (Figure 5. We again observed the importance of both late resetting and iterative pruning, as log perplexity for one-shot winning tickets without late resetting rose very quickly, with performance worse than the random ticket once 80% of parameters had been pruned. Interestingly, late resetting primarily seemed to help models at low pruning fractions, as iteratively pruned models with and without late resetting converged to similar performance past pruning rates of 91%. This result shows that winning ticket behavior in recurrent LSTMs trained on NLP tasks is similar to that of supervised image classification.

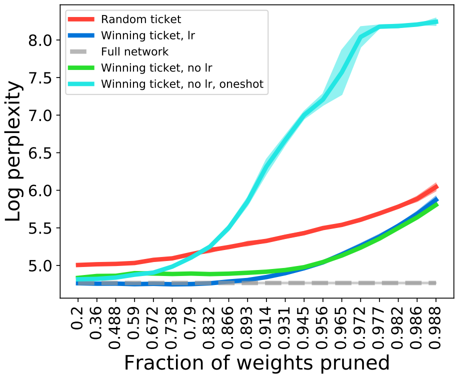

**Figure 5:** Performance of winning ticket initializations for LSTM models trained on Wikitext-2.

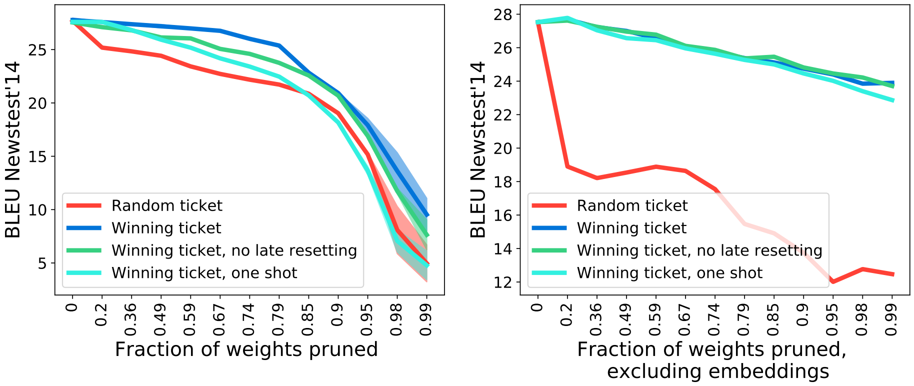

**Figure 6:** Winning ticket initialization performance for Transformer models trained on machine translation.

### 4.2.2 Machine translation with Transformers

We next evaluate whether winning tickets are present in Transformer models trained on machine translation. Our baseline machine translation model achieves a BLEU score of 27.6 (compared to 27.3 in [30]). We perform global iterative pruning with and without late resetting to the parameters of the baseline model after 1000 updates. Consistent with our previous results, winning tickets outperform random tickets in Transformer models (Figure 6 left). Additionally, we also found that iterative pruning and late resetting significantly improved performance, explaining why our results differ from [10], which only used one-shot pruning without late resetting.

We also tried a version of the Transformer model in which only Transformer layer weights (attention and fully connected layers) were pruned, but embeddings were left intact (Figure 6 right). Results in this setting were noticeably different from when we pruned all weights. First, the random ticket performance drops off at a much faster rate than in the full pruning setting. This suggests that, for random initializations, a large fraction of embedding weights can be pruned without damaging network performance, but very few transformer layer weights can be pruned. Second, and in stark contrast to the random ticket case, we observed that winning ticket performance was remarkably robust to pruning of only the transformer layer weights, with a roughly linear drop in BLEU score.

## 5 Conclusion

In this study, we investigated whether the lottery ticket hypothesis holds in regimes beyond simple supervised image classification by analyzing both RL and NLP domains. For RL, we found that winning ticket initializations substantially outperformed random tickets on classic control problems and for many, but not all, Atari games. For NLP, we found that winning ticket initializations beat random tickets both for recurrent LSTM models trained on language modeling and Transformer models trained on machine translation. Together, these results suggest that the lottery ticket phenomenon is not restricted to supervised image classification, but rather represents a general feature of deep neural network training.

## References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. November 2018.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via Over-Parameterization. November 2018.

[3] Babajide O Ayinde, Tamer Inanc, and Jacek M Zurada. Redundant feature pruning for accelerated inference in deep neural networks. *Neural networks: the official journal of the International Neural Network Society*, May 2019.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. December 2018.

[6] Simon S Du and Jason D Lee. On the power of over-parametrization in neural networks with quadratic activation. March 2018.

[7] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. October 2018.

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[9] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. The lottery ticket hypothesis at scale. March 2019.

[10] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. February 2019.

[11] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[12] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, 2018.

[13] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better ImageNet models transfer better? May 2018.

[14] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified Game-Theoretic approach to multiagent reinforcement learning. November 2017.

[15] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

[16] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. January 2013.

[18] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

[19] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. November 2016.

[20] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan‘ Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019.

[21] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. December 2014.

[22] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[23] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proc. of WMT*, 2018.

[24] Zhuwei Qin, Fuxun Yu, Chenchen Liu, and Xiang Chen. Interpretable convolutional filter pruning, 2019.

[25] Maithra Raghu, Alex Irpan, Jacob Andreas, Robert Kleinberg, Quoc V Le, and Jon Kleinberg. Can deep reinforcement learning solve Erdos-Selfridge-Spencer games? November 2017.

[26] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. March 2014.

[27] Avraham Ruderman, Richard Everett, Bristy Sikder, Hubert Soyer, Jonathan Uesato, Ananya Kumar, Charlie Beattie, and Pushmeet Kohli. Uncovering surprising behaviors in reinforcement learning via worst-case analysis, 2019.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[29] Yuandong Tian, Qucheng Gong, Wenling Shang, Yuxin Wu, and C. Lawrence Zitnick. Elf: An extensive, lightweight and flexible research platform for real-time strategy games. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[31] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., 2014.

[32] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. May 2019.