# Chopout: A Simple Way to Train Variable Sized Neural Networks at Once

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Large deep neural networks require huge memory to run and their running speed is sometimes too slow for real applications. Therefore network size reduction with keeping accuracy is crucial for practical applications. We present a novel neural network operator, *chopout*, with which neural networks are trained, even in a single training process, so as to truncated sub-networks perform as well as possible. *Chopout* is easy to implement and integrate into most type of existing neural networks. Furthermore it enables to reduce size of networks and latent representations even after training just by truncating layers. We show its effectiveness through several experiments.

## 1 Introduction

Deep neural networks are crucial building blocks for current machine learning because of their outstanding performance in accuracy and ease of use. However, such deep neural networks sometimes run too slow and consume too much memories. Therefore, neural networks with less parameters are preferable for applications.

For this end, various parameter size reduction techniques are developed. They includes (1) pruning techniques which aim to prune weights, channels or layers of neural networks (Han et al. [2015a], Aghasi et al. [2017], Dong et al. [2017], Molchanov et al. [2016], Li et al. [2017], Luo et al. [2017], Ye et al. [2018], Liu et al. [2017], He et al. [2017]), (2) quantization techniques which aim to quantize weights of neural networks into $\{+1, -1\}$ or lower precision floating points (e.g. fp16) (Han et al. [2015b], Courbariaux et al. [2015], Rastegari et al. [2016], Zhou et al. [2016], Zhu et al. [2016], Wu et al. [2016], Hubara et al. [2017]), (3) decomposition techniques which aim to decompose weights with combinations of smaller components (e.g. SVD) (Denton et al. [2014], Jaderberg et al. [2014], Lebedev et al. [2014], Yang et al. [2015], Novikov et al. [2015]), (4) distillation techniques which aim to train smaller neural networks (student networks) to mimic trained larger neural networks (teacher networks) (Hinton et al. [2015], Mishra and Marr [2017], Polino et al. [2018]) and (5) techniques which aim to design more compact but accurate neural networks (Iandola et al. [2016], Howard et al. [2017], Sandler et al. [2018], Zhang et al. [2017]).

These techniques sometimes can achieve remarkable parameter size reduction without too much accuracy decrease. However some of them have limitations of network architectures, are hard to implement using modern deep learning frameworks such as TensorFlow (Abadi et al. [2016]), Pytorch (Paszke et al. [2017]) or MxNet (Chen et al. [2015]) or require model specific adaptations.

In this research, we proposed a novel simple stochastic operator *chopout* which is similar to *dropout* (Srivastava et al. [2014]) but randomly truncate latter dimensions/channels of layers in neural networks, with which deep neural networks are trained so that their truncated sub-networks also perform well.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

(a) input (b) dropout (c) chopout

Figure 1: Instead of zeroing out cells in *dropout*, *chopout* truncates random-length latter cells.

Table 1: Autoencoders used in experiments. L(n) denotes the linear layer with $n$-dimensional output, R denotes the rectified linear unit function and S denotes the sigmoid function. *Chopout* and *dropout* is applied right after applying the encoder.

| dataset | encoder | decoder |
|---|---|---|
| MNIST | L(100)-R-L(100)-R-L(100) | L(100)-R-L(100)-R-L(784)-S |

# 2 Method

Firstly we define *chopout* for 1-dimensional case. At training time, *chopout* is defined as a truncation of random-length latter dimensions of vectors as follows (Figure 1):

$$m \sim P_m = P(\{0, 1, \cdots, d\}), \ \ \mathbf{x} \leftarrow proj_m(\mathbf{x}) := (x_1, x_2, ..., x_m, 0, ..., 0)$$

where $P_m = P(\{0, 1, \cdots, d\})$ is an arbitrary discrete distribution over $\{0, 1, \cdots, d\}$ (e.g. uniform distribution), $\mathbf{x} \in \mathbb{R}^d$ is a vector and $proj_m(\cdot)$ is a projection onto first $m$-th dimensions. *Chopout*'s behavior is similar to *dropout* but, instead of zeroing out random elements in *dropout*, *chopout* zeros out (or truncates) random latter consecutive elements. In back-propagation, the same latter dimensions of gradients are also zeroed out as well

$$\mathbf{grad} \leftarrow proj_m(\mathbf{grad}) := (grad_1, grad_2, ..., grad_m, 0, ..., 0)$$

where $\mathbf{grad}$ is a gradient and $m$ is the number drawn in the forward pass.

At test time, *chopout* is defined to behave as a identity function, that is, just pass through the input vector without any modification. This definition of *chopout* in prediction mode is contrastive to that of *dropout*, which, in prediction time, *dropout* scale inputs to make it consistent with training time.

Training a fully-connected neural network with applying *chopout* can be interpreted as simultaneous training of randomly sampled sub-networks which are obtained by cuttinng out *former parts* of the original fully-connected neural network with sharing parameters.

In higher dimensional cases, *chopout* can be easily extended as a random truncation of channels instead of dimensions. For example, when applied to a tensor $\mathbf{x} \in \mathbb{R}^{c \times h \times w}$, the forward-propagation of *chopout* is defined as

$$m \sim P_m = P(\{0, 1, \cdots, c\}), \ \ x_{kij} \leftarrow \begin{cases} x_{kij} & (k \leq m) \\ 0 & (otherwise) \end{cases}$$

where $P(\{0, 1, \cdots, c\})$ is an arbitrary distribution. Back-propagation is defined in the same way.

# 3 Experiments

Throughout experiments, we use uniform distributions over $\{1, \cdots, d\}$ for $P_m(\{0, 1, \cdots, d\})$.

## 3.1 Autoencoder

We train autoencoders on MNIST (LeCun et al. [1998], Table 1, Figure 2). We see that by applying *chopout* on the hidden layer of the autoencoder, the reconstruction is kept well even after the hidden layer is truncated.

## 3.2 Skip-gram

We apply *chopout* for embeddings trained through skip-gram models (Mikolov et al. [2013a,b]). We use text8 corpus[1]. We set the window size to 5 and ignore infrequent words which appear less than 20 times in the corpus. The result (Table 2) shows the consistency of embeddings.

[1] http://mattmahoney.net/dc/text8.zip

(a) normal (b) with dropout(p=0.5) (c) with chopout

Figure 2: (a) reconstruction of test images by a autoencoder wihout *chopout* and *dropout*, (b) with *dropout* and (c) with *chopout*. In each figure, the left most column represents input images and, from next column to the most right one, each column correspond to the reconstruction results with truncating latter 0, 10, 20, 30, ..., 90 dimensions of hidden layers.

Table 2: top-5 most similar words for learnt 512-dim embeddings. The results of 64-dim embeddings obtained by truncation are also shown.

| ran (512) | ran (64) | news (512) | news (64) | good (512) | good (64) |
|---|---|---|---|---|---|
| stopped | rides | unofficial | openoffice | clever | balanced |
| stood | stayed | headline | unofficial | strong | queueing |
| graduated | shot | homepage | overviews | suitable | transparent |
| struck | sank | portal | bbc | unusually | recursive |
| shot | fired | online | portal | very | shorthand |

### 3.3 Image classification

LeNet-300-100 (LeCun et al. [1998]) are trained on MNIST with/without *chopout* and *dropout* in each intermediate layer (Figure 3). This is a very initial experiment but the result shows training with *chopout* enhance the robustness of networks against pruning.

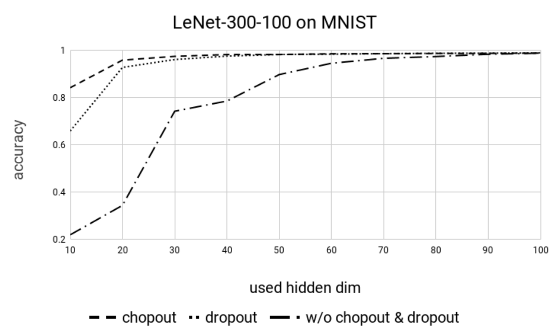


Figure 3: chopout/dropout is applied in each intermediate layer.

## 4 Discussion

We introduced a novel stochastic operator *chopout* and showed it gathers important information in former parts of lafyers. Because the concept of *chopout* is very simple and flexible, there could be broad direction of further research.

(1) The distribution $P_m(\{0, 1, \cdots, d\})$ should be explored. If we put *chopout*s in every layer of a neural network, then, in training, there could be a layer where drawn $m \sim P_m(\{0, 1, \cdots, d\})$ is very small and it could be a bottleneck of the prediction accuracy.

(2) *Chopout* can be used for network pruning. The information-gathering property of *chopout* enables to prune latter dimensions/channels of layers with keeping accuracy but the extent of pruning is still should be explored. For this end, reinforcement learning techniques (e.g. bandit algorithms) can be used to detect the appropriate pruning ratio. Combination with weight pruning methods are also interesting as well.

## 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. *arXiv preprint arXiv:1605.08695*, 2016.

Alireza Aghasi, Afshin Abdi, Nam Nguyen, and Justin Romberg. Net-trim: Convex pruning of deep neural networks with performance guarantee. In *Advances in Neural Information Processing Systems*, pages 3177–3186, 2017.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, pages 4857–4867, 2017.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR),*, 2015a.

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

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, 2017.

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.

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

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Graf. Pruning filters for efficient ConvNets. In *International Conference on Learning Representation (ICLR)*, 2017.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Asit Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *International Conference on Learning Representation (ICLR)*, 2017.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representation (ICLR)*, 2016.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pages 442–450, 2015.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *International Conference on Learning Representation (ICLR)*, 2018.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

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

Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *International Conference on Learning Representation (ICLR)*, 2018.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *International Conference on Learning Representation (ICLR)*, 2016.