# DOUBLY SPARSE: SPARSE MIXTURE OF SPARSE EXPERTS FOR EFFICIENT SOFTMAX INFERENCE

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Computations for the softmax function in neural network models are expensive when the number of output classes is large. This can become a significant issue in both training and inference for such models. In this paper, we present Doubly Sparse Softmax (DS-Softmax), *Sparse Mixture of Sparse of Sparse Experts*, to improve the efficiency for softmax inference. During training, our method learns a two-level class hierarchy by dividing entire output class space into several partially overlapping experts. Each expert is responsible for a learned subset of the output class space and each output class only belongs to a small number of those experts. During inference, our method quickly locates the most probable expert to compute small-scale softmax. Our method is learning-based and requires no knowledge of the output class partition space a priori. We empirically evaluate our method on several real-world tasks and demonstrate that we can achieve significant computation reductions without loss of performance.

## 1 INTRODUCTION

Deep learning models have demonstrated impressive performance in many classification problems (LeCun et al., 2015). In many of these models, softmax function/layer is commonly used to produce categorical distributions over the output space. Due to its linear complexity, computation for softmax layer can become a bottleneck with large output dimensions, such as language modelling (Bengio et al., 2003), neural machine translation (Bahdanau et al., 2014) and face recognition (Sun et al., 2014). In some models, softmax contributes to more than 95% computation. This becomes more of an issue when computational resource is limited, like mobile devices (Howard et al., 2017).

Many methods have been proposed to reduce softmax complexity for both training and inference phases. For training, the goal is reduce the training time. Sampling based (Gutmann & Hyvärinen, 2012) and hierarchical based methods (Goodman, 2001; Morin & Bengio, 2005) were introduced. D-Softmax (Chen et al., 2015) and Adaptive-Softmax (Grave et al., 2016), construct two level-hierarchies for the output classes based on the unbalanced word distribution for training speedup. The hierarchies used in these methods are either pre-defined or constructed manually, which can be unavailable or sub-optimal. Unlike training, in inference, our goal is not to computing the exact categorical distribution over the whole vocabulary, but rather to search for top-K classes accurately and efficiently. Existing work (Shrivastava & Li, 2014; Shim et al., 2017; Zhang et al., 2018) on this direction focus on designing efficient approximation techniques to find the top-K classes given a *trained* model. Detailed discussions of related works are to be found in Section 4.

Our work aims to improve the inference efficiency of the softmax layer. We propose a novel Doubly Sparse softmax (DS-Softmax) layer. The proposed method is motivated by (Shazeer et al., 2017), and it learns a two-level overlapping hierarchy using *sparse* mixture of *sparse* experts. Each expert is trained to only contain a small subset of entire output class space, while each class is permitted to belong to more than one expert. Given a set of experts and an input vector, the DS-Softmax first selects the top expert that is most related to the input (in contrast to a dense mixture of experts), and then the chosen expert could return a scored list of most probable classes in it sparse subset. This method can reduce the linear complexity in original softmax significantly since it does not need to consider the whole vocabulary.

We conduct experiments in different real tasks, ranging from language modeling to neural machine translation. We demonstrate our method can reduce softmax computation dramatically without loss of

prediction performance. For example, we achieved more than 23x speedup in language modelling and 15x speedup in translation with similar performances. Qualitatively, we also demonstrate that learned two-level overlapping hierarchy is usually semantically meaningful on natural language modelling tasks.

## 2 DS-SOFTMAX: SPARSE MIXTURE OF SPARSE EXPERTS

### 2.1 BACKGROUND

Before introducing our method, we first provide an overview of the background.

**Hierarchical softmax.** Hierarchical softmax uses a tree to organize output space where a path represents a class (Morin & Bengio, 2005). There are a few ways to construct such hierarchies. Some previous works(Morin & Bengio, 2005; Chen et al., 2015; Grave et al., 2016) focus on building hierarchies with prior knowledge. Other approaches, like Mnih & Hinton (2009), performed clustering on embeddings to construct a hierarchy. Our work aims to learn a a two-level hierarchy while the major difference is that we allow overlapping in the learned hierarchy.

**Sparsely-gated mixture-of-experts.** Shazeer et al. (2017) designed a sparsely gated mixture of experts model so that outrageously large networks can achieve significantly better performance in language modelling and translation. They borrowed conditional computation idea (Bengio et al., 2015) to keep similar computation even though the number of parameters has been increased dramatically. Their proposed sparsely-gated Mixture of Experts (MoE) only use a few experts selected by sparsely gating network for computation on each example. The original MoE cannot speedup softmax computation but serves as an inspiration for our model design.

**Group lasso.** Group lasso has been commonly used to reduce effective features in linear model (Friedman et al., 2010; Meier et al., 2008). Recently it has been applied in neural network for regularization (Scardapane et al., 2017) and convolutional deep neural network speedup (Wen et al., 2016). It has been demonstrated as an effective method to reduce number of nodes in neural network. In this work, we use group lasso to sparsify the experts.

### 2.2 OUR METHOD

Goodman (2001) studied a two-level hierarchy for language modeling, where each word belongs to one unique cluster. (A "cluster" here refers to a cluster of words.) From this perspective, our method can be as an extension of their method to allow overlapping hierarchy. This is because, in language modeling, it is often difficult to exactly assign a word to a single cluster. For example, if we want to predict next word of "I want to eat ___" and one possible correct answer is "cookie", we can quickly notice that possible answer belongs to something eatable. So if we only search right answer inside words with the eatable property, we can dramatically increase the efficiency. On the other, though "cookie" is one of the correct answers, it might also like appear under some non-eatable context, such as "a piece of data" in computer science. Thus, a two-level overlapping hierarchy can naturally accommodate word homonyms like this by allowing each word to belong to more than one cluster. We believe this observation is likely to be true in other applications besides language modeling.

Doubly Sparse softmax (DS-Softmax) is designed to capture such overlapped two-level hierarchy among output classes. In DS-Softmax, the first level is the *sparse* mixture and second level contains several *sparse* experts. (Here an expert can be thought as a similar concept as cluster.) The sparse mixture is to choose the right expert/cluster while sparse experts are responsible to separate full output space into multiple, overlapped and small class clusters. The design of mixture gating is inspired by Shazeer et al. (2017) but each expert in their model needs to search whole output space, while DS-Softmax only searches a small subset. This becomes much faster given large output space.

**Sparse gating.** The first level of sparsification is a sparse gating mechanism inspired by Shazeer et al. (2017), which is to design to choose the right experts. The sparse gating outputs a sparse activation over a set of experts. For faster inference purpose, only the top-one expert is chosen here. One major difference comparing to Shazeer et al. (2017) is described as follows. Suppose

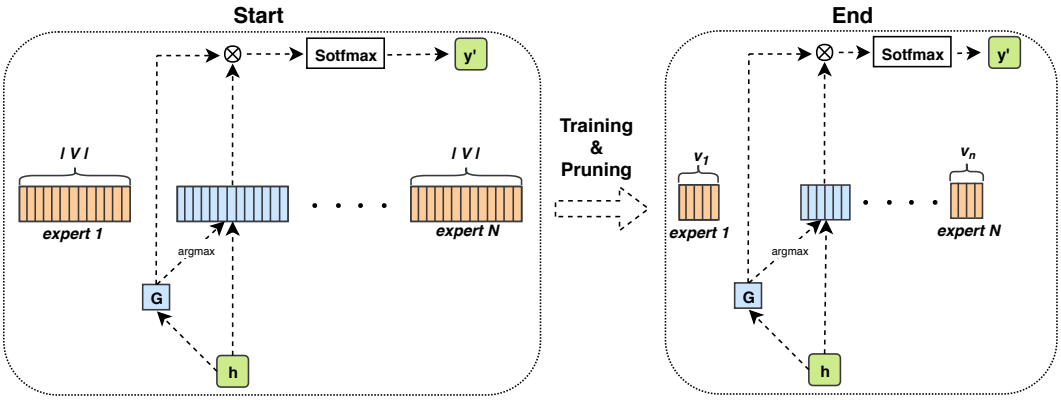



Figure 1: Overview of DS-Softmax. Initial model is similar to sparsly gating mixture of experts model. After pruning, each expert will only consists partial outputs $v_n$ instead of $|V|$.

we have $K$ experts . Given input activation vector $h \in \mathbb{R}^d$, gating values $G_k(h)$, $k = 1, ..., K$, are normalized prior to the selection as shown in Eq. 1 and then we choose the gate with the largest value $g_k = \max_i G_i(h)$ and set all other gates to be zero. Also, corresponding $k$-th expert is chosen.

$$
\begin{aligned}
G_k(h) &= \frac{\exp(W_k^g h)}{\sum_{k'} \exp(W_{k'}^g h)}, \\
g_k &= \begin{cases} G_k(h), & \text{if } k = \arg\max_i G_i(h), \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{1}
$$

This allows gradient to be back-propagated to whole $W^g$ instead of $W_k^g$ only, $W^g \in \mathbb{R}^{K \times d}$. In Shazeer et al. (2017), normalization is done after top-K experts are selected. We can not do that since we only choose top-1 expert since it will carry no gradient information since it becomes constant 1. Given the sparse gate, we compute the probability of class $c$ as,

$$
O(h) = p(c|h) = \frac{\exp(\sum_k g_k W_{(c,k)}^e h)}{\sum_{c'} \exp(\sum_k g_k W_{(c',k)}^e h)}, \tag{2}
$$

where $W_{(c,k)}^e \in \mathbb{R}^d$ is softmax embedding weight vector for class $c$ in expert $k$. Note that only one $g_k$ (the chosen expert) is nonzero in the formulation above. The gating values can be interpreted as an inverse temperature term for final categorical distribution produced by the chosen expert $k$ Hinton et al. (2015), shown in Eq. 2. A smaller $g_k$ gives a more uniform distribution and larger $g_k$ corresponds to a sharper one.

**Sparse experts with group lasso.** The second level sparsification is the sparse experts, which output a categorical distribution for only a subset output classes. To sparsify each expert, we apply group lasso loss to restrain the $W_{(c,k)}^e$, shown in Eq. 3. Then, pruning is carried out for $W_{(c,k)}^e$ during training with $\gamma$ is a lasso threshold according to Eq. 4.

$$
\mathcal{L}_{lasso} = \sum_k \sum_c \|W_{(c,k)}^e\|_2, \tag{3}
$$

$$
W_{(c,k)}^e = \begin{cases} W_{(c,k)}^e, & \text{if } \|W_{(c,k)}^e\|_2 > \gamma, \\ 0, & \text{otherwise.} \end{cases} \tag{4}
$$

**Loading Balance.** We denote the sparsity percentage out of full softmax in $k$-th expert as $\text{sparsity}_k$ and proportion of $k$-th expert activated as $\text{utilization}_k$. Then, the overall speedup compared to the full softmax can be calculated as as $1/\sum_k(\text{utilization}_k * \text{sparsity}_k)$. Thus, better utilization is essential for speedup as well. For example, there is no speedup if the expert with full output space is always chosen.

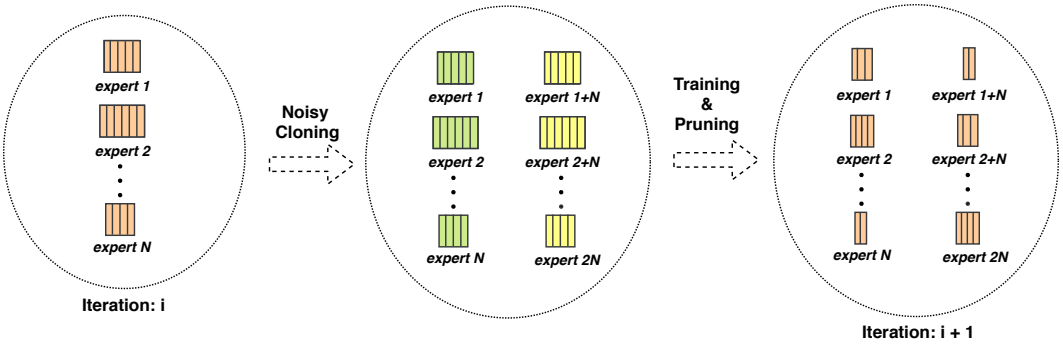



Figure 2: The mitosis training scheme: the sparsity is inherited when parent experts produce offspring, reducing the memory requirements for training with more experts.

**Algorithm 1**

**Initialization:** Let $x$ be the input, $y$ be the corresponding label, $H$ be the pretrained function, $V$ be the output dimension and $D(y', y)$ be an arbitrarily distance function. Set $W^e \leftarrow$ parameters for experts and $W^g \leftarrow$ parameters for the gating network. The hyper-parameter $t$ denotes target performance.
**while** $epoch < \text{Max}$ **do**
  $epoch = epoch + 1$
  $\mathcal{L}_{task} = D(O(H(x)), y)$
  $\mathcal{L}_{all} = \mathcal{L}_{task} + \lambda_{load}\,\mathcal{L}_{load} + \lambda_{lasso}\,\mathcal{L}_{lasso} + \lambda_{expert}\,\mathcal{L}_{expert}$
  $W^e = W^e - \alpha \frac{\partial}{\partial W^e} \mathcal{L}_{all}(x, y; W^e, W^g)$
  $W^g = W^g - \alpha \frac{\partial}{\partial W^g} \mathcal{L}_{all}(x, y; W^e, W^g)$
  **if** $L_{task} < t$ **then**
    **for all** $W^e_{(c,k)} \in W^e$ **do**
      $W^e_{(c,k)} = 0$, if $\|W^e_{(c,k)}\|_2 < \gamma$

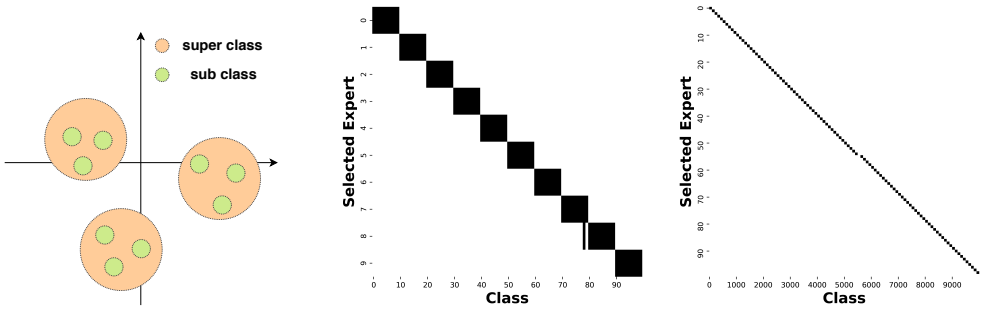
We borrow a similar loading balance function from Shazeer et al. (2017) in Eq. 5. It encourages the utilization percentage of each expert to be balanced by maximizing the coefficient of variation (CV) for gating outputs. In addition, to encourage the exclusiveness of classes, we incorporate group lasso loss on expert level where each class should only exist in only one expert as shown in Eq. 6.

$$\mathcal{L}_{load} = -\text{CV}\left(\sum_{h \in H(x)} G(h)\right), \tag{5}$$

$$\mathcal{L}_{expert} = \sum_k \sqrt{\sum_c \|W^e_{(c,k)}\|_2^2}. \tag{6}$$

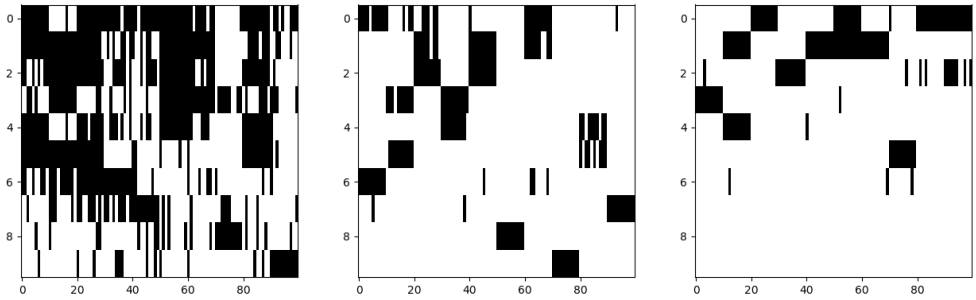
**Mitosis training.** Memory might become a bottleneck during training if we initialize all experts with full softmax. Therefore, we design one training scheme, called mitosis training, to reduce memory requirement. The method is to initialize with a smaller model (fewer number of experts) and then gradually breed to a bigger one after noisy cloning shown in Fig. 2. For each cloning, the sparsity is inherited so that less memory is required. For example, in one of our experiments, we only need 3.25x memory with 64 experts compared to a full softmax implementation.

**The final training algorithm.** Our final training objective, $\mathcal{L}_{all}$, consists of a combination of the related contributions discussed above. We describe our training procedure in Algorithm 1.




(a) Synthetic data (b) Results on 10 * 10 (c) Results on 100 * 100

Figure 3: (a) Illustration of data generation. (b) and (c) Results on discovered sparse experts on 10x10 and 100x100 datasets. The x-axis indicates class and y-axis shows the selected expert for handling this class. The order of x-axis is arranged through their super class information. For example, each 10 sub classes are belonged to one super classes in (b).



(a) No Group Lasso (b) No Expert Group Lasso (c) No Balancing

Figure 4: Ablation analysis of each loss component by removing it. (a), (b) and (c) illustrate the model trained without group lasso, expert level group lasso and balancing factor respectively.

## 3 EXPERIMENTS

We evaluate the proposed method on both synthetic and real tasks. For the synthetic task, our goal is to demonstrate that our learning method could discover the hidden two-level hierarchy automatically. We also evaluate both theoretical speedup (FLOPs) and real device speedup (latency on CPU) on three different real tasks: natural language modelling, neural machine translation and Chinese handwritten character recognition.

All layers except the DS-Softmax layer are pre-trained in all tasks. For hyper-parameters, $\lambda_{load}$ is fixed as large constant for all experiments. $\lambda_{lasso}$ and $\lambda_{expert}$ share same value but need to be tuned for each task on a validation set. Lasso threshold is also needed to tune to achieve best performance for each task.

### 3.1 SYNTHETIC TASK

One two-level hierarchy synthetic dataset is illustrated Fig 3a. The details of data generation is shown in Appendix A. Two different sizes are evaluated, 10x10 (super classes x sub classes) and 100x100. The result is illustrated in Fig. 3b and Fig. 3c. We found our DS-Softmax can perfectly capture the hierarchy. For sanity check and visualization purposes, the ground-truth two hierachy in the synthetic data does not have overlappings. We did further analysis on the results on 10x10 synthetic as shown in Fig 4 to study the effects of group lasso and balancing. As we can see, both are important to our model.

| Task | Method | Testing Accuracy | | | FLOPs |
|---|---|---|---|---|---|
| | | **Top 1** | **Top 5** | **Top 10** | |
| PTB (10,000) | Full Softmax | 0.252 | 0.436 | 0.515 | - |
| | DS-8 | 0.257 | 0.448 | 0.530 | 2.84x |
| | DS-16 | 0.258 | 0.450 | 0.529 | 5.13x |
| | DS-32 | 0.259 | 0.449 | 0.529 | 9.43x |
| | DS-64 | 0.258 | 0.450 | 0.529 | 15.99x |
| wiki-2 (33,278) | Full Softmax | 0.257 | 0.456 | 0.533 | - |
| | DS-8 | 0.259 | 0.459 | 0.536 | 3.52x |
| | DS-16 | 0.264 | 0.469 | 0.547 | 6.58x |
| | DS-32 | 0.260 | 0.460 | 0.535 | 11.59x |
| | DS-64 | 0.259 | 0.458 | 0.533 | 23.86x |

Table 1: Word level natural language modelling results on PTB and WikiText-2. The output dimensions are 10,000 and 33,278 respectively. The top1, top5 and top10 accuracies is used as metric.

| Task | Method | Bleu Score | FLOPs |
|---|---|---|---|
| IWSLT En-Ve (7,709) | Full Softmax | 25.2 | - |
| | DS-8 | 25.3 | 4.38x |
| | DS-16 | 25.1 | 6.08x |
| | DS-32 | 25.4 | 10.69x |
| | DS-64 | 25.0 | 15.08x |

Table 2: Speedup comparison for neural machine translation on IWSLT English to Vietnamese and the vocabulary size is 7,709. Bleu score on testing with greedy searching is used as metric.

### 3.2 Language Modeling

Language modelling usually contains a large output dimension. We use two standard datasets for word level language modelling: PennTree Bank (PTB) [Marcus et al. (1994)] and WikiText-2 [Merity et al. (2016)]. The output dimensions are 10,000 and 33,278 respectively. Standard two-layers LSTM model [Gers et al. (1999)] with 200 hidden size is used [1]. Top 1, Top 5 and Top 10 accuracies on testing datasets are used as metric rather than perplexity. The accuracy is common metric [Chen et al. (1998)] in natural language modelling especially in real application when extrinsic reward is given, such as voice recognition. We demonstrate that 15.99x and 23.86x times speed up can be achieved with 64 experts without loss of accuracy shown in Table 1.[2]

### 3.3 Neural Machine Translation

As language related task, neural machine translation task is usually used for softmax speedup evaluation. We use IWSLT English to Vietnamese dataset (Luong & Manning, 2015) and evaluate performance by BLEU score (Papineni et al., 2002) with greedy searching. The BLEU is assessed on testing dataset. The vanilla softmax model is seq2seq (Sutskever et al., 2014) and implemented using tensorflow (Abadi et al.) [3]. Dataset preprocessing is same as original implementation and number of output words is 7,709.

### 3.4 Chinese Character Recognition

For Chinese handwriting character recognition task, we ues the offline and filtered CASIA dataset (Liu et al., 2011). We chose this task because these type of models are likely to be used on resource-limited devices such as mobile phones. CAISA is popular Chinese character recognition dataset with more than four thousand characters. We removed some special characters for better vanilla softmax model performance. Two thirds of the data is chosen for training and rest for testing. Table 3 shows the results and we can achieve signifcant speedup on this task too.

[1] https://github.com/tensorflow/models/tree/master/tutorials/rnn/ptb

[2] We require that at least one copy for each word is kept given some expert. Otherwise we can achieve more than 80x speed up without loss of accuracy at the cost of not modeling some very low frequency words.

[3] https://github.com/tensorflow/nmt

| Task | Method | Accuracy | FLOPs |
|---|---|---|---|
| CASIA (3,740) | Full Softmax | 90.6 | - |
| | DS-8 | 90.8 | 1.77x |
| | DS-16 | 90.2 | 2.82x |
| | DS-32 | 89.9 | 4.72x |
| | DS-64 | 90.1 | 6.91x |

Table 3: Result on Chinese handwritten character recognition. The accuracy in testing dataset is reported as the performance for each model.

| Task | Full | | DS-64 (Ours) | | | SVD-5 | | | SVD-10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Value | ms | Value | FLOPs | ms | Value | FLOPs | ms | Value | FLOPs | ms |
| PTB | 0.252 | 0.73 | **0.258** | **15.99x** | **0.05** | 0.249 | 6.67x | 0.12 | 0.251 | 5.00x | 0.18 |
| Wiki-2 | 0.257 | 3.07 | **0.259** | **23.86x** | **0.12** | 0.253 | 7.35x | 0.43 | 0.255 | 5.38x | 0.63 |
| En-Ve | **25.2** | 1.91 | 25.0 | **15.08x** | **0.12** | 25.0 | 6.77x | 0.39 | 25.1 | 5.06x | 0.42 |
| CASIA | **0.906** | 1.61 | 0.901 | **6.91x** | **0.25** | 0.899 | 3.00x | 0.59 | 0.902 | 2.61x | 0.68 |

Table 4: Comparison with SVD-softmax on real device latency. The "ms" indicates the latency in microseconds. Bold fonts indicate better results.

### 3.5 Real device comparison

Real device experiments were conducted on machine with Two Intel(R) Xeon(R) CPU @ 2.20GHz and 16G memory. All tested models are re-implemented using numpy. Two configurations of SVD-Softmax Shim et al. (2017) are evaluated, SVD-5 and SVD-10. They use top 5% and 10% dimension for final evaluation in their preview window and window width is 16. Indexing and sorting are computationally heavy for SVD-softmax with our numpy implementation. For a fair comparison, we report latency without sorting and indexing for SVD-softmax. However, regards to full softmax and DS-Softmax, full latency is reported. Latency of each sample is shown in table 4. According to the result, our DS-Softmax can achieve not only better FLOPs speedup but also much better performance on latency.

### 3.6 Mitosis training

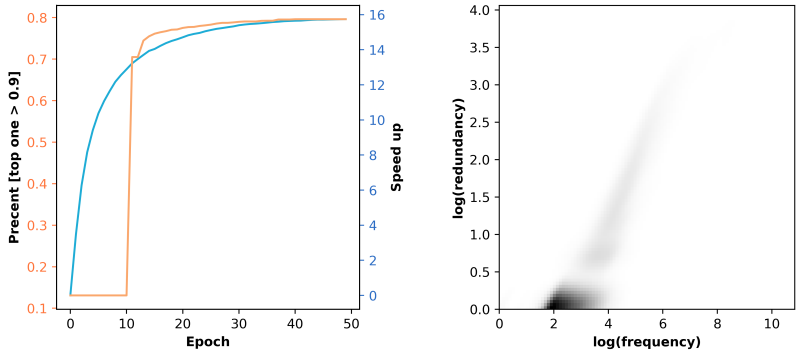
In this paragraph, we present some details on the mitosis training scheme on PTB language modelling task. The model is initialized with 2 experts, and clone to 4, 8, 16, 32 and 64 experts sequentially. As demonstrated in Appendix B, the model only requires at most 3.25x memory to train DS-64 model and achieve similar performance, significantly smaller than original 64-fold memory. In addition, the final model only has less than 1.5 redundancy. This indicates that memory required for inference is relatively small.

### 3.7 Qualitative analysis of sparsity

We also investigate how the sparsity changes during training on language modelling task with PTB dataset, DS-64. For first layer sparsity , we would like inspect how certain experts are chosen. As shown in Fig 5a, certainty is higher after longer training and higher speedup as expected. We use percentage of high gating value as the indication of certainty, where more high gating value means higher certainty. For second layer, we interrogate the left words for each expert manually, and the smallest expert is chosen for validation. We found the words in that expert are actually semantically related. Also, we found the frequency is highly correlated with redundancy that high frequent words appear in more experts. This is a similar phenomenon like the topic models in (Blei et al., 2003; Wallach, 2006). Detail is shown in Appendix C.

## 4 Related Work

In many use cases of softmax inference, such as language modeling or recommender systems, only top-K most probable classes are needed. And perhaps to this reason, many existing methods for speeding up softmax inference focus on approximately finding top-K classes given the already




(a) Uncertainty and speedup

(b) Frequency and redundancy

Figure 5: (a) Correlation between uncertainty and training time. We denote uncertainty by the proportion of high gating activation, which is higher than 0.9. (b) Correlation between word frequency and redundancy. Redundancy means the number of appearance for one word in experts. Darker color indicates higher density.

trained softmax layer. We term them as post-processing methods. For example, the SVD-Softmax (Shim et al., 2017) method decomposes learned softmax embedding matrix using singular value decomposition and approximates top-K classes through a smaller preview matrix. Approximation methods incorporating traditional searching methods are also popular including Locality Sensitive Hashing (LSH) (Shrivastava & Li, 2014; Maddison et al., 2014; Mussmann et al., 2017; Spring & Shrivastava, 2017) and small word graph (Zhang et al., 2018). Those post-processing methods depends on a well-learned softmax and might become costly when high precision is mandatory.

Other related works proposed to speed up inference by changing training scheme and we term them as learning-based methods. Our method can be regarded as this category. For example, differentiated softmax (Chen et al., 2015) and adaptive softmax (Grave et al., 2016) can speed up both training and inference by partially activating parameters. Their methods use prior knowledge such as unbalanced distribution of word frequencies to obtained a non-overlapping hierarchy for the output class space. The basic ideas were also derived form the original hierarchical methods (Morin & Bengio, 2005; Mnih & Hinton, 2009; Mikolov et al., 2011; Le et al., 2011). Combinations of learning-based and post-processing are also proposed and studied in Levy et al. (2018). Our proposed method learns a two-level hierarchy of experts but each class can belong to more than one experts.

## 5 Conclusion

In this paper, we present *doubly sparse: sparse mixture of sparse experts* for efficient softmax inference. Our method is trained end-to-end. It learns a two-level overlapping class hierarchy. Each expert is learned to be only responsible for a small subset of the output class space. During inference, our method first identifies the responsible expert and then perform a small scale softmax computation just for that expert. Our experiments on several real-world tasks have demonstrated the efficacy of our proposed method.

## References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Stanley F Chen, Douglas Beeferman, and Ronald Rosenfeld. Evaluation metrics for language models. In *DARPA Broadcast News Transcription and Understanding Workshop*, pp. 275–280. Citeseer, 1998.

Welin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906*, 2015.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

Joshua Goodman. Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pp. 561–564. IEEE, 2001.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for gpus. *arXiv preprint arXiv:1609.04309*, 2016.

Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13 (Feb):307–361, 2012.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. Structured output layer neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pp. 5524–5527. IEEE, 2011.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Daniel Levy, Danlu Chan, and Stefano Ermon. Lsh softmax: Sub-linear learning and inference of the softmax layer in deep architectures. 2018.

Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. Casia online and offline chinese handwriting databases. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp. 37–41. IEEE, 2011.

Minh-Thang Luong and Christopher D. Manning. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam, 2015.

Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, pp. 3086–3094, 2014.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pp. 114–119. Association for Computational Linguistics, 1994.

Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5528–5531, 2011.

Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pp. 1081–1088, 2009.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pp. 246–252. Citeseer, 2005.

Stephen Mussmann, Daniel Levy, and Stefano Ermon. Fast amortized inference and learning in log-linear models with randomly perturbed nearest neighbor search. *arXiv preprint arXiv:1707.03372*, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Kyuhong Shim, Minjae Lee, Iksoo Choi, Yoonho Boo, and Wonyong Sung. Svd-softmax: Fast softmax approximation on large vocabulary neural networks. In *Advances in Neural Information Processing Systems*, pp. 5463–5473, 2017.

Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pp. 2321–2329, 2014.

Ryan Spring and Anshumali Shrivastava. A new unbiased and efficient class of lsh-based samplers and estimators for partition function computation in log-linear models. *arXiv preprint arXiv:1703.05160*, 2017.

Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1891–1898, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Hanna M Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pp. 977–984. ACM, 2006.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2074–2082, 2016.

Minjia Zhang, Xiaodong Liu, Wenhan Wang, Jianfeng Gao, and Yuxiong He. Navigating with graph representations for fast and scalable decoding of neural language models. *arXiv preprint arXiv:1806.04189*, 2018.

## A SYNTHETIC DATA GENERATION

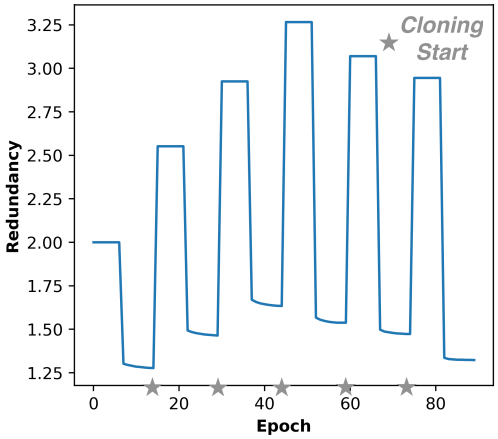
We design the data generation process as follows: (1) sample super classes $C_i^{\text{super}}$ from given multivariate Gaussian (2) sample sub classes $C_j^{\text{sub}}$ from multivariate Gaussian with mean as $C_i^{\text{super}}$ and a smaller variance (3) The actual input $X_j^{\text{input}}$ is sampled around $C_j^{\text{sub}}$ and even smaller variance, and the corresponding label is $j$. The detail of implementation is shown in Eq. 7.

$$C_i^{\text{super}} \sim \mathcal{N}(0, d^3 I), \quad C_j^{\text{sub}} \sim \mathcal{N}(C_i^{\text{super}}, d^2 I), \quad X_j^{\text{input}} \sim \mathcal{N}(C_j^{\text{sub}}, dI), \tag{7}$$

where we choose $d = 10$ for our experiments.

## B MITOSIS TRAINING

In mitosis training, the model is first initialized with 2 experts, and then is cloned to have to 4, 8, 16, 32 and 64 experts gradually. Cloning happens for each 15 epochs and pruning starts 10 epochs after cloning. The result is illustrated in Fig. 6. As we can see, only less than 3.25x memory is required to train DS-64 model instead of 64x memory.




(a) Mitosis Training Result

Figure 6: Illustration of memory requirement needed to train DS-64 starting with DS-2

## C ANALYSIS OF EXPERTS

We choose the smallest expert for qualitative analysis. High frequency words are filtered out and 64 words remain. We found these words can be classified as three major groups: *money, time* and *comparison* related. For example, "million", "billion", "trillion" appear as money related group. All the weekday words are identified as time related words. The detail is shown as following:

- **Money related**: million, billion, trillion, earnings, share, rate, stake, bond, cents, bid, cash, fine, payable
- **Time related**: years, while, since, before, early, late, yesterday, annual, currently, monthly, annually, monday, tuesday, wednesday, thursday, friday
- **Comparison related**: up, down, under, above, below, next, through, against, during, within, including, range, higher, lower, drop, rise, growth, increase, less, compared, unchanged