HYPERPRUNING: EFFICIENT PRUNING THROUGH LYAPUNOV METRIC HYPERSEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

A variety of pruning methods have been introduced for over-parameterized recurrent neural networks to improve efficiency in terms of power and storage requirements. With the advance in pruning methods and their variety, a new problem of 'hyperpruning' is becoming apparent: finding a suitable pruning method with optimal hyperparameter configuration for a particular task. Such search is different from the standard hyperparameter search, where the accuracy of the optimal configuration is unknown. In the context of network pruning, the accuracy of the non-pruned model sets the target for the accuracy of the pruned model. Thereby, the goal of hyperpruning is to reach or even surpass this target. It is critical to develop efficient strategies for hyperpruning, since direct search through pruned variants would require time-consuming training without guarantees for improved performance. In this work, we propose an efficient and first-of-its-kind hyperpruning approach. Specifically, we introduce an efficient search for pruning RNN that employs the Lyapunov Spectrum (LS)-based metric. The metric estimates, early in training, the accuracy that pruned variants will achieve after extensive training, and provides a space to compare the variants with the non-pruned model. Our experiments on stacked-LSTM and RHN language models trained on the Penn TreeBank dataset show that the LS-based metric is critical for hyperpruning, since it is robust and predictive of the performance that the model will achieve after full training, in contrast to the metric directly based on loss. LS-based early prediction leads us to propose a novel method called LSH that minimizes search time by an order of magnitude compared to loss-based full training search. We show that for a given budget of training epochs and desired pruning ratio, LSH obtains more optimal variants than standard search methods. Furthermore, as a result of the search, the LSH identifies variants that outperform the perplexity of pruned state-of-the-art language models on the Penn TreeBank dataset and significantly outperform the dense model.

1 INTRODUCTION

Over the last decade, the performance of Recurrent Neural Networks (RNN) has been significantly enhanced in various applications such as action recognition (59), video summarization (72) and voice conversion (25). In particular, some RNN variants such as LSTM (23; 41) and RHN (75) shine in various NLP application ranging from machine translation (69) to language modeling (27). However, the architecturally inherent computational demanding of RNN which is linear to input sequence length and quadratic to model size slows down RNN in both training and inference stages, and therefore hinders it from being deployed on resource-limited devices such as mobile. Multiple methods have been proposed to alleviate this problem, including network quantization (22; 19) and weight sharing (62). Among these approaches is network pruning which is advantageous in achieving a sparse model and thus requires less computational resources. A particularly notable pruning approach is dense-to-sparse, where the network is gradually pruned from a full dense model. While the inference time of the pruned network will eventually decrease, the training time remains the same or even longer than the time of training a dense model to the extent of days or weeks. The Dynamic Sparse Training approach (DST) was introduced recently to meet the rising demand of reducing training time. Instead of starting from a dense model, DST is a sparse-to-sparse pruning method that starts with a sparse model and maintains the number of non-zero parameters over the entire training

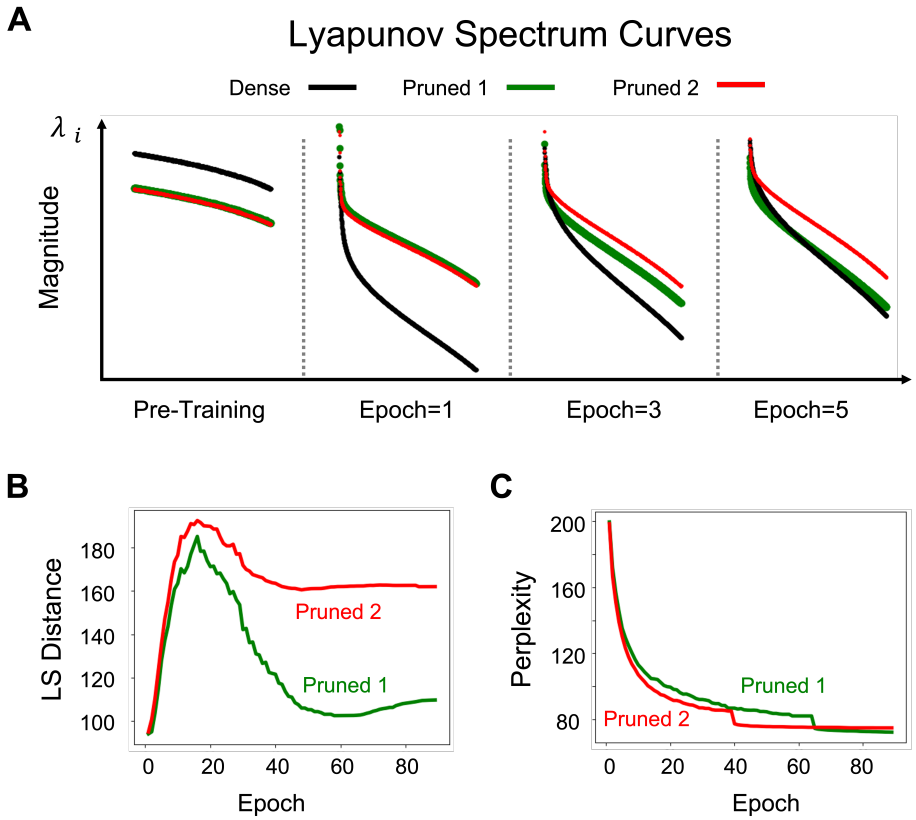

Figure 1: (A) Lyapunov Spectra of Dense (purple), and two pruned networks (green and red) at pre-training, epoch 1, 3, and 5. (B) The L_2 distance of two candidates (green and red) to the dense reference in LS space over training, respectively. (C) Perplexity curves for two pruned candidate models during training.

to improve not only the inference speed but also the training speed. DST involves three main steps for every pruning procedure: weight removal, growth, and redistribution. Different salience can be used to decide how each step is conducted, and there is no universal salience applicable to all tasks, as each has its intended applications.

This unique correlation between application and salience prevents generalizing a standard rule over all applications. Therefore, for a certain application, the salience characterizing pruning methods becomes a new hyperparameter space to search from. We term this type of hyperparameter search as 'Hyperpruning'. Hyperpruning concerns selecting the configuration variant of the pruning method along with other hyperparameters for a particular task. Specifically, hyperpruning requires searching over methodological and non-methodological hyperparameters. Methodological hyperparameters, including the aforementioned salience, identify the pruning method, and non-methodological hyperparameters are independent of pruning methods and applicable across the methods. One unique feature of hyperpruning, due to the pruning setup, is the awareness of estimated target accuracy, i.e., the optimal accuracy of its non-pruned/dense counterpart is available and can serve as a loose upper bound for the pruned network to guide the hyperpruning process while this knowledge of a target accuracy does not typically exist in a general hyperparameter search task. However, even with this knowledge, searching through all pruning methods that individually require extensive training, is a time-consuming task, and there is no guarantee for achieving a better configuration after training many bad configurations.

Many hyperparameter optimization algorithms accelerate the search process by implementing a less time-consuming metric to evaluate the pruning configuration variant. In particular, such a metric will estimate the performance of a considered configuration without proceeding with full training.

This search can be further sped up by training more promising configurations which are generated according to the history of trained configurations and their corresponding accuracies. A commonly used metric is the loss curve from early training episodes. However, in many scenarios, such a direct metric is misleading, and it is rarely guaranteed that models that perform well at the beginning of training will reach the desired accuracy at the end, as illustrated in Fig. 1-C. Therefore, finding an effective and reliable metric is essential to hyperpruning.

In the context of RNN, previous work has shown that these networks can be treated as a dynamical system, and their dynamic flow can be analyzed using Lyapunov Spectrum (LS) which measures the expansion and contraction ratio of the hidden states over time. It was also proposed that LS could be a potential indicator of the trainability of the network and the accuracy that it could reach (64). Some recent work suggests that an AutoEncoder trained on the LS of slightly trained-RNN samples with random hyperparameter configurations provides a low-dimensional embedding space from which RNN variants are arranged according to their training accuracy (65). Indeed, this arrangement based on accuracy is also observed in the network pruning setup during early training, as shown in Fig. 1-A. Between those two pruned networks, pruned 1 eventually achieves better performance than pruned 2 as shown in Fig.1-C. At the pre-training stage, the LS of the dense network (purple) is apart from pruned 1 (green) and pruned 2 (red) which overlap with each other. Similarly, at epoch 1, the LS of pruned 1 and pruned 2 are close to each other but isolated from the dense one. However, at epoch 3, pruned 1 starts to move away from pruned 2 and get closer to the dense. At epoch 5, the LS of pruned 1 and dense networks are overlapping while pruned 2 is drifting away from them. This closeness between the LS of dense and pruned 1 network is consistent over the entire training (over 100 epochs) no matter what the current accuracy is, as shown in Fig. 1-B.

Inspired by these observations that the distance in LS space between the dense and the pruned variants at early epoch of training is indicative of the closeness of the variants to the dense model, we propose to utilize LS and its embedding (LS Space) as a metric to guide hyperpruning. This leads us in proposing a novel and effective algorithm, called *LS*-based *H*yperpruning (***LSH***), which utilizes the distance in LS space (LS-based metric) to find optimal pruning variants at a fraction of the cost required for training them. We summarize the main contributions of our work below:

1. We propose a novel metric for hyperpruning. The metric is based on Lyapunov Spectrum (LS) and is capable of estimating distance in accuracy between variants of the pruned model with different methodological and non-methodological hyperparameters.
2. We thereby propose a novel algorithm called LSH that utilizes the LS-based metric as an early stop-criterion and allows it to sift through pruned variants with a significantly smaller number of training epochs than utilizing the loss-based metric for the same search.
3. Our experiments on stacked LSTM and RHN language model trained with Penn TreeBank Dataset shows that LSH achieves more accurate pruned models for a given pruning ratio, with an order of magnitude less time spent on training variants.

2 Related Work

Network Pruning The idea of network pruning is not new. In its early applications, it was proposed for the improvement of inference speed of pretrained dense networks by iteratively pruning and fine-tuning until reaching the target sparsity (28; 48; 47; 32; 21). Following this dense-to-spare pipeline, magnitude-based pruning on a pretrained model paired with fine-tuning was applied to modern convolutional networks and while pruned showed only a marginal decrease in performance (20; 18). Gradual Magnitude Pruning (GMP) (49; 74; 17) removes the requirement of a pretrained model by gradually pruning over time and has been implemented on a broader range of network architectures, such as RNN, deep CNN, LSTM, transformer, etc. Later approaches proposed generalizing the pruning process by learning the weight and the pruning mask jointly through a unified optimization or differential reparameterization function (38; 30; 37) instead of setting the threshold and learning the weights separately. Indeed, as claimed in Lottery Ticket Hypothesis (LTH) (15), the value of network pruning is beyond increasing the inference speed and it is possible to further optimize accuracy through pruning and achieve a pruned network through iterative magnitude-based pruning that outperforms its dense non-pruned counterpart. This lottery ticket can even be identified during early training to save training cost (71). In addition to direct pruning on weights, pruning can also be realized indirectly through L_0 (40), L_1 regularization (68), and Variational Dropout (44).

Nevertheless, according to (17), those methods fail to match the performance of magnitude-based pruning on large-scale learning tasks. While gradual pruning approaches succeed in achieving faster inference time, their training time is similar to or even larger than training a dense network. To improve the training efficiency and avoid complex pruning schedules, it has been proposed to prune at initialization according to some salience criteria and maintain the pruning mask throughout training (34; 33; 66; 60), These methods are advantageous but fail in extreme sparsity cases due to layer-collapse where an entire layer is pruned, making the network untrainable (33), and the training efficiency typically comes at the price of the performance (66).

Recent approaches proposed Dynamic Sparse Training (DST), a sparse-to-sparse pruning concept, to achieve inference and training efficiency without sacrificing accuracy. DST maintains the total number of non-zeros during training to keep the efficiency and alters the position of non-zeros through a prune-regrow process to make the model more flexible. This idea was first introduced in Deep-R (3), which rewires the network from a posterior. Inspired by biological neural networks, SET (43) simplified this process by replacing it with magnitude-based pruning and random weight growing to maintain the sparsity during training. Starting from a spare seed network, NeST (9) prunes and grows both neurons and weight according to magnitude and gradient, respectively. DSR (46) introduced non-uniform sparsity on different layers instead of uniform sparsity in SET. These methods, however, are still not comparable to their dense-to-sparse counterpart. SNFS (11) proposed to use momentum for weight growing which achieves better performance than random growth and matches or even outperforms the dense-to-sparse method but at the price of computational efficiency (11). To tackle the inefficiency, RigL (14) adapts lazy gradient calculation into magnitude-based pruning, and Top-KAST (29) further releases the computation burden by not requiring a forward pass with the dense parameter or calculating the dense gradient. Recently, a DST pruning method called Selfish-RNN has been proposed specifically for RNN. It showed a significant improvement over the performance of sparse RNN via non-uniform weight redistribution across gates and SNT-ASGD (39). The importance of parameter allocation among different layer types is also found in (30; 16). For a more comprehensive survey about network pruning, readers can refer to (24; 67).

Hyperparameter Search Multiple Hyperparameter Optimization (HPO) algorithms have been proposed to find optimal hyperparameter configurations for a specific task. Most HPO algorithms iteratively propose configurations and then evaluate the performance of these configurations. Therefore, HPO can be categorized by the way they propose new configurations and how they evaluate configurations. In proposing configurations, grid search and manual search are the most intuitive HPO, however also inefficient. Instead, random search finds comparable or better models within a limited time (5). Unlike these approaches, more advanced HPO algorithms store proposed configurations and their performance in an archive. They then decide between exploitation and exploration, i.e., exploiting the current best configuration (local search) or exploring a new random configuration (random search). Bayesian Optimization, such as TPE, uses the archive to fit a surrogate model and then adaptively finds configurations (57; 26; 4). Given this extra information from the archive, Bayesian Optimization empirically outperforms a plain random search (61; 12; 58). Another variation of HPO is multifidelity search which focuses on accelerating configuration evaluation, i.e., fast evaluation is used to infer the configuration performance (6). This approach aims to find promising candidates and allocate more resources to them. Typically, there is a trade-off between computational cost and evaluation accuracy, i.e., the more computation spent on evaluating a configuration, the more accurate the evaluation is. Therefore, one common practice is to remove less promising candidates with increasing computation resources successively. This approach can be incorporated into random search or other HPO to explore a more expansive searching space within the same resource budget (36). Searching over a larger space is necessary for complex architectures, such as sparse stacked LSTM and RHN, due to many possible hyperparameter configurations. Therefore, a fast and effective configuration evaluation method becomes essential. Here we show that Lyapunov Spectrum (LS) is a candidate for such a fast metric.

Lyapunov Spectrum Treating RNN as dynamical systems and predicting their long-term performance is an emerging topic in deep learning research (8; 73; 63; 53). Lyapunov Spectrum (LS), formed by Lyapunov Exponents, is targeted to identify nonlinear dynamical systems' characteristics (54; 51). LS was initially used to provide insight into autonomous neural network models (45; 13), but RNNs are typically non-autonomous due to external inputs. However, as long as input sequences are sampled from a stationary distribution, LS can be generalized to a non-autonomous dynamical system such as RNN according to the random dynamical theory (2). Another property of LS is that if RNN sequence

is long enough, the resulting LS converges to the same curve, as proved by Oseledets theorem (55; 50). These results indicate that knowledge from LS could be indicative of network dynamics. In (35; 52; 31), the correlation between the largest Lyapunov exponent sign and the existence of chaos in the network has been studied. In addition, research succeeded in connecting LS features, such as zeros, negative, mean, and variance, and concepts in dynamical systems, such as quasi-periodic orbits, fixed point attractor, heterogeneity, and rate of contraction (10; 1; 56; 7; 70). While each feature seems to extract knowledge about the network, their correlation with network accuracy is unclear (63). With the efficient LS computing algorithm proposed in (63), the correlation between LS and network performance was found via a separate AutoEncoder network (65) such that this AutoEncoder can be used to predict the performance of random hyperparameter configurations. Inspired by this work, we propose an LS-based metric that is used to exclude non-promising candidates.

3 METHODS

Starting with a set of configuration candidates, LS-based Hyperpruning (LSH) effectively eliminates non-promising candidates from the set based on their closeness to the dense reference in the LS space. It then generates new candidates given the remaining candidates in the set and inserts them to the set. After iterating this elimination and insertion step several times, the most promising candidates are selected and most of the training resources are allocated to them via extensive training. The LSH algorithm is illustrated in Fig. 2.

Lyapunov Spectrum Computation We adapt the algorithm from (65), which computes LS by monitoring the contraction and expansion history of the network over the entire time sequence. Specifically, it first randomly selects a batch $\mathcal{X}$ of K samples with sequence length T from the validation set $\mathcal{D}_{val}$, i.e., $\mathcal{X} \equiv \{\mathbf{x}^k = (x^{1,k}, x^{2,k}, \ldots, x^{T,k}), \mathbf{x}^k \in \mathcal{D}_{val}, k = 1, 2, \ldots, K\}$, where $x^{t,k}$ is the t-th time step of k-th sample in $\mathcal{X}$. A vector of zeros $\mathbf{0}$ and an identity matrix $\mathcal{I}$ are initialized as the original hidden state h^0 and orthogonal representation base $\mathcal{B}^0$. Hidden state h^t tracks the evolution of network and orthogonal representation base $\mathcal{B}^t$ captures contraction and expansion ratio of each direction in the hidden state till time step t. At each time step, h^t and $\mathcal{B}^t$ are updated via Eq. 1-3.

$$h^{t,k} = f(h^{t-1,k}, x^{t,k}) \tag{1}$$

$$J^{t,k} = \frac{\partial h^{t,k}}{\partial h^{t-1,k}} \tag{2}$$

$$B^{t,k}, R^{t,k} = qr(J^{t,k} \cdot B^{t-1,k}) \tag{3}$$

where f is the function of the network, J is the Jacobian matrix between hidden states, and qr means QR decomposition. The contraction and the expansion ratio at time step t is incorporated into the i-th vector of $\mathcal{B}^t$ which the i-th diagonal element of $\mathcal{R}^t$, r_i^t, can measure. The i-th LS, λ_i is then calculated by taking the average over the entire time sequence T and K samples.

$$\lambda_i = \frac{1}{K}\frac{1}{T}\sum_{k=1}^{K}\sum_{t=1}^{T} log(r_i^{t,k}) \tag{4}$$

LS-based Hyperpruning (LSH) Starting with a random candidate set $\mathcal{T}$ where each candidate is a hyperparameter configuration of a pruned network, the goal of hyperpruning is to find the best configuration. The procedure of LSH is guided by a dense network $\hat{t}$ since it provides a target performance for the pruned model to reach or even surpass. The selection is realized by iteratively executing LS-based elimination and adaptive insertion. Specifically, for the LS-based elimination step, each candidate $t^i \in \mathcal{T}$ is trained for e_0 epochs, and after each epoch of training, their LS is computed according to Eq. 4 and denoted as LS_j^i where i and j stands for the i-th candidate and j-th epoch, respectively. LS from all epochs of t^i is grouped as LS^i, i.e., $LS^i \equiv \{LS_j^i\}_{j=0}^{e_0}$. Similarly, all LS from the dense model till epoch e_0 are denoted as $\hat{LS}$. We then project $\hat{LS}$ and LS^i to an embedding space and calculate the distance between the last epoch of t^i and $\hat{t}$ as the indicator of their closeness, as shown in Eq. 5-6.

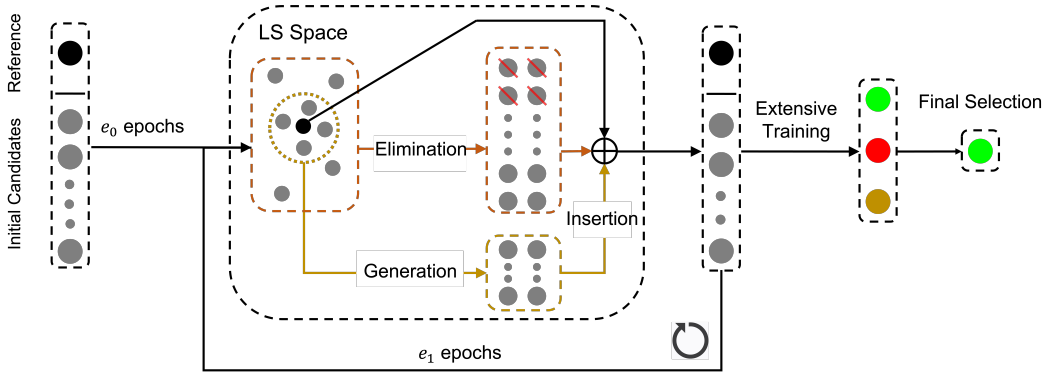

Figure 2: LS-based Hyperpruning: The process starts with an initial candidate set of hyperparameter configurations (gray circles) and a dense reference model (blue circle) and iteratively executes elimination and insertion. Elimination: The reference model and each candidate is trained for e epochs and projected to the LS space. Candidates are ranked and eliminated depending on their distance to the reference in the LS space. Insertion: New candidates are then adaptively generated given the remaining candidate in the set and inserted to the set. After several rounds of elimination and insertion, the remaining candidates in the set are extensively trained and the best one is picked as the final selection.

$$[\hat{x}_0, \ldots, \hat{x}_E, x_0^i, \ldots, x_E^i] = embedding([\hat{LS}, LS^i]) \tag{5}$$

$$s = distance(\hat{x}_E, x_E^i) \tag{6}$$

where E is the last training epoch so far. In our work, we choose PCA to construct the embedding space and project LS to a 2d space, called LS Space. We use L_2 distance to measure the closeness of networks in this LS space. The distance s between t^i and $\hat{t}$ is then used to rank the candidates in the $\mathcal{T}$ and half of the candidates with longer distances are eliminated. Even though LS-based elimination can efficiently find the best configuration in $\mathcal{T}$, it does not guarantee to find the best configuration for this task since the best configuration does not necessarily exist in the initial set. Therefore, we need to adaptively generate new candidates based on the remaining candidates in $\mathcal{T}$ and insert them to $\mathcal{T}$. The intuition is as more promising candidates are in $\mathcal{T}$, the chance of finding the best configuration goes higher. Typically, this new candidate generation and insertion happen after the LS-based elimination. After the elimination, given the remaining n candidates in $\mathcal{T}$, the knowledge of their configurations and distances s serves as an initial history/archive for sequential model-based optimization methods. We choose Tree Parzen Estimators (TPE) to demonstrate the process. Starting from this initial knowledge, TPE builds a surrogate model and generates new configuration candidates via Expected Improvement. $\frac{n}{2}$ new candidates are generated and then added to candidate set $\mathcal{T}$. This uneven number of elimination and insertion candidates from and to $\mathcal{T}$ monotonically decreases the number of candidates in $\mathcal{T}$ so that eventually only the most promising candidates are left in $\mathcal{T}$. The elimination/insertion schedule is controlled by the starting epoch e_0 and incremental epoch e_i which decides when to execute the first and future elimination, respectively. As a result, they manipulate the required time budget for the candidate selection process.

After candidate selection for m epochs, those remaining candidates are then extensively trained for P epochs where $P >> m$, and the best candidate is selected as the final selection.

4 Experiments

We perform multiple experiments comparing *LS-based* metric (LSH) and *Loss-based* metric as early stopping criteria for hyperparameter optimization algorithms in hyperpruning to demonstrate the effectiveness of using LS-based metric. The exact same size of the initial candidates set and elimination/insertion schedule is used for a fair comparison, i.e., the number of initial candidates n, the starting epoch e_0, and incremental epoch e_i are kept the same. This results in the same

resource budget for both methods since the resource budget only depends on the candidate set size and elimination/insertion schedule. We show the generality of our method by applying it to stacked LSTM and RHN trained with Penn TreeBank dataset (42) for a language modeling task. For each experiment, we follow the setup in a SOTA RNN pruning work (39). We also show the robustness of LSH by conducting experiments on stacked LSTM with different pruning ratios. Furthermore, we demonstrate the time efficiency of LSH by comparing the time difference between LSH and loss-based metric to achieve a target accuracy. The training and LS computation are conducted on an NVIDIA RTX 2080 Ti. More hyperparameters about the network architecture can be found in Appendix.

Table 1: The validation and test perplexity of stacked LSTM and RHN on PTB dataset for a language modeling task (lower perplexity means better performance). LSH applied with Grid Search (GS), Tree Parzen Estimator (TPE), and Adaptive TPE (ATPE). Vanilla GS, TPE, and ATPE use loss as the metric. DSR, SNSF, SET, RigL, GMP, Dense, and Selfish-RNN data are from (39).

Method	Stacked-LSTM			RHN RHN		
	Validation	Test	Parameters	Test	Validation	Parameters
DSR	90.0	88.2	21.8M	63.2	65.4	11.1M
SNSF	88.3	86.3	21.8M	71.0	74.0	11.1M
SET	87.3	85.8	21.8M	61.1	63.7	11.1M
RigL	78.3	75.9	21.8M	62.5	64.8	11.1M
GMP	76.8	74.8	21.8M	64.0	65.6	11.1M
Dense	74.5	72.4	66.0M	61.8	63.4	23.5M
GS	74.2	72.5	21.8M	60.4	62.1	11.1M
APTE	74.3	72.7	21.8M	61.6	63.4	11.1M
TPE	74.4	72.7	21.8M	60.2	62.1	11.1M
Selfish RNN	73.8	71.7	21.8M	60.4	62.1	11.1M
LSH-GS (Ours)	72.5	70.8	21.8M	60.4	62.1	11.1M
LSH-ATPE (Ours)	**72.1**	**69.9**	21.8M	**59.5**	**60.5**	11.1M
LSH-TPE (Ours)	**72.0**	**69.9**	21.8M	**59.0**	**60.2**	11.1M

Effectiveness. We first demonstrate the effectiveness of LSH for selecting hyperparameter configuration, which includes methodological and non-methodological hyperparameters. We consider sparse initialization, death mode, and redistribution mode for the methodological hyperparameters. Sparse initialization determines the initial pruning mask, which has two options, ER and uniform distribution. Death and redistribution modes determine how the weights in the network are pruned and redistributed before and after weight growth, respectively. There are four available death modes and three redistribution modes. For the non-methodological hyperparameters, we consider the death rate, which is the ratio of pruned weight and total nonzeros weight at each time. The death rate is generated from the range $[0.4, 0.9]$. We evaluate LS-based and Loss-based metrics on different hyperparameter optimization algorithms, such as Grid Search, TPE, and Adaptive TPE (APTE). For Stacked LSTM, the pruning ratio is 0.67 and for RHN, it is 0.53. The initial training epoch e_0 and incremental training epoch e_i are $[3, 3]$ and $[6, 3]$ for stacked-LSTM and RHN, respectively. For Grid search, we fix the death rate of stacked LSTM to 0.8 and RHN to 0.5. Therefore, only methodological hyperparameters are considered and result in a total of 24 configurations. For TPE and APTE in Table. 1, the size of the initial candidate set is 40 for both stacked LSTM and RHN. One thing to notice in Table. 1 is that GS, TPE, and APTE row uses loss as the metric instead LS-based metric. As shown in Table 1, LSH (bottom three rows) consistently outperforms dense models, and SOTA pruning methods by a margin. LSH with different optimization algorithms also achieves better performance than loss-based counterpart in most cases (5/6). The best result is achieved by LSH with TPE in both stacked LSTM and RNH case which outperforms the previous best result (69.6 vs. 71.7 for stacked LSTM and 59.0 vs. 60.4 for RHN). Another important observation is that LSH with Advanced hyperparameter optimization methods, such as TPE and ATPE, outperforms LSH with GS. Quantitatively, LSH obtain 0.9 and 1.4 perplexity improvement in stacked LSTM and RHN by switching from Grid Search to TPE while for loss-based counterpart, this improvement does not exist. This improvement means that LSH benefits advanced hyperparameter optimization methods to find better configuration by simultaneously optimizing over methodological and non-methodological hyperparameters.

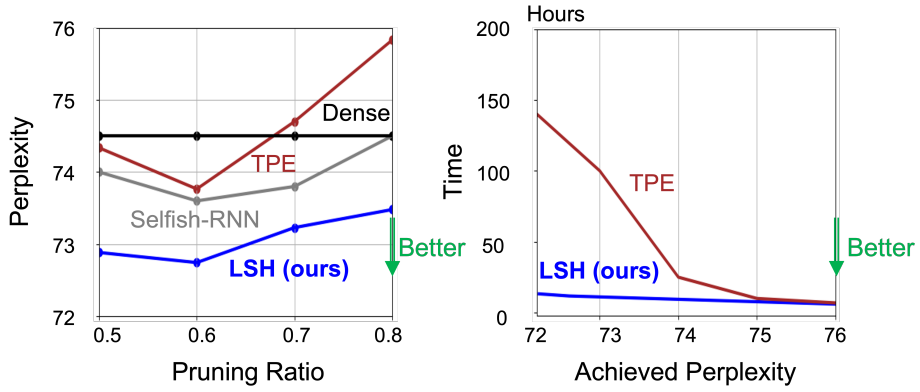

Figure 3: (Left) Pruning Ratio vs. Perplexity: Comparison among LSH (blue), TEP/loss-based (brown), and selfish-RNN (gray) on different pruning ratios.; (Right) Achieved Perplexity vs Time Budget: Comparison between LSH (blue) and loss-based full training search (brown) on required time budget (hours) to reach a target perplexity.

We also test LSH under different resource budgets, i.e, different sizes of the initial candidate set. We consider three resource budgets, low, moderate, and high which have 24, 30, and 40 candidates in the initial set, respectively. We demonstrate it on stacked LSTM with pruning ratio$= 0.67$. As shown in Table. 2, increasing the resource budget for LSH can improve the average performance ($70.9 \rightarrow 69.9$ for LSH with TPE and ATPE) and achieve more reliable result as confidence interval drops ($3.0 \rightarrow 2.6$ for TPE and $0.4 \rightarrow 0.3$ for ATPE)

Table 2: LS-based (LSH) vs. loss-based hyperpruning applied with GS, TPE, and ATPE under different budgets. Stacked LSTM with pruning ratio 0.67 trained on PTB is used here.

Methods	Low		Moderate		High	
	val	test	val	test	val	test
GS	74.2	72.5	74.2	72.5	74.2	72.5
ATPE	74.2 (1.2)	72.0 (1.9)	74.3 (0.7)	72.7 (0.1)	74.3 (0.7)	72.7 (0.1)
TPE	74.1 (1.3)	72.3 (0.8)	74.2 (0.6)	72.4 (1.0)	74.4 (0.5)	72.7 (1.1)
LSH-ATPE (Ours)	**73.3 (0.9)**	**70.9 (0.4)**	**72.1 (0.2)**	**69.9 (0.3)**	**72.1 (0.2)**	**69.9 (0.3)**
LSH-TPE (Ours)	**72.6 (1.9)**	**70.9 (3.0)**	**72.8 (1.1)**	**70.9 (2.6)**	**72.1 (0.2)**	**69.9 (2.6)**
LSH-GS (Ours)	**72.5**	**70.8**	**72.5**	**70.8**	**72.5**	**70.8**

Robustness. We also evaluate the robustness of LSH to pruning ratios. We test from moderate (0.5) to high (0.8) pruning ratios by a step of 0.1. For each experiment, TPE is used as the hyperparameter optimization algorithm and the size of the initial candidate set is limited to 20. The experiment is conducted on stacked LSTM and the same elimination/insertion schedule is used. As shown in Fig. 3(left), LSH (blue) consistently outperforms TPE/loss-based (brown), selfish-RNN (gray). It is also better than dense under all pruning ratios.

Time Efficiency. To show the efficiency of the LSH, we estimate the time that the LSH uses to reach a particular target perplexity. We compare it with the baseline loss-based full training search, which selects the optimal configuration after finishing training all candidates. We use stacked LSTM with a fixed pruning ratio of 0.67 for these experiments. Fig. 3 (right) shows the amount of time (y-axis) required for finding a configuration with a particular target perplexity (x-axis). We observe that LSH is more efficient than the loss-based approach for all perplexities. Particularly, for finding the optimal configuration (72), LSH ($\approx$15 hours) is an order of magnitude faster than loss-based approach ($\approx$150 hours).

Ablation Study. In this section, we perform the ablation study on stacked LSTM to show the completeness of LSH.

We first studied the efficiency and effectiveness of different elimination/insertion schedules set by the starting epoch e_0 and the incremental epoch e_i. We fixed the search method to be Grid Search, where three categorical hyperparameters are considered resulting in 24 different configurations. In Table. 3 we show the number of epochs required for the candidate selection process under each schedule. Cells with green color denote that the optimal configuration was indeed selected within the remaining two candidates after halving. Even though with $e_0 = 3, e_i = 1$, the minimum number of required epochs 130is achieved, the performance might not be robust across different experiments since, with a small change in e_0 direction, the selection might fail. Therefore, we set $e_0 = e_i = 3$ in all experiments which is robust is both e_0 and e_i directions. Even though this means the number of required epochs for candidate selection increases by 116 ($130 \rightarrow 246$) which is about 2 hour, and is still relatively shorter compared to the extensive training of two remaining candidates (> 20 hours).

Table 3: The efficiency and the effectiveness of scheduling. Each cell reports the number of required epochs for the halving procedure.

e_i \ e_0	1	2	3	4
1	82	106	130	154
2	140	164	188	212
3	198	222	246	270
4	256	280	304	328

Another factor that could delay LSH is the LS computation. Even though the computation of LS is relatively faster than training the network itself, i.e., 6s for computing LS for each sample vs. 120s for one training epoch. However, with a large LS computation batch size, the overall process can significantly slow down. Here we empirically show that only a few validation samples are needed for the LS computation. We demonstrate it by comparing the maximum, mean, and variance of LS computed from different numbers of validation samples across two pruned models. As discussed in (63), those are the main features that determine the characteristic of LS. As shown in Table. 4, maximum, mean and variance are similar for LS calculated from 2 and 10 validation samples. Their different Δ is one order of magnitude smaller compared to the difference between different pruned models.

Table 4: Effect of LS calculation batch size: The maximum, mean and variance of LS computed from different numbers of validation samples on two pruned models.

	Pruned 1			Pruned 2		
	2	10	Δ	2	10	Δ
maximum	-1.26	-1.20	0.06	-0.52	-0.46	0.06
mean	-4.40	-4.41	0.01	-3.97	-3.97	0
variance	2.59	2.67	0.08	1.92	1.98	0.06

5 Conclusion

In this work, we consider the problem of hyperpruning which requires selecting the optimal pruning method with the corresponding hyperparameter configuration for a particular task. To resolve that, we propose a novel Lyapunov Spectra(LS)-based hyperpruning algorithm, called LSH, to effectively and efficiently select the optimal configuration. LSH iteratively eliminates non-promising pruned variants based on their distance to a dense/non-pruned reference model in LS space and adaptively generate new variants. Eventually, we allocate most resource to the most promising candidates. Our experiment is conducted on stacked LSTM and RHN model trained with the Penn TreeBank Dataset on language modeling task. Our result shows that LSH outperforms its loss-based counterpart, other SOTA pruning method, and the dense model. Further, our method can be incorporated into different existing hyperparameter optimization algorithms, such as TPE and ATPE, and is robust to different pruning ratios. Our method achieves an order of magnitude boost compared to the loss-based full training search in terms of efficiency.

REFERENCES

[1] Henry DI Abarbanel, Reggie Brown, and Matthew B Kennel. Variation of lyapunov exponents on a strange attractor. *Journal of Nonlinear Science*, 1(2):175–199, 1991.

[2] Ludwig Arnold. Random dynamical systems. *Dynamical systems*, pages 1–43, 1995.

[3] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[6] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *arXiv preprint arXiv:2107.05847*, 2021.

[7] A Brandstäter, J Swift, Harry L Swinney, A Wolf, J Doyne Farmer, Erica Jen, and PJ Crutchfield. Low-dimensional chaos in a hydrodynamic system. *Physical Review Letters*, 51(16):1442, 1983.

[8] Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

[9] Xiaoliang Dai, Hongxu Yin, and Niraj K Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019.

[10] Silvina Dawson, Celso Grebogi, Tim Sauer, and James A Yorke. Obstructions to shadowing when a lyapunov exponent fluctuates about zero. *Physical review letters*, 73(14):1927, 1994.

[11] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[12] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, Kevin Leyton-Brown, et al. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10, 2013.

[13] Rainer Engelken, Fred Wolf, and Larry F Abbott. Lyapunov spectra of chaotic recurrent neural networks. *arXiv preprint arXiv:2006.02427*, 2020.

[14] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[15] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[16] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*, 2020.

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

[18] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

[19] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[20] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[21] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[22] Gerardo Hernández, Erik Zamora, Humberto Sossa, Germán Téllez, and Federico Furlán. Hybrid neural networks for big data classification. *Neurocomputing*, 390:327–340, 2020.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[25] Wen-Chin Huang, Tomoki Hayashi, Yi-Chiao Wu, Hirokazu Kameoka, and Tomoki Toda. Pretraining techniques for sequence-to-sequence voice conversion. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:745–755, 2021.

[26] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[27] Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Language modeling with deep transformers. *arXiv preprint arXiv:1905.04226*, 2019.

[28] Steven A Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989.

[29] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33:20744–20754, 2020.

[30] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, pages 5544–5555. PMLR, 2020.

[31] Thomas Laurent and James von Brecht. A recurrent neural network without chaos. *arXiv preprint arXiv:1612.06212*, 2016.

[32] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[33] Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip HS Torr. A signal propagation perspective for pruning neural networks at initialization. *arXiv preprint arXiv:1906.06307*, 2019.

[34] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[35] Robert Legenstein and Wolfgang Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural networks*, 20(3):323–334, 2007.

[36] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[37] Tao Lin, Sebastian U Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic model pruning with feedback. *arXiv preprint arXiv:2006.07253*, 2020.

[38] Junjie Liu, Zhe Xu, Runbin Shi, Ray CC Cheung, and Hayden KH So. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. *arXiv preprint arXiv:2005.06870*, 2020.

[39] Shiwei Liu, Decebal Constantin Mocanu, Yulong Pei, and Mykola Pechenizkiy. Selfish sparse rnn training. In *International Conference on Machine Learning*, pages 6893–6904. PMLR, 2021.

[40] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[41] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89, pages 89–94, 2015.

[42] Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, page 273, 1994.

[43] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[44] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017.

[45] Michael Monteforte and Fred Wolf. Dynamical entropy production in spiking neuron networks in the balanced state. *Physical review letters*, 105(26):268104, 2010.

[46] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pages 4646–4655. PMLR, 2019.

[47] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, 1, 1988.

[48] Michael C Mozer and Paul Smolensky. Using relevance to reduce network size automatically. *Connection Science*, 1(1):3–16, 1989.

[49] Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. Exploring sparsity in recurrent neural networks. arxiv e-prints, art. *arXiv preprint arXiv:1704.05119*, 2017.

[50] Gunter Ochs. Stability of oseledets spaces is equivalent to stability of lyapunov exponents. *Dynamics and Stability of Systems*, 14(2):183–201, 1999.

[51] Valery Oseledets. Oseledets theorem. *Scholarpedia*, 3(1):1846, 2008.

[52] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1924–1932. PMLR, 2018.

[53] Antônio H Ribeiro, Koen Tiels, Luis A Aguirre, and Thomas Schön. Beyond exploding and vanishing gradients: analysing rnn training using attractors and smoothness. In *International Conference on Artificial Intelligence and Statistics*, pages 2370–2380. PMLR, 2020.

[54] David Ruelle. Ergodic theory of differentiable dynamical systems. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 50(1):27–58, 1979.

[55] Nobuhiko Saitô and Atsushi Ichimura. Ergodic components in the stochastic region in a hamiltonian system. In *Stochastic Behavior in Classical and Quantum Hamiltonian Systems*, pages 137–144. Springer, 1979.

[56] Hiroshi Shibata. Ks entropy and mean lyapunov exponent for coupled map lattices. *Physica A: Statistical Mechanics and its Applications*, 292(1-4):182–192, 2001.

[57] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[58] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

[59] Kun Su, Xiulong Liu, and Eli Shlizerman. Predict & cluster: Unsupervised skeleton based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9631–9640, 2020.

[60] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33:6377–6389, 2020.

[61] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.

[62] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

[63] Ryan Vogt, Maximilian Puelma Touzel, Eli Shlizerman, and Guillaume Lajoie. On lyapunov exponents for rnns: Understanding information propagation using dynamical systems tools. *Frontiers in Applied Mathematics and Statistics*, 8, 2022.

[64] Ryan Vogt, Maximilian Puelma Touzel, Eli Shlizerman, and Guillaume Lajoie. On lyapunov exponents for rnns: Understanding information propagation using dynamical systems tools. *arXiv preprint arXiv:2006.14123*, 2020.

[65] Ryan Vogt, Yang Zheng, and Eli Shlizerman. Lyapunov-guided embedding for hyperparameter selection in recurrent neural networks. *arXiv preprint arXiv:2204.04876*, 2022.

[66] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

[67] Huan Wang, Can Qin, Yue Bai, Yulun Zhang, and Yun Fu. Recent advances on neural network pruning at initialization. *arXiv e-prints*, pages arXiv–2103, 2021.

[68] Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. Learning intrinsic sparse structures within long short-term memory. *arXiv preprint arXiv:1709.05027*, 2017.

[69] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[70] Michio Yamada and Koji Ohkitani. The inertial subrange and non-positive lyapunov exponents in fully-developed turbulence. *Progress of theoretical physics*, 79(6):1265–1268, 1988.

[71] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

[72] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7405–7414, 2018.

[73] Yang Zheng and Eli Shlizerman. R-force: Robust learning for random recurrent neural networks. *arXiv preprint arXiv:2003.11660*, 2020.

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

[75] Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutnık, and Jürgen Schmidhuber. Recurrent highway networks. In *International conference on machine learning*, pages 4189–4198. PMLR, 2017.