The Fixed-Size Ordinally-Forgetting Encoding Method for Neural Network Language Models

Shiliang Zhang[1], **Hui Jiang**[2], **Mingbin Xu**[2], **Junfeng Hou**[1], **Lirong Dai**[1]
[1]National Engineering Laboratory for Speech and Language Information Processing
University of Science and Technology of China, Hefei, Anhui, China
[2]Department of Electrical Engineering and Computer Science
York University, 4700 Keele Street, Toronto, Ontario, M3J 1P3, Canada
{zsl2008,hjf176}@mail.ustc.edu.cn, {hj,xmb}@cse.yorku.ca, lrdai@ustc.edu.cn

Abstract

In this paper, we propose the new fixed-size ordinally-forgetting encoding (FOFE) method, which can almost uniquely encode any variable-length sequence of words into a fixed-size representation. FOFE can model the word order in a sequence using a simple ordinally-forgetting mechanism according to the positions of words. In this work, we have applied FOFE to feedforward neural network language models (FNN-LMs). Experimental results have shown that without using any recurrent feedbacks, FOFE based FNN-LMs can significantly outperform not only the standard fixed-input FNN-LMs but also the popular recurrent neural network (RNN) LMs.

1 Introduction

Language models play an important role in many applications like speech recognition, machine translation, information retrieval and nature language understanding. Traditionally, the back-off n-gram models (Katz, 1987; Kneser, 1995) are the standard approach to language modeling. Recently, neural networks have been successfully applied to language modeling, yielding the state-of-the-art performance in many tasks. In neural network language models (NNLM), the feedforward neural networks (FNN) and recurrent neural networks (RNN) (Elman, 1990) are two popular architectures. The basic idea of NNLMs is to use a projection layer to project discrete words into a continuous space and estimate word conditional probabilities in this space, which may be smoother to better generalize to unseen contexts. FNN language models (FNN-LM) (Bengio and Ducharme, 2001; Bengio, 2003) usually use a limited history within a fixed-size context window to predict the next word. RNN language models (RNN-LM) (Mikolov, 2010; Mikolov, 2012) adopt a time-delayed recursive architecture for the hidden layers to memorize the long-term dependency in language. Therefore, it is widely reported that RNN-LMs usually outperform FNN-LMs in language modeling. While RNNs are theoretically powerful, the learning of RNNs needs to use the so-called back-propagation through time (BPTT) (Werbos, 1990) due to the internal recurrent feedback cycles. The BPTT significantly increases the computational complexity of the learning algorithms and it may cause many problems in learning, such as gradient vanishing and exploding (Bengio, 1994). More recently, some new architectures have been proposed to solve these problems. For example, the long short term memory (LSTM) RNN (Hochreiter, 1997) is an enhanced architecture to implement the recurrent feedbacks using various learnable gates, and it has obtained promising results on handwriting recognition (Graves, 2009) and sequence modeling (Graves, 2013).

Comparing with RNN-LMs, FNN-LMs can be learned in a simpler and more efficient way. However, FNN-LMs can not model the long-term dependency in language due to the fixed-size input window. In this paper, we propose a novel encoding method for discrete sequences, named *fixed-size ordinally-forgetting encoding* (FOFE), which can almost uniquely encode any variable-length word sequence into a fixed-size code. Relying on a constant forgetting factor, FOFE can model the word order in a sequence based on a simple ordinally-forgetting mechanism, which uses the position of each word in the sequence. Both the theoretical analysis and the experimental simulation have shown that FOFE can provide *almost* unique codes for variable-length word sequences as long as the forgetting factor is properly selected. In this work, we apply FOFE to

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 495–500,
Beijing, China, July 26-31, 2015.

neural network language models, where the fixed-size FOFE codes are fed to FNNs as input to predict next word, enabling FNN-LMs to model long-term dependency in language. Experiments on two benchmark tasks, Penn Treebank Corpus (PTB) and Large Text Compression Benchmark (LTCB), have shown that FOFE-based FNN-LMs can not only significantly outperform the standard fixed-input FNN-LMs but also achieve better performance than the popular RNN-LMs with or without using LSTM. Moreover, our implementation also shows that FOFE based FNN-LMs can be learned very efficiently on GPUs without the complex BPTT procedure.

2 Our Approach: FOFE

Assume vocabulary size is K, NNLMs adopt the 1-of-K encoding vectors as input. In this case, each word in vocabulary is represented as a one-hot vector $\mathbf{e} \in \mathbb{R}^K$. The 1-of-K representation is a context independent encoding method. When the 1-of-K representation is used to model a word in a sequence, it can not model its history or context.

2.1 Fixed-size Ordinally Forgetting Encoding

We propose a simple context-dependent encoding method for any sequence consisting of discrete symbols, namely *fixed-size ordinally-forgetting encoding* (FOFE). Given a sequence of words (or any discrete symbols), $S = \{w_1, w_2, \cdots, w_T\}$, each word w_t is first represented by a 1-of-K representation $\mathbf{e}_t$, from the first word $t = 1$ to the end of the sequence $t = T$, FOFE encodes each partial sequence (history) based on a simple recursive formula (with $\mathbf{z}_0 = \mathbf{0}$) as:

$$\mathbf{z}_t = \alpha \cdot \mathbf{z}_{t-1} + \mathbf{e}_t \quad (1 \leq t \leq T) \qquad (1)$$

where $\mathbf{z}_t$ denotes the FOFE code for the partial sequence up to w_t, and α ($0 < \alpha < 1$) is a constant forgetting factor to control the influence of the history on the current position. Let's take a simple example here, assume we have three symbols in vocabulary, e.g., *A*, *B*, *C*, whose 1-of-K codes are $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ respectively. In this case, the FOFE code for the sequence $\{ABC\}$ is $[\alpha^2, \alpha, 1]$, and that of $\{ABCBC\}$ is $[\alpha^4, \alpha + \alpha^3, 1 + \alpha^2]$.

Obviously, FOFE can encode any variable-length discrete sequence into a fixed-size code. Moreover, it is a recursive context dependent encoding method that smartly models the order information by various powers of the forgetting factor. Furthermore, FOFE has an appealing property in modeling natural languages that the far-away context will be gradually forgotten due to $\alpha < 1$ and the nearby contexts play much larger role in the resultant FOFE codes.

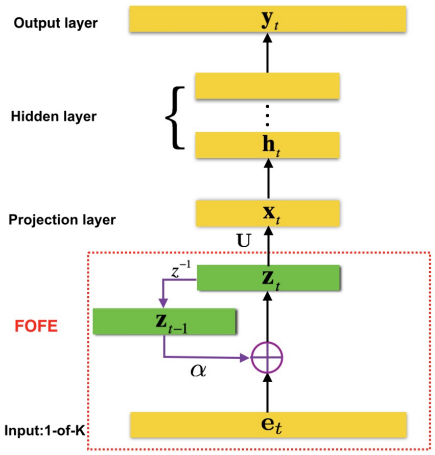

Figure 1: The FOFE-based FNN language model.

2.2 Uniqueness of FOFE codes

Given the vocabulary (of K symbols), for any sequence S with a length of T, based on the FOFE code $\mathbf{z}_T$ computed as above, if we can always decode the original sequence S unambiguously (perfectly recovering S from $\mathbf{z}_T$), we say FOFE is unique.

Theorem 1 *If the forgetting factor α satisfies $0 < \alpha \leq 0.5$,* FOFE *is unique for any K and T.*

The proof is simple because if the FOFE code has a value α^t in its i-th element, we may determine the word w_i occurs in the position t of S without ambiguity since no matter how many times w_i occurs in the far-away contexts ($< t$), they do not sum to α^t (due to $\alpha \leq 0.5$). If w_i appears in any closer context ($> t$), the i-th element must be larger than α^t.

Theorem 2 *For $0.5 < \alpha < 1$, given any finite values of K and T,* FOFE *is almost unique everywhere for $\alpha \in (0.5, 1.0)$, except only a finite set of countable choices of α.*

Refer to (Zhang et. al., 2015a) for the complete proof. Based on Theorem 2, FOFE is unique almost everywhere between $(0.5, 1.0)$ only except a countable set of isolated choices of α. In practice, the chance to exactly choose these isolated values between $(0.5, 1.0)$ is extremely slim, realistically

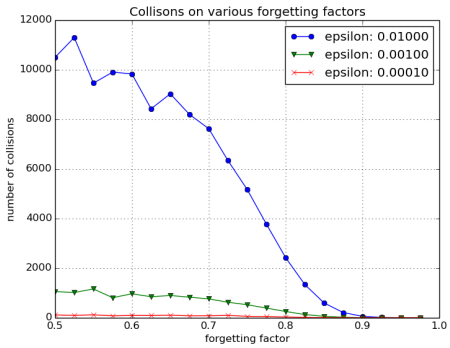

Figure 2: Numbers of collisions in simulation.

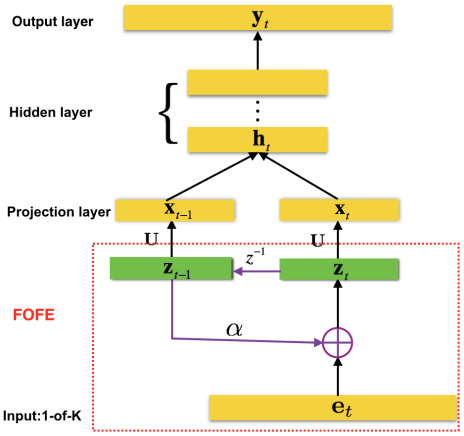

Figure 3: Diagram of 2nd-order FOFE FNN-LM.

almost impossible due to quantization errors in the system. To verify this, we have run simulation experiments for all possible sequences up to $T = 20$ symbols to count the number of collisions. Each collision is defined as the maximum element-wise difference between two FOFE codes (generated from two different sequences) is less than a small threshold ϵ. In Figure 2, we have shown the number of collisions (out of the total 2^{20} tested cases) for various α values when $\epsilon = 0.01$, 0.001 and 0.0001.[1] The simulation experiments have shown that the chance of collision is extremely small even when we allow a word to appear any times in the context. Obviously, in a natural language, a word normally does not appear repeatedly within a near context. Moreover, we have run the simulation to examine whether collisions actually occur in two real text corpora, namely PTB (1M words) and LTCB (160M words), using $\epsilon = 0.01$, we have not observed a single collision for nine different α values between $[0.55, 1.0]$ (incremental 0.05).

2.3 Implement FOFE for FNN-LMs

The architecture of a FOFE based neural network language model (FOFE-FNNLM) is shown in Figure 1. It is similar to regular bigram FNN-LMs except that it uses a FOFE code to feed into neural network LM at each time. Moreover, the FOFE can be easily scaled to higher orders like n-gram NNLMs. For example, Figure 3 is an illustration of a second order FOFE-based neural network language model.

FOFE is a simple recursive encoding method but a direct sequential implementation may not be efficient for the parallel computation platform like GPUs. Here, we will show that the FOFE computation can be efficiently implemented as sentence-by-sentence matrix multiplications, which are suitable for the mini-batch based stochastic gradient descent (SGD) method running on GPUs.

Given a sentence, $S = \{w_1, w_2, \cdots, w_T\}$, where each word is represented by a 1-of-K code as $\mathbf{e}_t$ $(1 \leq t \leq T)$. The FOFE codes for all partial sequences in S can be computed based on the following matrix multiplication:

$$\mathbf{S} = \begin{bmatrix} 1 & & & & \\ \alpha & 1 & & & \\ \alpha^2 & \alpha & 1 & & \\ \vdots & & \ddots & 1 & \\ \alpha^{T-1} & \cdots & & \alpha & 1 \end{bmatrix} \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \\ \vdots \\ \mathbf{e}_T \end{bmatrix} = \mathbf{MV}$$

where $\mathbf{V}$ is a matrix arranging all 1-of-K codes of the words in the sentence row by row, and $\mathbf{M}$ is a T-th order lower triangular matrix. Each row vector of $\mathbf{S}$ represents a FOFE code of the partial sequence up to each position in the sentence.

This matrix formulation can be easily extended to a mini-batch consisting of several sentences. Assume that a mini-batch is composed of N sequences, $\mathcal{L} = \{S_1\ S_2 \cdots S_N\}$, we can compute the FOFE codes for all sentences in the mini-batch as follows:

$$\bar{\mathbf{S}} = \begin{bmatrix} \mathbf{M}_1 & & & \\ & \mathbf{M}_2 & & \\ & & \ddots & \\ & & & \mathbf{M}_N \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_N \end{bmatrix} = \bar{\mathbf{M}}\bar{\mathbf{V}}.$$

[1] When we use a bigger value for α, the magnitudes of the resultant FOFE codes become much larger. As a result, the number of collisions (as measured by a fixed absolute threshold ϵ) becomes smaller.

When feeding the FOFE codes to FNN as shown in Figure 1, we can compute the activation signals (assume f is the activation function) in the first hidden layer for all histories in S as follows:

$$\mathbf{H} = f\Big((\bar{\mathbf{M}}\bar{\mathbf{V}})\mathbf{U}\mathbf{W}+\mathbf{b}\Big) = f\Big(\bar{\mathbf{M}}(\bar{\mathbf{V}}\mathbf{U})\mathbf{W}+\mathbf{b}\Big)$$

where $\mathbf{U}$ denotes the word embedding matrix that projects the word indices onto a continuous low-dimensional continuous space. As above, $\bar{\mathbf{V}}\mathbf{U}$ can be done efficiently by looking up the embedding matrix. Therefore, for the computational efficiency purpose, we may apply FOFE to the word embedding vectors instead of the original high-dimensional one-hot vectors. In the backward pass, we can calculate the gradients with the standard back-propagation (BP) algorithm rather than BPTT. As a result, FOFE based FNN-LMs are the same as the standard FNN-LMs in terms of computational complexity in training, which is much more efficient than RNN-LMs.

3 Experiments

We have evaluated the FOFE method for NNLMs on two benchmark tasks: i) the Penn Treebank (PTB) corpus of about 1M words, following the same setup as (Mikolov, 2011). The vocabulary size is limited to 10k. The preprocessing method and the way to split data into training/validation/test sets are the same as (Mikolov, 2011). ii) The Large Text Compression Benchmark (LTCB) (Mahoney, 2011). In LTCB, we use the *enwik9* dataset, which is composed of the first 10^9 bytes of enwiki-20060303-pages-articles.xml. We split it into three parts: training (153M), validation (8.9M) and test (8.9M) sets. We limit the vocabulary size to 80k for LTCB and replace all out-of-vocabulary words by <UNK>. [2]

3.1 Experimental results on PTB

We have first evaluated the performance of the traditional FNN-LMs, taking the previous several words as input, denoted as n-gram FNN-LMs here. We have trained neural networks with a linear projection layer (of 200 hidden nodes) and two hidden layers (of 400 nodes per layer). All hidden units in networks use the rectified linear activation function, i.e., $f(x) = \max(0, x)$. The nets are initialized based on the normalized initialization

[2]Matlab codes are available at `https://wiki.eecs.yorku.ca/lab/MLL/projects:fofe:start` for readers to reproduce all results reported in this paper.

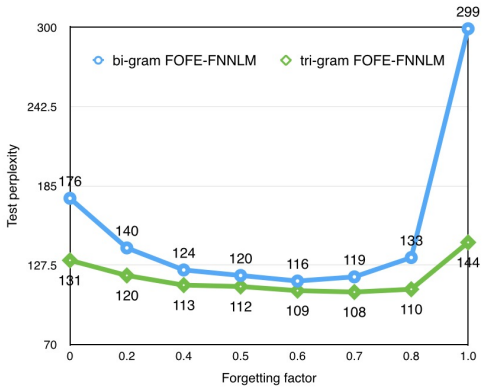

Figure 4: Perplexities of FOFE FNNLMs as a function of the forgetting factor.

in (Glorot, 2010), without using any pre-training. We use SGD with a mini-batch size of 200 and an initial learning rate of 0.4. The learning rate is kept fixed as long as the perplexity on the validation set decreases by at least 1. After that, we continue six more epochs of training, where the learning rate is halved after each epoch. The performance (in perplexity) of several n-gram FNN-LMs (from bi-gram to 6-gram) is shown in Table 1.

For the FOFE-FNNLMs, the net architecture and the parameter setting are the same as above. The mini-batch size is also 200 and each mini-batch is composed of several sentences up to 200 words (the last sentence may be truncated). All sentences in the corpus are randomly shuffled at the beginning of each epoch. In this experiment, we first investigate how the forgetting factor α may affect the performance of LMs. We have trained two FOFE-FNNLMs: i) 1st-order (using $\mathbf{z}_t$ as input to FNN for each time t; ii) 2nd-order (using both $\mathbf{z}_t$ and $\mathbf{z}_{t-1}$ as input for each time t, with a forgetting factor varying between $[0.0, 1.0]$. Experimental results in Figure 4 have shown that a good choice of α lies between $[0.5, 0.8]$. Using a too large or too small forgetting factor will hurt the performance. A too small forgetting factor may limit the memory of the encoding while a too large α may confuse LM with a far-away history. In the following experiments, we set $\alpha = 0.7$ for the rest experiments in this paper.

In Table 1, we have summarized the perplexities on the PTB test set for various models. The proposed FOFE-FNNLMs can significantly outperform the baseline FNN-LMs using the same architecture. For example, the perplexity of the baseline bigram FNNLM is 176, while the FOFE-

Table 1: Perplexities on PTB for various LMs.

Model	Test PPL
KN 5-gram (Mikolov, 2011)	141
FNNLM (Mikolov, 2012)	140
RNNLM (Mikolov, 2011)	123
LSTM (Graves, 2013)	117
bigram FNNLM	176
trigram FNNLM	131
4-gram FNNLM	118
5-gram FNNLM	114
6-gram FNNLM	113
1st-order FOFE-FNNLM	116
2nd-order FOFE-FNNLM	**108**

Table 2: Perplexities on LTCB for various language models. [M*N] denotes the sizes of the input context window and projection layer.

Model	Architecture	Test PPL
KN 3-gram	-	156
KN 5-gram	-	132
FNN-LM	[1*200]-400-400-80k	241
	[2*200]-400-400-80k	155
	[2*200]-600-600-80k	150
	[3*200]-400-400-80k	131
	[4*200]-400-400-80k	125
RNN-LM	[1*600]-80k	112
FOFE FNN-LM	[1*200]-400-400-80k	120
	[1*200]-600-600-80k	115
	[2*200]-400-400-80k	112
	[2*200]-600-600-80k	**107**

FNNLM can improve to 116. Moreover, the FOFE-FNNLMs can even overtake a well-trained RNNLM (400 hidden units) in (Mikolov, 2011) and an LSTM in (Graves, 2013). It indicates FOFE-FNNLMs can effectively model the long-term dependency in language without using any recurrent feedback. At last, the 2nd-order FOFE-FNNLM can provide further improvement, yielding the perplexity of 108 on PTB. It also outperforms all higher-order FNN-LMs (4-gram, 5-gram and 6-gram), which are bigger in model size. To our knowledge, this is one of the best reported results on PTB without model combination.

3.2 Experimental results on LTCB

We have further examined the FOFE based FNN-LMs on a much larger text corpus, i.e. LTCB, which contains articles from Wikipedia. We have trained several baseline systems: i) two n-gram LMs (3-gram and 5-gram) using the modified Kneser-Ney smoothing without count cutoffs; ii) several traditional FNN-LMs with different model sizes and input context windows (bigram, trigram, 4-gram and 5-gram); iii) an RNN-LM with one hidden layer of 600 nodes using the toolkit in (Mikolov, 2010), in which we have further used a spliced sentence bunch in (Chen et al. 2014) to speed up the training on GPUs. Moreover, we have examined four FOFE based FNN-LMs with various model sizes and input window sizes (two 1st-order FOFE models and two 2nd-order ones). For all NNLMs, we have used an output layer of the full vocabulary (80k words). In these experiments, we have used an initial learning rate of 0.01, and a bigger mini-batch of 500 for FNN-LMMs and of 256 sentences for the RNN and FOFE models. Experimental results in Table 2 have shown that the FOFE-based FNN-LMs can significantly outperform the baseline FNN-LMs (including some larger higher-order models) and also slightly overtake the popular RNN-based LM, yielding the best result (perplexity of 107) on the test set.

4 Conclusions

In this paper, we propose the fixed-size ordinally-forgetting encoding (FOFE) method to *almost* uniquely encode any variable-length sequence into a fixed-size code. In this work, FOFE has been successfully applied to neural network language modeling. Next, FOFE may be combined with neural networks (Zhang and Jiang, 2015; Zhang et. al., 2015b) for other NLP tasks, such as sentence modeling/matching, paraphrase detection, machine translation, question and answer and etc.

Acknowledgments

This work was supported in part by the Science and Technology Development of Anhui Province, China (Grants No. 2014z02006) and the Fundamental Research Funds for the Central Universities from China, as well as an NSERC Discovery grant from Canadian federal government. We appreciate Dr. Barlas Oguz at Microsoft for his insightful comments and constructive suggestions on Theorem 2.

References

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP)*, Volume 35, no 3, pages 400-401.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 181-184.

Paul Werbos. 1990. Back-propagation through time: what it does and how to do it. *Proceedings of the IEEE*, volume 78, no 10, pages 1550-1560.

Yoshua Bengio, Patrice Simard and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* volume 5, no 2, pages 157-166.

Yoshua Bengio and Rejean Ducharme. 2001. A neural probabilistic language model. In *Proc. of NIPS*, volume 13.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, volume 3, no 2, pages 1137-1155.

Jeffery Elman. 1990. Finding structure in time. *Cognitive science*, volume 14, no 2, pages 179-211.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of Interspeech*, pages 1045-1048.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528-5531.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Proc. of SLT*, pages 234-239.

X. Chen, Y. Wang, X. Liu, et al. 2014. Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch. In *Proc. of Interspeech*.

Ilya Sutskever and Geoffrey Hinton. 2010. Temporal-kernel recurrent neural networks. *Neural Networks*. pages 239-243.

Yong-Zhe Shi, Wei-Qiang Zhang, Meng Cai and Jia Liu. 2013. Temporal kernel neural network language model. In *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 8247-8251.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, volume 9, no 8, pages 1735-1780.

Alex Graves and Jurgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Proc. of NIPS*. pages 545-552.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv*:1308.0850.

Glorot Xavier and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of AISTATS*.

Matt Mahoney. 2011. Large Text Compression Benchmark. In *http://mattmahoney.net/dc/textdata.html*.

Barlas Oguz. 2015. Personal Communications.

Shiling Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou and LiRong Dai. 2015a. A Fixed-Size Encoding Method for Variable-Length Sequences with its Application to Neural Network Language Models. *arXiv:1505.01504*.

Shiliang Zhang and Hui Jiang. 2015. Hybrid Orthogonal Projection and Estimation (HOPE): A New Framework to Probe and Learn Neural Networks. *arXiv:1502.00702*.

Shiliang Zhang, Hui Jiang and Lirong Dai. 2015b. The New HOPE Way to Learn Neural Networks. *Proc. of Deep Learning Workshop at ICML 2015*.