Evaluating Lottery Tickets Under Distributional Shifts

Shrey Desai[*,1] **Hongyuan Zhan**[2] **Ahmed Aly**[2]
[1]The University of Texas at Austin
[2]Facebook Assistant
shreydesai@utexas.edu
{hyzhan, ahhegazy}@fb.com

Abstract

The Lottery Ticket Hypothesis (Frankle and Carbin, 2019) suggests large, over-parameterized neural networks consist of small, sparse subnetworks that can be trained in isolation to reach a similar (or better) test accuracy. However, the initialization and generalizability of the obtained sparse subnetworks have been recently called into question. Our work focuses on evaluating the initialization of sparse subnetworks under distributional shifts. Specifically, we investigate the extent to which a sparse subnetwork obtained in a source domain can be re-trained in isolation in a dissimilar, target domain. In addition, we examine the effects of different initialization strategies at transfer-time. Our experiments show that sparse subnetworks obtained through lottery ticket training do not simply overfit to particular domains, but rather reflect an inductive bias of deep neural networks that can be exploited in multiple domains.

1 Introduction

Recent research has suggested deep neural networks are dramatically over-parametrized. In natural language processing alone, most state-of-the-art neural networks have computational and memory complexities that scale with the size of the vocabulary. Practitioners have developed numerous methods to reduce the complexity of these models—either before, during, or after training—while retaining existing performance. Some of these methods include quantization (Gong et al., 2014; Hubara et al., 2017), and different flavors of pruning (Zhu and Gupta, 2017; Liu et al., 2018b; Frankle and Carbin, 2019; Gale et al., 2019).

In particular, the Lottery Ticket Hypothesis (Frankle and Carbin, 2019) proposes that small, sparse subnetworks are embedded within large, over-parametrized neural networks. When trained in isolation, these subnetworks can achieve commensurate performance using the same initialization as the original model. The lottery ticket training procedure is formalized as an iterative three-stage approach: (1) train an over-parametrized model with initial parameters θ_0; (2) prune the trained model by applying a mask $m \in \{0,1\}^{|\theta|}$ identified by a sparsification algorithm; (3) reinitialize the sparse subnetwork by resetting its non-zero weights to the initial values ($m \odot \theta_0$) and retrain it. These three stages are repeated for multiple rounds. If the final subnetwork achieves similar (or better) test performance in comparison to the original network, a winning *lottery ticket* has been identified.

Evidence of the existence of winning tickets has been empirically shown on a range of tasks, including computer vision, reinforcement learning, and natural language processing (Frankle and Carbin, 2019; Yu et al., 2019). However, the merits of lottery ticket training has recently been called into question. In particular, (1) whether keeping the same initialization (e.g., θ_0) is crucial for acquiring tickets (Liu et al., 2018b); and (2) if tickets can generalize across multiple datasets (Morcos et al., 2019).

Our paper investigates the efficacy of lottery tickets when the data distribution changes. We define multiple data domains such that their input distributions are varied. Then, we consider whether subnetworks obtained in a source domain $\mathcal{D}_s$ can be used to specify and train subnetworks in a target domain $\mathcal{D}_t$ where $s \neq t$. Inspired by Liu et al. (2018b), we also experiment with different initialization methods at transfer-time, probing at the importance of initial (source domain) values in disparate target domains. We find that subnetworks obtained through lottery ticket training do not completely overfit to particular input dis-

[*]Work done during an internship at Facebook.

tributions, showing some generalization potential when distributional shifts occur. In addition, we discover a *phase transition* point, at which subnetworks reset to their initial values show better and more stable generalization performance when transferred to an arbitrary target domain.

In summary, our contributions are (1) continuing the line of work on the Lottery Ticket Hypothesis (Frankle and Carbin, 2019), showing that tickets exist in noisy textual domains; (2) performing comprehensive experiments pointing towards the transferability of lottery tickets under distributional shifts in natural language processing; and (3) publicly releasing our code and datasets to promote further discussion on these topics[*].

2 Related Work

There is a large body of work on transfer learning for neural networks (Deng et al., 2013; Yosinski et al., 2014; Liu et al., 2017; Zoph et al., 2018; Kornblith et al., 2019). Most of these works focus on improving the transferred representation across tasks and datasets. The representation from a source dataset is fine-tuned or learned collaborately on a target dataset. In contrast, we focus on understanding whether the *architecture* can be transferred and retrained, and whether transferring the initialization is required. Our work is also related to Neural Architecture Search (NAS) (Zoph et al., 2018; Liu et al., 2018a; Elsken et al., 2018). The goal of NAS is to identify well-performing neural networks automatically. Network pruning can be viewed as a form of NAS, where the search space is the sparse topologies within the original over-parameterized network (Liu et al., 2018b; Gale et al., 2019; Frankle and Carbin, 2019).

Iterative magnitude pruning (Frankle and Carbin, 2019; Frankle et al., 2019) is a recently proposed method for finding small, sparse subnetworks from large, over-parameterized neural networks that can be trained in isolation to reach a similar (or better) test accuracy. To obtain these re-trainable sparse subnetworks, Frankle and Carbin (2019) uses an iterative pipeline that involves training a model, removing "redundant" network connections identified by a sparsification algorithm, re-training the subnetwork with the remaining connections. In particular, the experiments in Frankle and Carbin (2019) show it is critical to re-initialize the subnetworks using the *same* initial values after each round of the iterative pipeline.

[*]https://github.com/facebookresearch/pytext

However, the importance of re-using the original initialization is questioned in Liu et al. (2018b), where the authors show that competitive performance of the sparse subnetworks can be achieved with random initialization as well. Morcos et al. (2019) investigate the transferability of lottery tickets across multiple optimizers and datasets for supervised image classification, showing that tickets can indeed generalize (Morcos et al., 2019). Beyond the differences between our domain, task, and datasets, our work carries an important distinction. In Morcos et al. (2019), the authors refer to the *transfer of initialization* as both the *transfer of the sparse topologies* and the *transfer of the initial values* of the subnetworks. Therefore, it is unclear whether the *sparse topology* alone can be transferred across datasets or the topology combined with the initial values must be exploited jointly to achieve transferability. In our work, we decouple this question by investigating the influence of different initialization strategies on the sparse architecture during the process of finding the winning tickets and after the transfer to other domains.

3 Task and Datasets

Distributional Shifts Let $(x_i^s, y_i^s) \in \mathcal{X} \times \mathcal{Y}$ denote a pair of training samples from domain $\mathcal{D}_s$. Let $f(x;\theta)$ be a function (e.g., deep neural network) that maps an input from $\mathcal{X}$ to the label space $\mathcal{Y}$, parameterized by θ. In this work, the sparsity of θ is induced by the lottery ticket training process (Frankle and Carbin, 2019). To model distributional shifts, we characterize each domain $\mathcal{D}_i$ as a dataset from the Amazon Reviews corpus (McAuley and Leskovec, 2013). The differences in unigram frequencies, semantic content, and random noise mimic the type of distributional shifts that occur in machine learning.

Subword Vocabulary We ensure each domain $\mathcal{D}$ shares an identical support on $\mathcal{X}$ by encoding the inputs using a vocabulary common across all datasets. Word-level vocabularies may introduce problems during domain transfer as certain words potentially only appear within a particular domain. On the other end of the spectrum, character-level vocabularies ameliorate this issue but may not contain enough expressive power to model the data. We elect to use a subword vo-

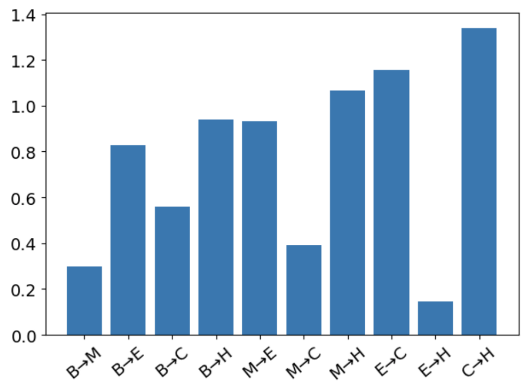

Figure 1: Jenson-Shannon Divergence scores on subword unigram distributions for each domain pair $(\mathcal{D}_i, \mathcal{D}_{i'})$. Domains include Books (B), Electronics (E), Movies (M), CDs (C), and Home (H). Values are scaled by $1e^5$ for presentation.

cabulary, balancing the out-of-vocabulary and effectiveness problems introduced by the word- and character-level vocabularies, respectively. Technical details for creating the shared subword vocabulary are presented in §4.1.

Divergence Scores Given an identical support for all data distributions, we now quantify the distributional shifts between our domains using Jenson-Shannon Divergence (JSD). JSD is a symmetric measure of similarity between two (continuous) probability distributions p and q with a proxy, averaged distribution $m = \frac{1}{2}(p+q)$:

$$\text{JSD}(p||q) = \frac{1}{2}\text{KL}(p||m) + \frac{1}{2}\text{KL}(q||m) \quad (1)$$

where $\text{KL}(p||q)$ in Eq. 1 denotes the Kullback-Leibler divergence, defined as:

$$\text{KL}(p||q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \quad (2)$$

Figure 1 displays the divergence scores between our datasets. On average, there is high disagreement with respect to the prevalence and usage of subwords in each domain, with Electronics→Home the most similar and CDs→Home the most dissimilar.

Sentiment Analysis Finally, we introduce our base task for experimentation. Our models are evaluated on a binary sentiment analysis task constructed from five categories in the Amazon Reviews corpus: books (B), electronics (E), movies (M), CDs (C), and home (H). The dataset originally provides fine-grained sentiment labels (1 through 5) so we group 1, 2 as negative and 4, 5 as positive. Following Peng et al. (2018), reviews with neutral ratings (3) are discarded. We sample 20K train, 10K validation, and 10K test samples from each category, ensuring there is an equal distribution of positive and negative reviews.

4 Methods

In this section, we discuss our technical methods. First, we describe the subword vocabulary creation process (§4.1). Second, we cover the underlying model used in the sentiment analysis task (§4.2). Third, we detail the lottery ticket training and transferring methods (§4.3).

4.1 Vocabulary

We use the SentencePiece[†] library to create a joint subword vocabulary for our datasets (Kudo and Richardson, 2018). The subword model is trained on the concatenation of all five training datasets (100K sentences) using the byte-pair encoding algorithm (Sennrich et al., 2016). We set the vocabulary size to 8K. The final character coverage is 0.9995, ensuring minimal out-of-vocabulary problems during domain transfer.

4.2 Model

We use convolutional networks (CNN) as the underlying model given their strong performance on numerous text classification tasks (Kim, 2014; Mou et al., 2016; Gehring et al., 2017). Let V and n represent the vocabulary of the corpus and maximum sequence length, respectively. Sentences are encoded as an integer sequence $t_1, \cdots, t_n$ where $t_i \in V$. The embedding layer replaces each token t_i with a vector $\mathbf{t}_i \in \mathbb{R}^d$ that serves as the corresponding d-dimensional embedding. The vectors $\mathbf{t}_1, \cdots, \mathbf{t}_n$ are concatenated row-wise to form a token embedding matrix $\mathbf{T} \in \mathbb{R}^{n \times d}$.

Our model ingests the embedding matrix $\mathbf{T}$, then performs a series of convolutions to extract salient features from the input. We define a convolutional filter $\mathbf{W} \in \mathbb{R}^{h \times d}$ where h represents the *height* of the filter. The filter is not strided, padded, or dilated, Let $\mathbf{T}[i:j] \in \mathbb{R}^{h \times d}$ represent a sub-matrix of $\mathbf{T}$ extracted from rows i through j, inclusive. The feature map $\mathbf{c} \in \mathbb{R}^{n-h+1}$ is induced by applying the filter to each possible window of h words, i.e.,

$$c_i = f\Big(\big\langle \mathbf{T}[i:i+h], \mathbf{W} \big\rangle_{\text{fro}} + b\Big) \quad (3)$$

[†]https://github.com/google/sentencepiece

for $1 \leq i \leq n - h + 1$, where $b \in \mathbb{R}$ is a bias term, f is a non-linear function, and the Frobenius inner product is denoted by $\langle \mathbf{A}, \mathbf{B} \rangle_{\text{fro}} = \sum_{i=1}^{h} \sum_{j=1}^{d} \mathbf{A}_{ij} \mathbf{B}_{ij}$. 1-max pooling (Collobert et al., 2011) is applied on $\mathbf{c}$, defined as $\hat{c} = \max\{\mathbf{c}\}$. This is performed to propagate the maximum signal throughout the network and reduce the dimensionality of the input.

The process described above creates *one* feature from *one* convolution with window h followed by a pooling operation. To extract multiple features, the model uses several convolutions with varying h to obtain features from different sized n-grams in the sequence. The convolutional (and pooled) outputs are concatenated along the channel dimension, then fed into a one-layer MLP to obtain a distribution over the c classes.

4.3 Lottery Tickets

4.3.1 Initialization

The embedding matrix is initialized from a unit Gaussian, $\mathbf{T} \sim \mathcal{N}(0, 1)$. The convolutional and MLP layers use He initialization (He et al., 2015), whose bound is defined as

$$b = \sqrt{\frac{6}{(1 + a^2) \times \text{fan\_in}}} \tag{4}$$

where a and fan_in are parameters calculated for each weight. The resulting weights have values uniformly sampled from $\mathcal{U}(-b, b)$.

4.3.2 Training

We use iterative pruning with alternating cycles of training and pruning to obtain the tickets (Han et al., 2015; Frankle and Carbin, 2019). For clarity, we define a *round* as training a network for a fixed number of epochs. We begin with a seed round r_0 where the model does not undergo any pruning, then begin to procure tickets in a series of lottery ticket training rounds.

In each successive round $r_{i>0}$, a fraction p of the weights that survived round r_{i-1} are pruned (according to a sparsification algorithm, discussed below) to obtain a smaller, sparser subnetwork; this is denoted by $f(x; m_i \odot \theta_i)$ where m_i and θ_i represent the sparse mask and weights at round r_i. The weights θ_i of this subnetwork are set according to an *initialization strategy* and the subnetwork is re-trained to convergence. We refer to the *sparsity* as the fraction of weights in the network that are exactly zero. In each round, we prune $p\%$ of

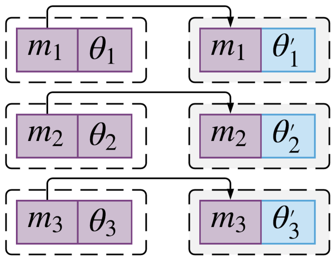

Figure 2: Visualization of the subnetwork transfer process. Purple denotes elements from the source domain, while blue denotes elements from the target domain. Tickets are composed of two elements: (1) the sparsified mask (m_i) and (2) the initial parameter values (θ_i). During transfer, we create subnetworks in the source domain with the mask borrowed from the source domain, but with potentially different parameters. We use θ'_i to denote that these parameters are set according to some *initialization strategy*, which we discuss further in our experiments (§5).

the weights in the model. Therefore, the resulting ticket has sparsity $1-(1-p\%)^{r_{total}}$, where r_{total} is the total number of lottery ticket training rounds.

Next, we discuss the sparsification algorithm used to prune weights in each round r_i. Let $\mathbf{p}_i$ denote the vectorized collection of trainable parameters in layer $i \geq 0$, with the embedding layer as layer 0. After re-training the (sub-)networks in each round, we apply the ℓ_0 projection on the parameters in each layer, i.e.

$$\underset{\mathbf{p}}{\text{argmin}}\, ||\mathbf{p} - \mathbf{p}_i||_2^2 \tag{5}$$

subject to $\text{card}(\mathbf{p}) \leq k_i$, where $\text{card}(\mathbf{p})$ denotes the number of non-zeros in $\mathbf{p}$. The optimization problem in Eq. 5 can be solved analytically by sorting the elements of $\mathbf{p}_i$ with respect to their absolute values and picking the top k_i elements with the largest magnitude (Jain et al., 2017; Zhu and Gupta, 2017). We use the sparsity hyperparameter p introduced above to decide k_i for each layer. Let $\text{len}(\mathbf{p}_i)$ denote the total number of trainable parameters in layer i. We set $k_i = p\% \times \text{len}(\mathbf{p}_i)$ for each layer. In accordance with our training procedure, once a weight is pruned, it is no longer a trainable parameter; hence, $\text{len}(\mathbf{p}_i)$ is strictly decreasing after each round.

4.3.3 Transferring

The lottery ticket training procedure outlined in §4.3.2 yields a batch of subnetworks $f(x^s; m_1 \odot$

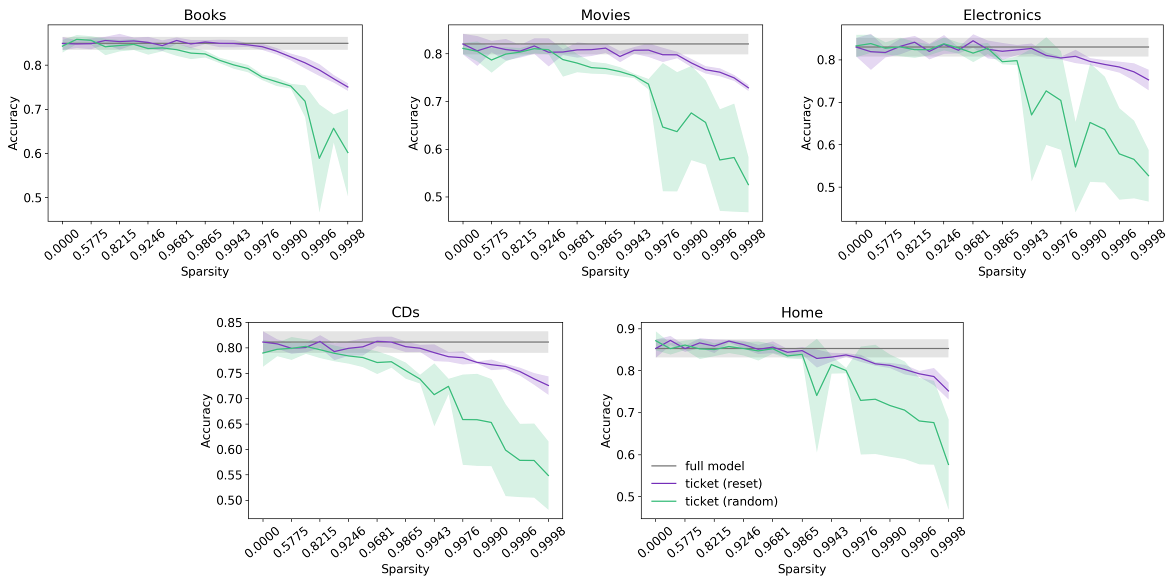

Figure 3: Results obtaining lottery tickets on the Books, Movies, Electronics, CDs, and Home categories of the Amazon Reviews dataset (McAuley and Leskovec, 2013). Experiments are repeated five times, where the solid lines represent the mean and shaded regions represent the standard deviation. Note that the x-axis ticks are *not* uniformly spaced.

$\theta), \cdots, f(x^s; m_n \odot \theta)$ where x^s represents the inputs from a *source* domain $\mathcal{D}_s$ and m_i represents the sparse mask used to prune weights at round r_i. During transfer, we construct a new batch of subnetworks $f(x^t; m_1 \odot \theta'), \cdots, f(x^t; m_n \odot \theta')$ to be evaluated on inputs from a (non-identical) *target* domain $\mathcal{D}_t$ with masks derived from the *source* domain. The change in parameter notation ($\theta \rightarrow \theta'$) implies that the subnetworks evaluated in a disparate domain can potentially use a different *transfer* initialization strategy. We clarify this process in Figure 2. In contrast, Morcos et al. (2019) transfers the entire ticket (sparse masks and initial values) to the target domain. Finally, using the new batch of subnetworks, we evaluate each subnetwork $f(x^t; m_i \odot \theta')$ in the target domain for r_{total} rounds. Unlike the canonical ticket training rounds, we do not (additionally) sparsify the subnetworks during transfer. All in all, our transfer task is designed to answer the following question: can the *sparse masks* found in a source domain using lottery ticket training (§4.3) be transferred to a target domain with *different initialization strategies* to match the performance of a ticket obtained in same target domain?

5 Experiments

5.1 Settings

Our CNN uses three filters ($h \in [3, 4, 5]$), each with 127 channels, and ReLU activation (Nair and Hinton, 2010). We fix the maximum sequence length to 500 subwords. The embeddings are 417-dimensional and trained alongside the model. We opt not to use pre-trained embeddings to ensure the generalizability of our results. Additionally, we regularize the embeddings with dropout (Srivastava et al., 2014), $p = 0.285$. The MLP contains one hidden layer with a dimension of 117. Hyperparameters were discovered using Bayesian hyperparameter optimization (Snoek et al., 2012) on the Books validation set. The models are trained with a batch size of 32 for a maximum of 15 epochs. Early stopping is used to save iterative model versions that perform well on a development set. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $1e^{-3}$ and ℓ_2 regularization with a weight of $1e^{-5}$.

5.2 Obtaining Tickets

First, we use the lottery ticket training procedure outlined in §4.3.2 to obtain tickets for our five datasets with $p = 35\%$ and $r_{total} = 20$. We compare the test performance of the subnetworks using the following baselines:

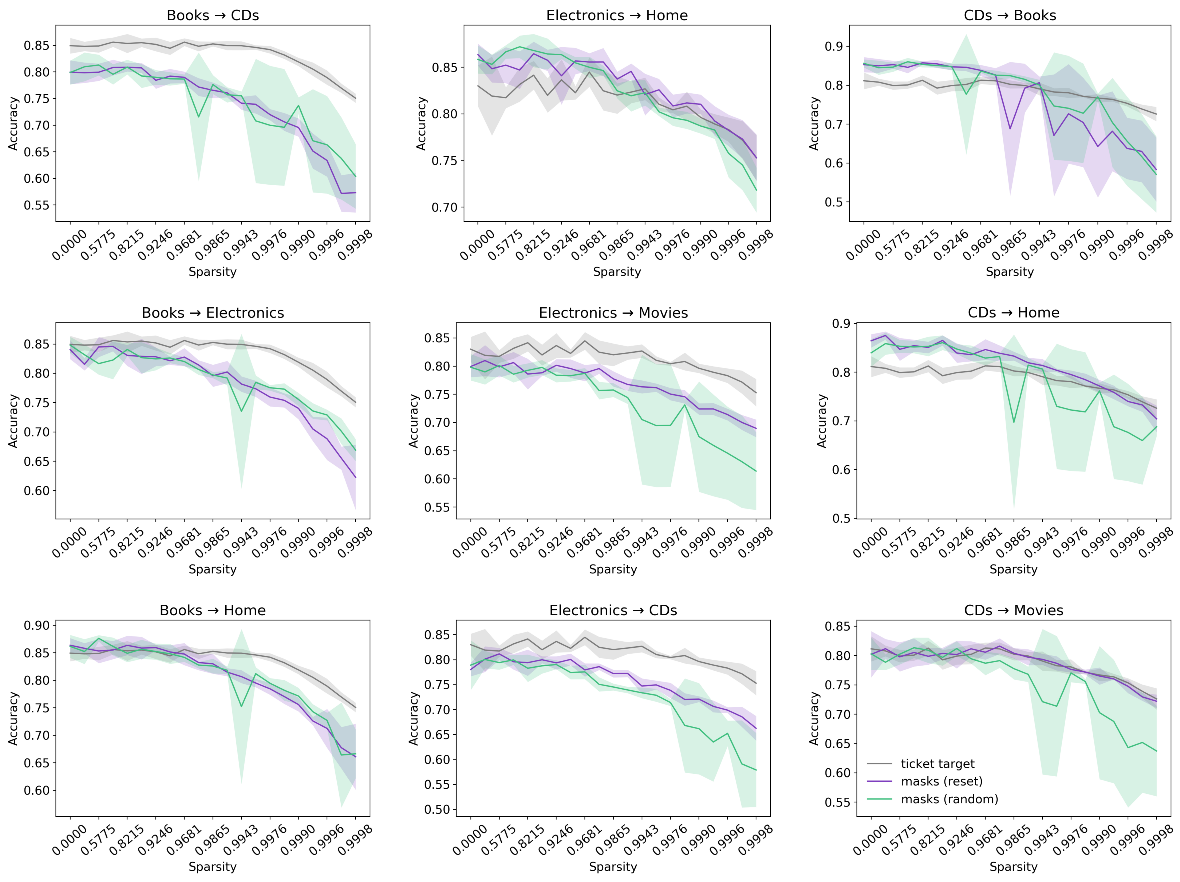

Figure 4: Results transferring lottery tickets on nine transfer tasks constructed from the five categories of the Amazon Reviews dataset (McAuley and Leskovec, 2013). Experiments are repeated five times, where the solid lines represent the mean and shaded regions represent the standard deviation. Note that the x-axis ticks are *not* uniformly spaced.

- FULL-MODEL: This baseline evaluates the performance of the original network *without* any pruning. In other words, we train a model for a seed round r_0, then record its performance.

- TICKET-RESET: The values of the subnetwork are reset to their *original values* before training. This initialization strategy was used in the earliest formation of the Lottery Ticket Hypothesis (Frankle and Carbin, 2019).

- TICKET-RANDOM: The values of the subnetwork are reset to *random values* drawn from the initialization distribution(s) of the original network. We sample weights from the distributions outlined in §4.3.1 to initialize the subnetworks.

The results are shown in Figure 3. For all datasets, TICKET-RESET shows the best performance, notably outperforming FULL-MODEL in early stages of sparsification (0-90%) for the Books, Electronics, and Home datasets. This demonstrates that deep neural networks—especially those for sentiment analysis—are highly over-parameterized, and the sparsity induced by lottery ticket training can help to increase performance. This observation is consistent with Louizos et al. (2018), which also showed sparse networks fashion a regularization effect that results in better generalization performance. In addition, we observe that TICKET-RESET and TICKET-RANDOM have similar test performance until about 96% sparsity. This casts some doubt around whether the initial values truly matter for sparse models as the randomly sampled values seem to fit sparse masks well.

However, a *phase transition* occurs in the high sparsity regime, where the differences between TICKET-RESET and TICKET-RANDOM are significantly enlarged. The performance of TICKET-

RANDOM becomes highly unstable and drops off much faster than TICKET-RESET after 96% sparsity. In contrast, TICKET-RESET remains relatively stable—even with sparsity levels over 99.9%—pointing towards the enigmatic importance of original values in extreme levels of sparsity.

5.3 Transferring Tickets

Next, we use the lottery ticket transferring procedure outlined in §4.3 to transfer (obtained) subnetworks from a *source* domain to a non-identical *target* domain. Identical to the previous experiment, we use $r_{total} = 20$. We compare the test performance of the *transferred* subnetworks using the following baselines:

- TICKET-TARGET: This baseline is comprised of the subnetworks obtained in the target domain using lottery ticket training. We borrow the values for this baseline (without modification) from the TICKET-RESET subnetworks shown in Figure 3, albeit from the domain of interest.
- MASKS-RESET: Under this initialization strategy, the masks obtained in the source domain is used on the target domain and the subnetwork is trained from the *same* initial values as in the source domain.
- MASKS-RANDOM: Under this initialization strategy, *only* the masks are used from the subnetwork obtained in the source domain. The parameters are randomly initialized from the distributions outlined in §4.3.1 before training on the target domain.

The results are shown in Figure 4. Both MASKS-RESET and MASKS-RANDOM show signs of generalization in the early stages of sparsification. Most notably, subnetworks obtained in the CDs domain are extremely robust; both the MASKS-RESET and MASKS-RANDOM results show stronger performance than TICKET-TARGET, even in sparsity levels over 99%. This is relatively surprising as the FULL-MODEL in §5.2 achieved the worst performance in the CDs domain. Further inspection of representations learned in this domain will be required to understand its strong ticket performance, which may or may not be a coincidence.

We see a 3-5% dropoff in performance (up to 90% sparsity) from tickets identified from the Books and Electronics tasks after transferring. These results together imply that tickets are not completely immune to distributional shifts, although the degradation in test accuracy is not substantial until reaching high sparsity. Nevertheless, we notice the accuracies of MASKS-RESET and MASKS-RANDOM stay relatively stable from 0-90% sparsity; they only begin to steadily decline after this point.

Finally, we compare the performance of MASKS-RESET and MASKS-RANDOM. In the Books tasks, MASKS-RANDOM performs better overall in comparison to MASKS-RESET. Its performance is slightly worse in the Electronics and CDs tasks, although it is relatively comparable to MASKS-RESET up to 96%. Similar to the results in §5.2, we notice a *phase transition* point where the initial values (e.g., MASKS-RESET) play a much bigger role in maintaining stability and performance in the deeper stages of sparsification.

6 Discussion

In this section, we briefly recap our findings, highlighting key points observed through our ticket procuring and transfer experiments. For each section, we also touch on areas for future work.

Evidence of transferability of winning tickets in natural language processing. Our experiments show that "winning tickets" can indeed be identified in a sentiment task formulated from noisy, user-generated datasets. Moreover, the "winning tickets", up to extreme level of sparsity (e.g., 90%), can be transferred across domains without much loss in accuracy. The fact that tickets can be obtained in noisy environments shows its prominence across multiple data sources. However, our work only considers a binary sentiment analysis task. Future work can explore other tasks such as multi-class text classification, language modeling, and machine translation.

Randomly initialized tickets are strong baselines. Consistent with the observations in Liu et al. (2018b), initializing tickets to their *original values* before training is not necessarily required for strong performance. In our experiments, we show that in high sparsity conditions (up to 90%), there is no noticeable difference between the performance of the *originally* and *randomly* initialized subnetworks. Although the sparse masks build on top of each other from round r_i to r_{i+1}, randomly initialized subnetworks are still able to settle in a local minima with comparable performance to that of the originally initialized subnetworks. However, our work fixes the optimizer and learning rate across experiments. It may be possible that randomly initialized subnetworks using varying optimization reach better minima.

A *phase transition* point largely influences ticket performance. As alluded to above, there is almost no difference in performance when considering originally and randomly initialized subnetworks. However, our experiments point towards a crucial turning point—the *phase transition*—in which the initialization begins to matter. In particular, especially in extreme levels of sparsity (e.g., 99.99%) originally initialized networks exhibit less variance than randomly initialized tickets in test accuracy. However, the specific sparsity at which the phase transition happens is dataset-dependent. Understanding why this occurs and its relation with other models, datasets, and optimization algorithms can further unveil and explain the phenomena behind lottery tickets.

7 Applications in Federated Learning

Federated learning is a scenario where a centralized model is trained over decentralized data, distributed across millions (if not billions) of clients (e.g., electronic devices) (Konen et al., 2016; Bonawitz et al., 2019). Crucially, the clients are not allowed to exchange *data* with the central server or each other. Instead, each client can fine-tune a model for a couple of iterations on their own data, then send their (encrypted) parameters or gradients to a server for aggregation. This "collaborative learning" setup effectively maintains a level of user privacy by ensuring the data always stays on-device. However, this poses several challenges for optimization; as the centralized server does not have access to the data distribution of each client, any neural architecture selection has to be done on either (a) a *different* data source the server has access to or (b) on each individual client. Since (b) is generally quite expensive, the server usually maintains some seed data, as alluded to in (a).

With the transferability of lottery tickets, the server can procure lottery tickets on server-accessible data, then retrain the tickets on client data under the federated learning framework. While there may be a large performance drop when transferring *extremely* sparse networks, our results show that clients can still re-train *moderately* sparse networks with commensurate performance. We believe that this "sparsify and transfer" procedure has two immediate benefits: (1) past work—including the original incarnation of the lottery ticket hypothesis—has shown that sparse networks can be, under certain conditions, easier to optimize (Frankle and Carbin, 2019; Morcos et al., 2019; Gale et al., 2019); and (2) sparser subnetworks have significantly less capacity than their large, over-parameterized counterparts, which can alleviate client-server communication costs (e.g., model uploading and downloading) (Konen et al., 2016; Sattler et al., 2019).

8 Conclusion

The Lottery Ticket Hypothesis (Frankle and Carbin, 2019) posits that large, over-parameterized networks contain small, sparse subnetworks that can be re-trained in isolation with commensurate test performance. In this paper, we examine whether these tickets are robust against distributional shifts. In particular, we set up domain transfer tasks with the Amazon Reviews dataset (McAuley and Leskovec, 2013) to obtain tickets in a *source* domain and transfer them in a disparate *target* domain. Moreover, we experiment with the *transfer* initialization of the networks, determining if resetting to initial values (obtained in the source domain) are required for strong performance in the target domain. Our experiments show that tickets (under several initialization strategies) can be transferred across different text domains without much loss up to a very high level of sparsity.

In addition, there is a lot of debate on whether initial value resetting is critical to achieve commensurate test performance. While Frankle and Carbin (2019); Frankle et al. (2019) present evidence supporting the importance of resetting, Gale et al. (2019); Liu et al. (2018b) show that sparse re-trainable subnetworks can be found independent of resetting. Our experiments show that this is *not* a yes or no question. Specifically, we show there is a *phase transition* related to sparsity. Resetting is not critical before extreme levels of sparsity (i.e., below 99%), but the effect of resetting is magnified in high sparsity regimes. Finally, we demonstrate the practical applications of our results in federated learning.

Acknowledgments

Thanks to Veselin Stoyanov and our anonymous reviewers for their helpful comments.

References

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Jun Deng, Zixing Zhang, Erik Marchi, and Björn Schuller. 2013. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 511–516. IEEE.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2018. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.

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

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. 2019. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611v2*.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.

Prateek Jain, Purushottam Kar, et al. 2017. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–336.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jakub Konen, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. 2019. Do better imagenet models transfer better? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2661–2671.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018a. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Jiaming Liu, Yali Wang, and Yu Qiao. 2017. Sparse deep transfer learning for convolutional neural network. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018b. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.

Ari S Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. 2019. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *arXiv preprint arXiv:1906.02773*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Minlong Peng, Qi Zhang, Yu-gang Jiang, and Xuanjing Huang. 2018. Cross-domain sentiment classification with target domain specific information. In *56th Annual Meeting of the Association for Computational Linguistics (Long Papers).*

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Robust and communication-efficient federated learning from non-iid data. *arXiv preprint arXiv:1903.02891.*

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2019. Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP. *CoRR*, abs/1906.02768.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878.*

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.