Image Captioning with Sparse Recurrent Neural Network

Jia Huei Tan* **Chee Seng Chan†**
Center of Image and Signal Processing,
Faculty of Computer Science and Technology,
University of Malaya, 50603 Kuala Lumpur
Malaysia

Joon Huang Chuah‡
Department of Electrical Engineering,
Faculty of Engineering,
University of Malaya,
50603 Kuala Lumpur, Malaysia

Abstract

Recurrent Neural Network (RNN) has been deployed as the de facto model to tackle a wide variety of language generation problems and achieved state-of-the-art (SOTA) performance. However despite it's impressive results, the large number of parameters in the RNN model makes deployment in mobile and embedded devices infeasible. Driven by this problem, many works have proposed a number of pruning methods to reduce the sizes of the RNN model. In this work, we propose an end-to-end pruning method for image captioning models equipped with visual attention. Our proposed method is able to achieve sparsity levels up to 97.5% without significant performance loss relative to the baseline ($\sim 1\%$ loss at $40\times$ compression of GRU model). Our method is also simple to use and tune, facilitating faster development times for neural network practitioners. We perform extensive experiments on the popular MS-COCO dataset in order to empirically validate the efficacy of our proposed method.

1 Introduction

Automatically generating a caption that describes an image, a problem known as image captioning, is a challenging problem where computer vision (CV) meets natural language processing (NLP). A well performing model not only has to identify the objects in the image, but also capture the semantic relationship between them, general context and the activities that they are involved in. Lastly, the model has to map the visual representation into a fully-formed sentence in a natural language such as English.

A good image captioning model can have many useful applications, which include helping the visually impaired to better understand the web contents, providing descriptive annotations of website contents, and enabling better context-based image retrieval by tagging images with accurate natural language descriptions.

Driven by user privacy concerns and the quest for lower user-perceived latency, deployment on edge devices away from remote servers is required. As edge devices usually have limited battery capacity and thermal limits, this presents a few key challenges in the form of storage size, power consumption and computational demands [1].

For models incorporating RNNs, on-device inference is often memory bandwidth-bound. As RNN parameters are fixed at every time step, parameter reading forms the bulk of the work [2, 1]. As such, RNN pruning offers the opportunity to not only reduce the amount of memory access but also fitting the model in on-chip SRAM cache rather than off-chip DRAM memory, both of which dramatically reduce power consumption [3, 4]. Similarly, sparsity patterns for pruned RNNs are fixed across time steps. This offers the potential to factorise scheduling and load balancing operations outside of the loop and enable reuse [2]. Lastly, pruning allows larger RNNs to be stored in memory and trained [2, 5]

*{tanjiahuei@siswa.um.edu.my}
†{cs.chan@um.edu.my}
‡{jhchuah@um.edu.my}

Preprint. 29 August 2019.

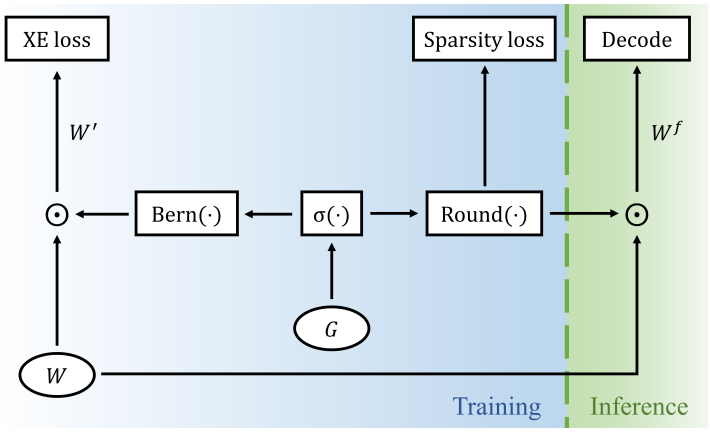


Figure 1: An overview of our proposed end-to-end pruning method. "XE" denotes cross-entropy. More details can be found in Sec. 3.2.

In this work, we propose a one-shot end-to-end pruning method to produce very sparse image captioning decoders (up to 97.5% sparsity) while maintaining good performance relative to the dense baseline model as well as competing methods. We detail our contributions in the following section (Sec. 2.2).

2 Related Works

Our work is most related to the current research on model pruning, particularly involving generative language RNNs. This section reviews the most relevant works on this topic.

2.1 Model pruning

Modern neural networks that provide good performance tend to be large and overparameterised, fuelled by observations that larger [6, 7, 8] networks tend to be easier to train. This in turn drives numerous efforts to reduce model size using techniques such as weight pruning and quantisation [9, 10, 11].

Early works like [12] and [13] explored pruning by computing the Hessian of the loss with respect to the parameters in order to assess the saliency of each parameter. Other works involving saliency computation include [14] and [15] where sensitivity of the loss with respect to neurons and weights are used respectively. On the other hand, works such as [16, 17] directly induce network sparsity by incorporating sparsity-enforcing penalty terms into the loss function.

Most of the recent works in network pruning focused on vision-centric classification tasks using Convolutional Neural Networks (CNNs) and occasionally RNNs. Techniques proposed include magnitude-based pruning [3, 4, 18] and variational pruning [19, 20, 21]. Among these, magnitude-based weight pruning have become popular due to their effectiveness and simplicity. Most notably, [3] employed a combination of pruning, quantization and Huffman encoding resulting in massive reductions in model size without affecting accuracy. While unstructured sparse connectivity provides reduction in storage size, it requires sparse General Matrix-Matrix Multiply (GEMM) libraries such as cuSPARSE and SPBLAS in order to achieve accelerated inference. Motivated by existing hardware architectures optimised for dense linear algebra, many works propose techniques to prune and induce sparsity in a structured way in which entire filters are removed [22, 23, 24].

On the other hand, works extending connection pruning to RNN networks are considerably fewer [25, 2, 1, 26]. See *et al*. [25] first explored magnitude-based pruning applied to deep multi-layer neural machine translation (NMT) model with Long-Short Term Memory (LSTM) [27]. In their work, three pruning schemes are evaluated which include class-blind, class-uniform and class-distribution. Class-blind pruning was found to produce the best result compared to the other two schemes. Narang *et al*. [2] introduced a gradual magnitude-based pruning scheme for speech recognition RNNs whereby all the weights in a layer less than some chosen threshold are pruned. Gradual pruning is performed in parallel with network training while pruning rate is controlled by a slope function with two distinct phases. This is extended by Zhu and Gupta [1] who simplified the gradual pruning scheme with reduced hyperparameters.

2.2 Our contribution

Our proposed end-to-end pruning method possesses three main qualities:

i) **Simple and fast.** Our approach enables easy pruning of the RNN decoder equipped with visual attention, whereby the best number of weights to prune in each layer is automatically determined. Compared to works such as [1, 2], our approach is simpler with a single hyperparameter versus 3-4 hyperparameters. Our method also does not rely on reinforcement learning techniques such as in the work of [28]. Moreover, our method applies pruning to all the weights in the RNN decoder and does not require special considerations to exclude pruning from certain weight classes. Lastly our method completes pruning in a single-shot process rather than requiring iterative train-and-prune process as in [29, 30, 31, 32].

ii) **Good performance-to-sparsity ratio enabling extreme sparsity.** Our approach achieves good performance across sparsity levels from 80% up until 97.5% ($40\times$ reduction in Number of Non-zeros (NNZ) parameters). This is in contrast with competing methods [1, 25] where there is a significant performance drop-off starting at sparsity level of 90%.

iii) **Easily tunable sparsity level.** Our approach provides a way for neural network practitioners to easily control the level of sparsity and compression desired. This allows for model solutions that are tailored for each particular scenario. In contrast, while the closely related works of [33, 34] also provide good performance with the incorporation of gating variables, there is not a straightforward way of controlling the final sparsity level. In their works, regularisers such as bi-modal, l_2 , l_1 and l_0 regulariser are used to encourage network sparsity. Their work also only focuses on image classification using CNNs.

While there are other works on compressing RNNs, most of the methods proposed either comes with structural constraints or are complementary to model pruning in principle. Examples include using low-rank matrix factorisations [35, 36], product quantisation on embeddings [37], factorising word predictions into multiple time steps [38, 39, 40], and grouping RNNs [41].

Lastly, another closely related work by [30] also incorporated model pruning into image captioning. However we note three notable differences: 1) their work is focused on proposing a new LSTM cell structure named the *H-LSTM*; 2) their work utilises the grow-and-prune (GP) method [31] which necessitates compute and time expensive iterative pruning; and 3) the compression figures stated are calculated based on the size of the LSTM cells instead of the entire decoder.

3 Proposed Method

Our proposed method involves incorporating learnable gating parameters into regular image captioning framework. We denote weight, bias and gating matrices as W, B and G respectively. For a model with L layers, the captioning and gating parameters are denoted as θ and ϕ such that $\theta = \{W_{1:L}, B_{1:L}\}$ and $\phi = \{G_{1:L}\}$.

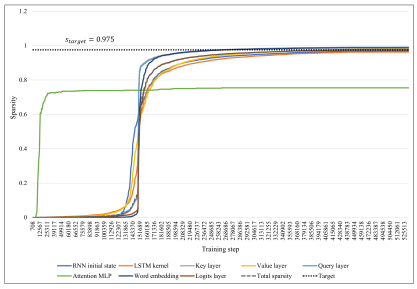
As there are substantial existing works focusing on pruning CNNs, we focus our efforts on pruning generative RNNs. As such, we only prune the RNN decoder. All model size calculations in this work include only the decoder (including attention module) while the encoder (i.e. CNN) is excluded.

3.1 Image captioning with visual attention

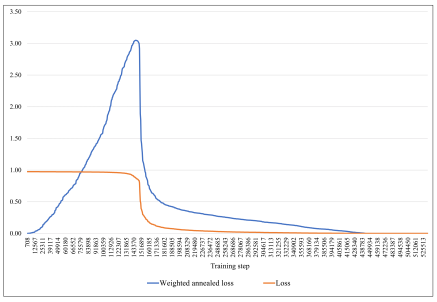
Our image captioning framework of interest is a simplified variant of the *Show, Attend and Tell* [42] model which uses a single layer RNN network equipped with visual attention on the CNN feature map. It is a popular framework that forms the basis for subsequent state-of-the-art (SOTA) works on image captioning [43, 44]. In this work, we employ LSTM and Gated Recurrent Unit (GRU) [45] as the RNN cell.

Suppose $\{S_0, \ ... \ , \ S_{T-1}\}$ is a sequence of words in a sentence of length T, the model directly maximises the probability of the correct description given an image I using the following formulation:

$$\log p\left(S \mid I\right) = \sum_{t=0}^{T} \log p\left(S_t \mid I, \ S_{0:t-1}, \ c_t\right) \tag{1}$$



(a) Sparsity level for different layers of the RNN decoder.



(b) Sparsity loss L_s.

Figure 2: Training progression of our proposed end-to-end pruning method. Best viewed in colour. Detailed explanation for (a) is given in Sec. 5.6. In (b), "Weighted annealed loss" refers to $\lambda_s L_s$ in Eq. 14 while "Loss" refers to L_s before applying cosine annealing in Eq. 11.

where t is the time step, $p\left(S_t \mid I,\ S_{0:t-1},\ c_t\right)$ is the probability of generating a word given an image I, previous words $S_{0:t-1}$, and context vector c_t.

For a RNN network with r units, the hidden state of RNN is initialised with the image embedding vector as follows:

$$h_{t=-1} = W_I\, I_{embed}\ ,\ m_{t=-1} = 0 \tag{2}$$

where $W_I \in \mathbb{R}^{\,r \times h}$ is a weight matrix and h is the size of I_{embed}.

The attention function used in this work is *soft-attention* introduced by [46] and used in [42], where a multilayer perceptron (MLP) with a single hidden layer is employed to calculate the attention weights on a particular feature map. The context vector c_t is then concatenated with previous predicted word embedding to serve as input to the RNN. Finally, a probability distribution over the vocabulary is produced from the hidden state h_t:

$$p_t = \text{Softmax}\left(E_o\, h_t\right) \tag{3}$$

$$h_t\ ,\ m_t = \text{RNN}\left(x_t\ ,\ h_{t-1}\ ,\ m_{t-1}\right) \tag{4}$$

$$x_t = \left[E_w\, S_{t-1},\ c_t\right] \tag{5}$$

$$c_t = \text{SoftAtt}\left(f\right) \tag{6}$$

where $E_w \in \mathbb{R}^{\,q \times v}$ and $E_o \in \mathbb{R}^{\,v \times r}$ are input and output embedding matrices respectively; p_t is the probability distribution over the vocabulary V; m_t is the memory state; x_t is the current input; $S_{t-1} \in \mathbb{R}^{\,q}$ is the one-hot vector of previous word; $c_t \in \mathbb{R}^{\,a}$ is the context vector; f is the CNN feature map; and $[\,,]$ is the concatenation operator. For GRU, all m_t terms are ignored.

Finally, the standard cross-entropy loss function for the captioning model θ is given by:

$$L_c = -\sum_{t}^{T} \log p_{\,t}\left(S_t\right) + \lambda_d \left\|\theta\right\|_2^2 \tag{7}$$

3.2 End-to-end pruning

Formulation. Similar to [1], TensorFlow framework is extended to prune network connections during training. Inspired by the concept of learnable *Supermasks* introduced by [33, 47], our proposed method achieves model pruning via learnable gating variables that are trained in an end-to-end fashion. An overview of our method is illustrated in Fig. 1.

For every weight variable matrix W to be pruned, we create a gating variable matrix G with the same shape as W. This gating matrix G functions as a masking mechanism that determines which of the parameter w in the weight matrix W participates in both forward-execution and back-propagation of the graph.

To achieve this masking effect, we calculate the effective weight tensor as follows:

$$W_l^{'} = W_l \odot G_l^b \tag{8}$$

$$G_l^b = \mathrm{z}(\sigma(G_l)) \tag{9}$$

where $W_l, G_l \in \mathbb{R}^D$ are the original weight and gating matrices from layer l with shape D; and superscript $(\cdot)^b$ indicates binary sampled variables. $\odot$ is element-wise multiplication; $\sigma(\cdot)$ is a point-wise function that transforms continuous values into the interval $(0, 1)$; and $\mathrm{z}(\cdot)$ is a point-wise function that samples from a Bernoulli distribution. The composite function $\mathrm{z}(\sigma(\cdot))$ thus effectively transforms continuous values into binary values.

Binary gating matrices G^b can be obtained by treating $\sigma(G)$ as Bernoulli random variables. While there are many possible choices for the σ function, we decided to use the logistic sigmoid function following [48] and [47]. To sample from the Bernoulli distribution, we can either perform a *unbiased draw* or a *maximum-likelihood* (ML) draw [33]. Unbiased draw is the usual sampling process where a gating value $g \in (0, 1)$ is binarised to 1.0 with probability g and 0.0 otherwise, whereas ML draw involves thresholding the value g at 0.5. In this work, we denote unbiased and ML draw using the sampling functions $\mathrm{z}(\cdot) = \mathrm{Bern}(\cdot)$ and $\mathrm{z}(\cdot) = \mathrm{Round}(\cdot)$ respectively. We back-propagate through both sampling functions using the *straight-through estimator* [48] (i.e. $\delta\, \mathrm{z}(g)/\delta g = 1$).

Prior to training, all the gating variables are initialised to the same constant value m while the weights and biases of the network are initialised using standard initialisation schemes (e.g. Xavier [49]). During training, both sampling functions $\mathrm{Bern}(\cdot)$ and $\mathrm{Round}(\cdot)$ are used in different ways. To obtain the effective weight tensor used to generate network activations, we utilised $\mathrm{Bern}(\cdot)$ to inject some stochasticity that helps with training and to mitigate the bias arising from the constant value initialisation. Thus the effective weight calculation becomes:

$$W_l^{'} = W_l \odot \mathrm{Bern}(\sigma(G_l)) \tag{10}$$

To drive the sparsity level of gating variables ϕ to the user-specified level s_{target}, we introduce a regularisation term L_s. Consistent with the observations in the works of [1] and [32], we found that annealing the loss over the course of training produces the best result. Annealing is done using a cosine curve α defined in Eq. 12. To ensure determinism when calculating sparsity, we use $\mathrm{Round}(\cdot)$ to sample from $\sigma(G)$:

$$L_s = (1 - \alpha) \times \left| \, s_{target} - \frac{p_{nnz}}{p_{total}} \, \right| \tag{11}$$

$$\alpha = \frac{1}{2}\left(1 + \cos\left(\frac{n\pi}{n_{max}}\right)\right) \tag{12}$$

$$p_{nnz} = \sum_{l=0}^{L} \sum_{j=0}^{J} \mathrm{Round}(\sigma(g_{j,l})) \tag{13}$$

where p_{nnz} is the number of NNZ gating parameters; p_{total} is the total number of gating parameters; n and n_{max} is the current and final training step respectively; $g_{j,l}$ is the gating parameter at position j in the matrix G_l from layer l; L is the number of layers; and J is the number of parameters in matrix G_l. The progression of sparsity loss L_s as well as the sparsity levels of various layers in the decoder are illustrated in Fig. 2b and 2a respectively.

The final objective function used to train the captioning model θ with gating variables ϕ is:

$$L\,(\,I,\, S,\, s_{target}\,) = L_c + \lambda_s L_s \tag{14}$$

Intuitively, the captioning loss term L_c provides supervision for learning of the saliency of each parameter where important parameters are retained with higher probability while unimportant ones are dropped more frequently. On the other hand, the sparsity regularisation term L_s pushes down the average value of the Bernoulli gating parameters so that most of them have a value less than 0.5 after sigmoid activation. The hyperparameter λ_s determines the weightage of L_s. If λ_s is too low, the target sparsity level might not be attained; whereas high values might slightly affect performance (see Sec. 5.1).

Table 1: The effects of varying gating variable initialisation value m on MS-COCO. Sparsity level is set to 0.8. **Bold text** indicates best overall performance.

Gating init. value	MS-COCO test set scores							
	B-1	B-2	B-3	B-4	M	R	C	S
$m = 5.0$	**71.6**	**54.8**	**41.4**	**31.4**	**24.6**	**52.8**	**94.4**	**17.5**
$m = 2.5$	71.3	54.5	41.1	31.1	24.4	52.5	93.1	17.4
$m = 0$	71.3	54.3	40.8	30.6	24.4	52.6	92.4	17.3
$m = -2.5$	70.8	53.8	40.2	30.1	24.1	52.1	91.1	17.0
$m = -5.0$	70.5	53.5	39.8	29.5	23.6	51.8	88.0	16.5

Table 2: The effects of varying sparsity loss weightage λ_s on MS-COCO. Sparsity target is set to $s_{target} = 0.9$. **Bold text** indicates best overall performance.

Gating init. value	Sparsity	MS-COCO test set scores							
		B-1	B-2	B-3	B-4	M	R	C	S
$\lambda_s = 1.0$	0.662	71.4	54.5	41.1	31.0	24.7	52.7	94.1	17.4
$\lambda_s = 2.0$	0.830	71.6	54.7	41.1	31.0	24.6	52.7	93.9	17.5
$\lambda_s = 5.0$	**0.900**	**71.4**	**54.3**	**40.8**	**30.8**	**24.4**	**52.4**	**93.1**	**17.3**
$\lambda_s = 10.0$	0.900	71.1	54.3	40.8	30.6	24.4	52.5	92.7	17.3

Training and Inference. The training process of the captioning model is divided into two distinct stages: decoder training and end-to-end fine-tuning. During the decoder training stage, we freeze the CNN parameters and only learn decoder and gating parameters by optimising the loss given in Eq. 14. For the fine-tuning stage, we restore all the parameters θ and ϕ from the last checkpoint at the end of decoder training and optimise the entire model including the CNN. During this stage, $\mathrm{Bern}(\cdot)$ is still used but all ϕ parameters are frozen.

After training is completed, all the weight matrices $W_{1:L}$ is transformed into sparse matrices by sampling from $G_{1:L}$ using $\mathrm{Round}(\cdot)$, after which G can be discarded. In other words, the final weights W^f are calculated as:

$$W_l^f = W_l \odot \mathrm{Round}(\sigma(G_l)) \tag{15}$$

4 Experiment Setup

Unless stated otherwise, all experiments have the following configurations. We did not perform extensive hyperparameter search due to limited resources.

4.1 Hyperparameters

Models are implemented using TensorFlow r1.9. The image encoder used in this work is *GoogLeNet (InceptionV1)* with batch normalisation [50, 51] pre-trained on ImageNet [52]. The input images are resized to 256×256, then randomly flipped and cropped to 224×224 before being fed to the CNN. The attention function $\mathrm{SoftAtt}(\cdot)$ operates on the *Mixed-4f* map $f \in \mathbb{R}^{196 \times 832}$. The size of context vector c_t and attention MLP is set to $a = 512$. A single layer LSTM or GRU network with hidden state size of $r = 512$ is used. The word size is set to $q = 256$ dimensions.

The optimiser used for decoder training is Adam [53], with batch size of 32. The initial learning rate (LR) is set to 1×10^{-2}, and annealed using the cosine curve α defined in Eq. 12, ending at 1×10^{-5}. All models are trained for 30 epochs. Weight decay rate is set to $\lambda_d = 1 \times 10^{-5}$. For fine-tuning, a smaller initial LR of 1×10^{-3} is used and the entire model is trained for 10 epochs. Captioning model parameters are initialised randomly using Xavier uniform initialisation [49].

The input and output dropout rates for dense RNN are both set to 0.35, while the attention map dropout rate is set to 0.1. Following [4, 2], a lower dropout rate is used for sparse networks where RNN and attention dropout rates are set to 0.11 and 0.03 respectively. This is done to account for the reduced capacity of the sparse models.

For fair comparison, we apply pruning to all weights of the captioning model for all of the pruning schemes. For our proposed method, we train the gating variables ϕ with a higher constant LR of

100 without annealing, which is consistent with [47]. We found that LR lower than 100 causes ϕ to train too slowly. We set λ_s according to this heuristic: $\lambda_s = \max(5, 0.5/(1 - s_{target}))$. All gating parameters ϕ are initialised to a constant $m = 5.0$, see Sec. 5.1 for other values.

For *gradual pruning* [1], pruning is started after first epoch is completed and ended at the end of epoch 15, following the general heuristics outlined in [2]. Pruning frequency is set to 1000. We use the standard scheme where each layer is pruned to the same pruning ratio at every step. For *hard pruning* [25], pruning is applied to the dense baseline model after training is completed. Retraining is then performed for 10 epochs. LR and annealing schedule are the same as used for dense baseline.

For inference, beam search is used in order to better approximate $S = \arg\max_{S'} p(S' \mid I)$. Beam size is set to $b = 3$ with no length normalisation. We evaluate the last checkpoint upon completion of training for all the experiments. We denote compression ratio as CR.

4.2 Dataset

The experiments are performed on the popular MS-COCO dataset [54]. It is a public English captioning dataset which contains $123,287$ images and each image is given at least 5 captions by different Amazon Mechanical Turk (AMT) workers. As there is no official test split with annotations available, the publicly available split[4] in the work of [55] is used in this work. The split assigns $5,000$ images for validation, another $5,000$ for testing and the rest for training. We reuse the publicly available tokenised captions. Words that occur less than 5 times are filtered out and sentences longer than 20 words are truncated.

All the scores are obtained using the publicly available MS-COCO evaluation toolkit[5] , which computes BLEU [56], METEOR [57], ROUGE-L [58], CIDEr [59] and SPICE [60]. For sake of brevity, we label BLEU-1 to BLEU-4 as B-1 to B-4, and METEOR, ROUGE-L, CIDEr, SPICE as M, R, C, S respectively.

5 Experiments and Discussion

5.1 Ablation study

Table 1 shows the effect of various gating initialisation values. From the table, we can see that the best overall performance is achieved when m is set to 5. Starting the gating parameters at a value of 5 allows all the captioning parameters θ to be retained with high probability at the early stages of training, allowing better convergence. This observation is also consistent with the works of [1] and [32], where the authors found that gradual pruning and late resetting can lead to better model performance. Thus, we recommend setting $m = 5.0$ for most cases.

Table 2 shows the effect of sparsity regularisation weightage λ_s. This is the important hyperparameter that could affect the final sparsity level at convergence. From the results, we can see that low values lead to insufficient sparsity, and higher sparsity target s_{target} requires higher λ_s. For image captioning on MS-COCO, we empirically determined that the heuristic given in Sec. 4.1 works sufficiently well for sparsity levels from 80% to 97.5% (see Table 3 and 4.

5.2 Comparison with RNN pruning methods

In this section, we provide extensive comparisons of our proposed method with the dense baselines as well as competing methods at multiple sparsity levels. All the models have been verified to have achieved the targeted sparsity levels. From Table 3 and 4, we can clearly see that our proposed end-to-end pruning provides good performance when compared to the dense baselines. This is true even at high pruning ratios of 90% and 95%. The relative drops in BLEU-4 and CIDEr scores are only -1.0% to -2.9% and -1.3% to -2.9% while having $10-20\times$ fewer NNZ parameters. This is in contrast with competing methods whose performance drops are double or even triple compared to ours, especially for LSTM.

The performance advantage provided by end-to-end pruning is even more apparent at the extreme pruning ratio of 97.5%, offering a big $40\times$ reduction in NNZ parameters. Even though we suffered relative degradations of -4.8% to -6.4% in BLEU-4 and CIDEr scores compared to baselines, our performance is still significantly better than the next-closest method which is gradual pruning. On the other hand, the performance achieved by our 80% pruned models are

[4] http://cs.stanford.edu/people/karpathy/deepimagesent/

[5] https://github.com/tylin/coco-caption

Table 3: Comparison with dense LSTM baseline and competing methods. **Bold text** indicates best overall performance. "Gradual" and "Hard" denote methods proposed in [1] and [25].

Approaches	NNZ parameters		MS-COCO test set scores							
	Sparsity	Overall CR	B-1	B-2	B-3	B-4	M	R	C	S
Dense LSTM baseline	0	1 ×	71.8	54.8	41.3	31.1	24.6	52.8	94.3	17.4
Hard (class-uniform)	0.800	5 ×	71.6	54.5	41.0	30.8	24.6	52.7	93.7	17.5
Hard (class-distribution)			71.5	54.5	40.9	30.8	24.7	52.7	93.5	17.4
Hard (class-blind)			71.5	54.7	41.2	31.1	24.7	52.7	94.2	17.5
Gradual			71.5	54.7	41.2	31.1	24.5	52.8	94.0	17.4
Our Proposed ($\lambda_s = 5$)			**71.6**	**54.8**	**41.4**	**31.4**	**24.6**	**52.8**	**94.4**	**17.5**
Hard (class-uniform)	0.900	10 ×	70.9	53.8	40.3	30.2	24.1	52.1	90.8	16.8
Hard (class-distribution)			70.7	53.7	40.2	30.1	24.0	52.1	90.9	16.9
Hard (class-blind)			71.1	53.9	40.4	30.3	24.2	52.2	91.8	17.2
Gradual			71.0	54.0	40.5	30.4	24.1	52.3	91.4	17.0
Our Proposed ($\lambda_s = 5$)			**71.4**	**54.3**	**40.8**	**30.8**	**24.4**	**52.4**	**93.1**	**17.3**
Hard (class-uniform)	0.950	20 ×	69.1	51.7	38.0	27.9	22.9	50.6	83.7	15.8
Hard (class-distribution)			68.8	51.4	37.7	27.6	22.8	50.4	83.2	15.8
Hard (class-blind)			69.5	52.5	38.9	29.0	23.3	51.3	87.0	16.3
Gradual			70.6	53.7	40.2	30.1	23.8	52.0	89.7	16.8
Our Proposed ($\lambda_s = 10$)			**71.2**	**54.2**	**40.7**	**30.6**	**24.3**	**52.4**	**92.1**	**17.2**
Hard (class-uniform)	0.975	40 ×	66.6	48.9	35.3	25.4	21.5	48.8	75.1	14.4
Hard (class-distribution)			65.9	48.1	34.7	25.0	21.1	48.3	72.2	14.0
Hard (class-blind)			66.9	48.9	35.3	25.6	21.6	48.9	75.9	14.6
Gradual			69.3	52.0	38.4	28.3	23.0	50.9	84.1	15.8
Our Proposed ($\lambda_s = 20$)			**70.4**	**53.4**	**39.8**	**29.6**	**23.7**	**51.8**	**88.5**	**16.7**

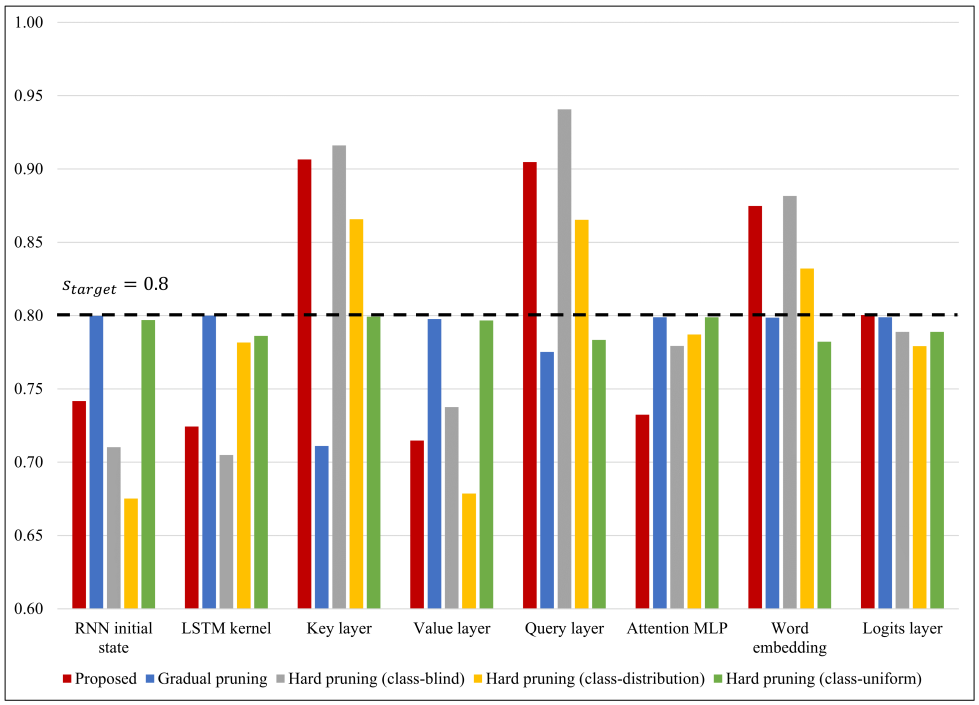
Table 4: Comparison with dense GRU baseline and competing methods. **Bold text** indicates best overall performance. "Gradual" and "Hard" denote methods proposed in [1] and [25].

Approaches	NNZ parameters		MS-COCO test set scores							
	Sparsity	Overall CR	B-1	B-2	B-3	B-4	M	R	C	S
Dense GRU baseline	0	1 ×	71.6	54.8	41.3	31.2	24.7	52.8	94.8	17.8
Hard (class-uniform)	0.800	5 ×	70.9	53.8	40.2	30.1	24.3	52.2	92.8	17.3
Hard (class-distribution)			71.5	54.5	40.9	30.6	24.6	52.6	93.9	17.6
Hard (class-blind)			71.3	54.2	40.7	30.6	24.6	52.6	94.1	17.7
Gradual			71.2	54.3	40.8	30.6	24.4	52.5	92.9	17.3
Our Proposed ($\lambda_s = 5$)			**71.3**	**54.4**	**41.0**	**30.9**	**24.6**	**52.6**	**94.2**	**17.5**
Hard (class-uniform)	0.900	10 ×	70.7	53.5	39.9	29.9	23.9	51.9	90.3	16.9
Hard (class-distribution)			70.7	53.5	40.0	30.0	24.0	51.9	90.5	17.0
Hard (class-blind)			**71.2**	**54.3**	**41.0**	**31.0**	**24.5**	**52.5**	**93.8**	**17.4**
Gradual			70.9	53.8	40.2	30.2	24.0	52.2	91.1	16.9
Our Proposed ($\lambda_s = 5$)			70.9	53.9	40.4	30.3	24.4	52.3	92.3	17.3
Hard (class-uniform)	0.950	20 ×	68.7	51.2	37.7	27.9	22.7	50.4	83.2	15.6
Hard (class-distribution)			68.6	51.1	37.6	27.7	22.7	50.3	83.0	15.6
Hard (class-blind)			70.6	53.4	39.8	29.7	23.8	51.8	89.3	16.7
Gradual			70.3	53.3	39.8	29.8	23.7	51.8	88.3	16.6
Our Proposed ($\lambda_s = 10$)			**71.0**	**54.0**	**40.5**	**30.4**	**24.3**	**52.2**	**92.1**	**17.2**
Hard (class-uniform)	0.975	40 ×	66.4	48.7	35.1	25.3	21.3	48.6	75.1	14.4
Hard (class-distribution)			66.1	48.1	34.3	24.5	21.0	48.3	72.5	14.1
Hard (class-blind)			69.1	51.5	37.8	27.7	22.6	50.5	83.2	15.6
Gradual			68.9	51.7	38.1	28.1	22.9	50.8	83.3	15.8
Our Proposed ($\lambda_s = 20$)			**70.2**	**53.1**	**39.4**	**29.2**	**23.6**	**51.7**	**88.7**	**16.5**

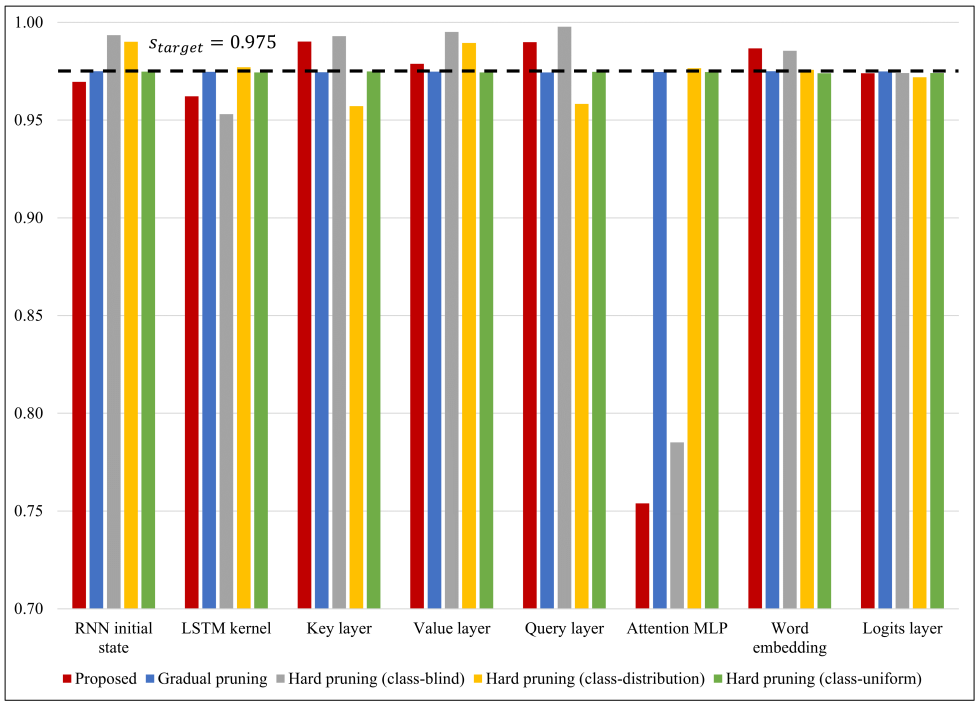
extremely close to that of baselines. Our sparse LSTM model even very slightly outperforms the baseline on some metrics, although we note that the standard deviation for CIDEr score across training runs is around 0.3 to 0.9.

Among the competing methods, we can see that gradual pruning usually outperforms hard pruning, especially at extreme sparsities of 95% and 97.5%. That being said, we can see that class-blind hard pruning is able to produce good results at moderate pruning rates of 80% and 90%, even outperforming gradual pruning. This is especially true for the GRU captioning model where it outperforms all other methods briefly at 90% sparsity, however we note that its performance on LSTM is generally lower. In contrast, our proposed approach achieves good performance on both LSTM and GRU models.

All in all, these results showcase the strength of our proposed method. Across pruning ratios from 80% to 97.5%, our approach consistently maintain relatively good performance when compared to the dense baselines while outperforming magnitude-based gradual and hard pruning methods in most cases.



(a) Pruning ratios at $s_{target} = 0.8$



(b) Pruning ratios at $s_{target} = 0.975$

Figure 3: Layer-wise comparison of final sparsity levels. Best viewed in colour.

Table 5: Comparison with dense LSTM and GRU baselines after CNN fine-tuning.

Approaches	NNZ parameters		MS-COCO test set scores							
	RNN size	Overall CR	B-1	B-2	B-3	B-4	M	R	C	S
H-LSTM [30] + GP [31]	394 K								95.4	
	163 K								93.3	
Dense LSTM baseline	2.62 M	1 ×	73.8	57.5	44.0	33.5	25.8	54.7	102.6	18.7
Our Proposed	725 K	5 ×	73.3	56.8	43.1	32.8	25.6	54.1	100.9	18.5
	402 K	10 ×	73.5	56.9	43.4	33.2	25.5	54.1	101.0	18.6
	205 K	20 ×	73.3	56.8	43.2	32.7	25.4	54.0	100.0	18.4
	101 K	40 ×	73.2	56.9	43.4	32.8	25.1	53.9	99.4	18.0
(Beam size = 2)	101 K	40 ×	73.6	57.0	43.1	32.3	25.1	53.8	99.1	18.1
Dense GRU baseline	1.97 M	1 ×	73.4	56.6	42.9	32.4	25.6	54.0	100.6	18.5
Our Proposed	589 K	5 ×	72.9	56.3	42.8	32.5	25.5	53.8	100.1	18.5
	361 K	10 ×	73.1	56.5	42.8	32.4	25.5	53.9	99.8	18.4
	185 K	20 ×	73.2	56.8	43.3	32.9	25.4	54.0	100.0	18.3
	89 K	40 ×	73.0	56.6	42.9	32.4	25.1	53.9	99.1	18.2

Table 6: Comparison of large-sparse and small-dense LSTM models.

Models	NNZ parameters			MS-COCO test set scores							
	Sparsity	Params.	Overall CR	B-1	B-2	B-3	B-4	M	R	C	S
LSTM-M	0	11.9 M	1 ×	71.8	54.8	41.3	31.1	24.6	52.8	94.3	17.4
LSTM-S	0	2.4 M	5 ×	69.6	52.5	38.5	28.1	22.7	50.9	82.7	15.7
LSTM-M	0.800	2.4 M	5 ×	71.6	54.8	41.4	31.4	24.6	52.8	94.4	17.5
	0.900	1.2 M	10 ×	71.4	54.3	40.8	30.8	24.4	52.4	93.1	17.3
	0.950	0.6 M	20 ×	71.2	54.2	40.7	30.6	24.3	52.4	92.1	17.2
	0.975	0.3 M	40 ×	70.4	53.4	39.8	29.6	23.7	51.8	88.5	16.7

5.3 Effect of fine-tuning

In this section, we investigate the potential impact of fine-tuning the entire captioning model in an end-to-end manner. From Table 5, we can see that model fine-tuning has a performance-recovering effect on the sparse models. This phenomenon is especially apparent on extremely sparse models with sparsity at 97.5%. On both LSTM and GRU models, the drops in performance suffered due to pruning have mostly reduced except for LSTM at 80% sparsity.

Notably, all the pruned GRU models have remarkably similar performance from 80% sparsity up until 97.5%. The score gap between them and the dense GRU baseline is also extremely small, ranging from $+1.5\%$ to -1.5% for both BLEU-4 and CIDEr. For LSTM models, even though the score gap is slightly larger at -0.9% to -3.1% on both BLEU-4 and CIDEr, it is still considerably smaller than without CNN fine-tuning (Table 3).

These results suggest that the Inception-V1 CNN pre-trained on ImageNet is not optimised to provide useful features for sparse decoders. As such, end-to-end fine-tuning together with sparse decoder allows features extracted by the CNN to be adapted where useful semantic information can be propagated through surviving connections in the decoder.

We also provided compression and performance comparison with the closely related work of [30] who utilised GP [31] method to produce sparse H-LSTM for image captioning. For fairness, we also provide scores obtained at CR of $40\times$ using beam size of 2 instead of 3. From the table, we can see that at overall CR of $20\times$ to $40\times$, our sparse models are able to outperform H-LSTM with lower NNZ parameters. This indicates that the effectiveness of our one-shot approach is at least comparable to the iterative process of grow-and-prune.

5.4 Large-sparse versus small-dense

In this section, we show that a large sparse LSTM image captioning model produced via end-to-end pruning is able to outperform a smaller dense LSTM trained normally. The small-dense model denoted as *LSTM-S* has a word embedding size of $q = 64$ dimensions, LSTM size of $r = 128$ units and finally attention MLP size of $a = 96$ units. The results are given in Table 6. From the results, we can see that the small-dense model with $5\times$ fewer parameters performs considerably worse than all the large-sparse models *LSTM-M* across the board.

Notably, we can see that the large-sparse *LSTM-M* model with $40\times$ fewer NNZ parameters still managed to outperform *LSTM-S* with a considerable margin. At equal NNZ parameters, the large-sparse model comfortably outperforms the small-dense model. This showcases further the strength of model pruning and solidifies the observations made in works on RNN pruning [2, 1].

Table 7: Comparison on caption uniqueness and length with dense baselines on MS-COCO test set.

Approaches	NNZ parameters		Before fine-tune		After fine-tune	
	Sparsity	Overall CR	Unique (%)	Average length	Unique (%)	Average length
Dense LSTM baseline	0	1 ×	42.1	9.09	46.0	9.07
Our Proposed	0.800	5 ×	42.5	9.11	45.3	9.12
	0.900	10 ×	41.9	9.07	46.8	9.09
	0.950	20 ×	41.9	9.05	47.0	9.05
	0.975	40 ×	44.4	8.99	48.4	8.97
Dense GRU baseline	0	1 ×	42.4	9.15	46.9	9.14
Our Proposed	0.800	5 ×	43.1	9.13	47.3	9.16
	0.900	10 ×	43.0	9.09	46.2	9.13
	0.950	20 ×	42.7	9.07	46.9	9.06
	0.975	40 ×	42.0	8.94	49.1	8.98

5.5 Caption uniqueness and length

In this section, we explore the potential effects of our proposed end-to-end pruning on the uniqueness and length of the generated captions. As pruning reduces the complexity and capacity of the decoder considerably, we wish to see if the sparse models show any signs of training data memorisation and hence potentially overfitting. In such cases, uniqueness of the generated captions would decrease as the decoder learns to simply repeat captions available in the training set. A generated caption is considered to be unique if it is not found in the training set.

From Table 7, we can see that despite the heavy reductions in NNZ parameters, the uniqueness of generated captions have not decreased. On the contrary, more unseen captions are being generated at higher levels of sparsity and compression. On the other hand, we can see that the average lengths of generated captions peaked at 80% sparsity in most cases and then decrease slightly as sparsity increase. That being said, the reductions in caption length are minimal ($+0.5\%$ to -2.3%) considering the substantial decoder compression rates of up to $40\times$.

Together with the good performance shown in Table 3 and 4, these results indicate that our approach is able to maintain both the variability of generated captions and their quality as measured by the metric scores.

5.6 Layer-wise sparsity comparison

Finally, we visualise the pruning ratio of each decoder layers when pruned using the different methods listed in Sec. 5.2. Among the approaches, both gradual and class-uniform pruning produces the same sparsity level across all the layers. To better showcase the differences in layer-wise pruning ratios, we decided to visualise two opposite ends in which the first has a relatively moderate sparsity of 80% while the other has a high sparsity of 97.5%.

In both Fig. 3a and 3b, we denote the decoder layers as follows: "RNN initial state" refers to W_I in Eq. 2; "LSTM kernel" is the concatenation of all gate kernels in LSTM (i.e. input, output, forget, cell); "Key", "Value" and "Query" layers refer to projection layers in the attention module (see [61] for details); "Attention MLP" is the second layer of the 2-layer attention MLP; and finally "Word" and "Logits" refer to the word embedding matrix E_w in Eq. 5 and E_o in Eq. 3 respectively.

From the figures, we can see that our proposed pruning method consistently prune "attention MLP" layer the least. This is followed by "LSTM kernel" and "Value" layers where they generally receive lesser pruning compared to others. On the flip side, "Key" and "Query" layers were pruned most heavily at levels often exceeding the targeted pruning rates. Finally, "Word embedding" consistently receives more pruning than "Logits layer". This may indicate that there exists substantial information redundancy in the word embeddings matrix as noted in works such as [37, 40, 62].

6 Conclusion and Future Work

In this work, we have investigated the effectiveness of model weight pruning on the task of image captioning with visual attention. In particular, we proposed an end-to-end pruning method that performs considerably better than competing methods at maintaining captioning performance while maximising compression rate. Our single-shot approach is simple and fast to use, provides good performance, and its sparsity level is easy to tune. Moreover, we have demonstrated by pruning decoder weights during training, we can find sparse models that performs better than

dense counterparts while significantly reducing model size. Our results pave the way towards deployment on mobile and embedded devices due to their small size and reduced memory requirements. In the future, we wish to investigate the generalisation capability of end-to-end pruning when applied on Transformer models [61]. We would also like to extend our method to other CV and NLP tasks including image classification, language modelling and natural language translation.

Bibliography

[1] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[2] S. Narang, E. Elsen, G. Diamos, and S. Sengupta, "Exploring sparsity in recurrent neural networks," *arXiv preprint arXiv:1704.05119*, 2017.

[3] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[4] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[5] G. Diamos, S. Sengupta, B. Catanzaro, M. Chrzanowski, A. Coates, E. Elsen, J. Engel, A. Hannun, and S. Satheesh, "Persistent RNNs: Stashing recurrent weights on-chip," in *International Conference on Machine Learning*, 2016, pp. 2024–2033.

[6] Y. Bengio, N. L. Roux, P. Vincent, O. Delalleau, and P. Marcotte, "Convex neural networks," in *Advances in neural information processing systems*, 2006, pp. 123–130.

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[8] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[9] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[10] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[11] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 525–542.

[12] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[13] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *IEEE international conference on neural networks*. IEEE, 1993, pp. 293–299.

[14] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in neural information processing systems*, 1989, pp. 107–115.

[15] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE transactions on neural networks*, vol. 1, no. 2, pp. 239–242, 1990.

[16] Y. Chauvin, "A back-propagation algorithm with optimal use of hidden units," in *Advances in neural information processing systems*, 1989, pp. 519–526.

[17] M. Ishikawa, "Structural learning with forgetting," *Neural networks*, vol. 9, no. 3, pp. 509–521, 1996.

[18] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Advances In Neural Information Processing Systems*, 2016, pp. 1379–1387.

[19] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems*, 2015, pp. 2575–2583.

[20] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2498–2507.

[21] B. Dai, C. Zhu, B. Guo, and D. Wipf, "Compressing neural networks using the variational information bottleneck," in *International Conference on Machine Learning*, 2018, pp. 1143–1152.

[22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," *arXiv preprint arXiv:1608.08710*, 2016.

[23] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[24] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9194–9203.

[25] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2016, pp. 291–301.

[26] N. Lee, T. Ajanthan, and P. H. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340*, 2018.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 784–800.

[29] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[30] X. Dai, H. Yin, and N. K. Jha, "Grow and prune compact, fast, and accurate LSTMs," *arXiv preprint arXiv:1805.11797*, 2018.

[31] X. Dai, H. Yin, and N. Jha, "NeST: A neural network synthesis tool based on a grow-and-prune paradigm," *IEEE Transactions on Computers*, 2019.

[32] H. Yu, S. Edunov, Y. Tian, and A. S. Morcos, "Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP," *arXiv preprint arXiv:1906.02768*, 2019.

[33] S. Srinivas, A. Subramanya, and R. Venkatesh Babu, "Training sparse neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 138–145.

[34] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.

[35] Z. Lu, V. Sindhwani, and T. N. Sainath, "Learning compact recurrent neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5960–5964.

[36] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "FastGRNN: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," in *Advances in Neural Information Processing Systems*, 2018, pp. 9017–9028.

[37] K. Shi and K. Yu, "Structured word embedding for low memory neural network language model," in *Interspeech*, 2018, pp. 1254–1258.

[38] X. Li, T. Qin, J. Yang, and T.-Y. Liu, "LightRNN: Memory and computation-efficient recurrent neural networks," in *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 4385–4393.

[39] S. N. Parameswaran, "Exploring memory and time efficient neural networks for image captioning," in *National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics*. Springer, 2017, pp. 338–347.

[40] J. H. Tan, C. S. Chan, and J. H. Chuah, "COMIC: Towards a compact image captioning model with attention," *IEEE Transactions on Multimedia*, 2019.

[41] F. Gao, L. Wu, L. Zhao, T. Qin, X. Cheng, and T.-Y. Liu, "Efficient sequence learning with group recurrent networks," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 799–808.

[42] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning (ICML)*, 2015, pp. 2048–2057.

[43] K. Fu, J. Jin, R. Cui, F. Sha, and C. Zhang, "Aligning Where to See and What to Tell: Image Captioning with Region-based Attention and Scene-specific Contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2321–2334, 2017.

[44] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6077–6086.

[45] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[46] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[47] H. Zhou, J. Lan, R. Liu, and J. Yosinski, "Deconstructing lottery tickets: Zeros, signs, and the Supermask," *arXiv preprint arXiv:1905.01067*, 2019.

[48] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[49] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[51] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[54] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.

[55] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3128–3137.

[56] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, 2002, pp. 311–318.

[57] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, vol. 29, 2005, pp. 65–72.

[58] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text summarization branches out: Proceedings of the ACL-04 workshop*, vol. 8, 2004, pp. 1–8.

[59] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "CIDEr: Consensus-based image description evaluation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4566–4575.

[60] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "SPICE: Semantic propositional image caption evaluation," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 382–398.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[62] R. Shu and H. Nakayama, "Compressing word embeddings via deep compositional code learning," *arXiv preprint arXiv:1711.01068*, 2017.