Survey of Dropout Methods for Deep Neural Networks

Alex Labach
University of Toronto
alex.labach@mail.utoronto.ca

Hojjat Salehinejad
University of Toronto
hojjat.salehinejad@mail.utoronto.ca

Shahrokh Valaee
University of Toronto
valaee@ece.utoronto.ca

Abstract

Dropout methods are a family of stochastic techniques used in neural network training or inference that have generated significant research interest and are widely used in practice. They have been successfully applied in neural network regularization, model compression, and in measuring the uncertainty of neural network outputs. While original formulated for dense neural network layers, recent advances have made dropout methods also applicable to convolutional and recurrent neural network layers. This paper summarizes the history of dropout methods, their various applications, and current areas of research interest. Important proposed methods are described in additional detail.

1 Introduction

Deep neural networks are a topic of massive interest in contemporary artificial intelligence and signal processing. Their high number of parameters make them particularly prone to overfitting, requiring regularization methods in practice. Dropout was introduced in 2012 as a technique to avoid overfitting [1] and was subsequently applied in the 2012 winning submission for the Large Scale Visual Recognition Challenge that revolutionized deep learning research [2]. The original method omitted each neuron in a neural network with probability 0.5 during each training iteration, with all neurons being included during testing. This technique was shown to significantly improve results on a variety of tasks [1].

In the years since, a wide range of stochastic techniques inspired by the original dropout method have been proposed. We use the term *dropout methods* to refer to them in general. They include dropconnect [3], standout [4], fast dropout [5], variational dropout [6], Monte Carlo dropout [7] and many others. Figure 1 illustrates research into dropout methods over time. Generally speaking, dropout methods involve randomly modifying parameters during neural network training or inference, or approximating this process. While originally used to avoid overfitting, dropout methods have since expanded to a variety of applications.

One such application is the use of dropout methods to compress deep neural networks. In [8], dropout was found to promote a sparse distribution of neural network weights. Later work has taken advantage of this property to design dropout methods that can be combined with neural pruning to compress neural networks with minimal loss in accuracy [9–11].

Another direction of research into dropout methods has been applying them to a wider range of neural network topologies. This includes methods for applying dropout to convolutional neural network layers [12] as well as to recurrent neural networks (RNNs) [13, 14]. RNN dropout methods

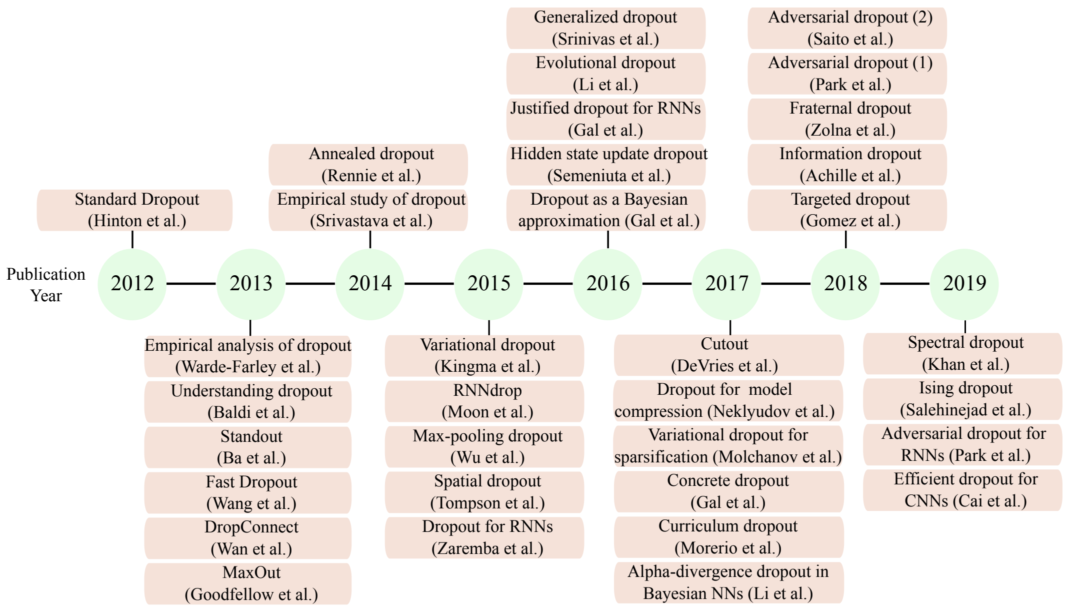
Publication Year
2012
2013
2014
2015
2016
2017
2018
2019
Standard Dropout (Hinton et al.)
Annealed dropout (Rennie et al.)
Empirical study of dropout (Srivastava et al.)
Generalized dropout (Srinivas et al.)
Evolutional dropout (Li et al.)
Justified dropout for RNNs (Gal et al.)
Hidden state update dropout (Semeniuta et al.)
Dropout as a Bayesian approximation (Gal et al.)
Adversarial dropout (2) (Saito et al.)
Adversarial dropout (1) (Park et al.)
Fraternal dropout (Zolna et al.)
Information dropout (Achille et al.)
Targeted dropout (Gomez et al.)
Empirical analysis of dropout (Warde-Farley et al.)
Understanding dropout (Baldi et al.)
Standout (Ba et al.)
Fast Dropout (Wang et al.)
DropConnect (Wan et al.)
MaxOut (Goodfellow et al.)
Variational dropout (Kingma et al.)
RNNdrop (Moon et al.)
Max-pooling dropout (Wu et al.)
Spatial dropout (Tompson et al.)
Dropout for RNNs (Zaremba et al.)
Cutout (DeVries et al.)
Dropout for model compression (Neklyudov et al.)
Variational dropout for sparsification (Molchanov et al.)
Concrete dropout (Gal et al.)
Curriculum dropout (Morerio et al.)
Alpha-divergence dropout in Bayesian NNs (Li et al.)
Spectral dropout (Khan et al.)
Ising dropout (Salehinejad et al.)
Adversarial dropout for RNNs (Park et al.)
Efficient dropout for CNNs (Cai et al.)

Figure 1: Some proposed methods and theoretical advances in dropout methods from 2012 to 2019.

in particular have become commonly used, and have been recently applied in achieving state-of-the-art results in natural language processing [15–17].

This paper is structured as follows. Section 2 describes the original dropout method proposed by Hinton et al. [1] and introduces basic concepts common to dropout methods. Section 3 describes dropout methods for general neural network training. Section 4 describes dropout methods specialized for training convolutional neural network layers and Section 5 describes methods specialized for recurrent neural network layers. Section 6 describes dropout methods for compressing neural networks. Section 7 describes Monte Carlo dropout. Finally, Section 8 discusses current and future research directions.

1.1 Notation

We follow a number of common conventions when providing formulas describing neural network cells or layers. Bold lower-case letters represent vectors and bold upper-case letters represent matrices. Most of the time, when multiplying an input vector by a weight matrix, a vector of learned biases is also added, for example producing $\mathbf{W}\mathbf{x} + \mathbf{b}$ from an input vector $\mathbf{x}$. To simplify notation, we generally treat the biases as elements of the weight matrix, with an element with value 1 implicitly appended to the vector $\mathbf{x}$. So, the previous operation would be written as $\mathbf{W}\mathbf{x}$. The operator $\circ$ represents element-wise (or Hadamard) multiplication.

2 Standard dropout

The original proposed dropout method, introduced in 2012, provides a simple technique for avoiding overfitting in feedforward neural networks [1]. During each training iteration, each neuron is omitted from the network with probability p. Once trained, the full network is used, although neuron outputs are multiplied by the probability p that the neuron was omitted. This compensates for the larger size of the network now that no neurons are dropped, and can be interpreted as averaging over the possible networks during training. The probability can vary for each layer, with the original paper recommending $p = 0.2$ for the input layer and $p = 0.5$ for hidden layers. Neurons in the output layer are not dropped. This technique is usually simply known as dropout, but for the purposes of this article we will call it *standard dropout*, to distinguish it from other dropout methods. This method is illustrated in Figure 2.

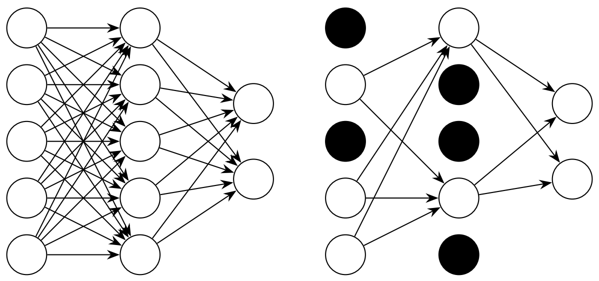
Figure 2: An example of standard dropout. The left network is fully connected, and the right has had neurons dropped with probability 0.5. Dropout is not applied to the output layer.

Mathematically, the behaviour of standard dropout during training for a neural network layer is given by:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x}) \circ \mathbf{m}, \quad m_i \sim \text{Bernoulli}(p) \tag{1}$$

where $f(\cdot)$ is the activation function, $\mathbf{x}$ is the layer input, $\mathbf{W}$ is the layer weight matrix, $\mathbf{y}$ is the layer output, and $\mathbf{m}$ is the layer dropout mask, with each element being 1 with probability p. During testing, the layer output is given by

$$\mathbf{y} = pf(\mathbf{W}\mathbf{x}). \tag{2}$$

Standard dropout is equivalent to adding an additional layer after a layer of neurons that simply sets values to zero with some probability during training, and multiplies them by p during testing. Other formulations of standard dropout may scale weights rather than outputs during testing, or scale outputs during testing rather than training, but both of these approaches have the same effect as the formulation given here.

This method proved effective for regularizing neural networks, enabling them to be trained for longer periods without overfitting and resulting in improved test accuracy [1, 8]. Standard dropout has since become widely used in practice.

A number of theoretical approaches have been proposed to explain the efficacy of standard dropout. The original paper interprets it as an approximate average over all possible networks that can be generated by realizations of dropout masks [1]. It also notes that dropout prevents co-adaptation of features, in which neurons rely excessively on other neurons to generate meaningful results [1]. This interpretation is expanded upon in [8] and is evaluated empirically in [18]. Averaging properties of dropout are studied in detail in [19].

A recent alternative theoretical interpretation analyzes dropout as a Bayesian approximation of a deep Gaussian process [7]. This approach has inspired a number of new dropout methods, including variational LSTM dropout, as described in Section 5, and Monte Carlo dropout, as described in Section 7.

3 Dropout methods for training

This section describes important dropout methods that, like standard dropout, are generally used to regularize dense feedforward neural network layers during training. Most of these methods were directly inspired by standard dropout, and seek to improve on its speed or regularization effectiveness. Dropout methods for other kinds of neural network layers or for applications other than regularization are described in later sections.

One of the first proposed variations on standard dropout was dropconnect, introduced in 2013 by Wan et al. [3]. This method is a generalization of dropout where individual weights and biases are set to zero with some probability rather than neuron outputs. So, in training, the output of a network layer is given by:

$$\mathbf{y} = f((\mathbf{W} \circ \mathbf{M})\mathbf{x}), \quad m_{ij} \sim \text{Bernoulli}(p), \tag{3}$$

where terms are defined as in 1, but with a dropout mask matrix rather than a vector. Dropconnect is illustrated in Figure 3.

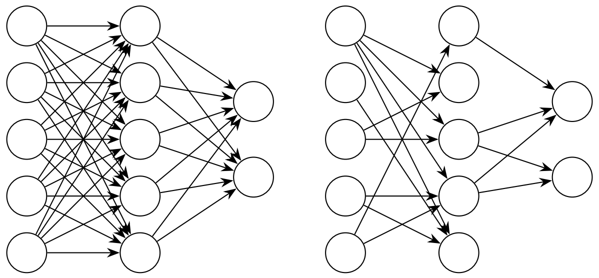

Figure 3: An example of dropconnect. The right network has had weights dropped with probability 0.5.

Dropconnect takes a different approach than standard dropout during test time. Rather than setting weights to their average value, the authors propose a Gaussian approximation of dropconnect at each neuron [3]. A sample is then taken from this Gaussian and passed to the neuron activation function. This makes dropconnect a stochastic method at test time as well as during training.

Dropout methods that drop weights rather than neurons are sometimes are also sometimes called weight dropout.

Standout [4] is a dropout method that seeks to improve on standard dropout by choosing neurons to omit adaptively rather than randomly. This is done by overlaying a binary belief network onto a neural network which controls the neural network's architecture. For each weight in the original neural network, Standout adds a corresponding weight parameter in the binary belief network. A layer's output during training is given by:

$$y = f(\mathbf{W}\mathbf{x}) \circ \mathbf{m}, \quad m_i \sim \text{Bernoulli}(g(\mathbf{W}_s\mathbf{x})) \tag{4}$$

where terms are defined as in (1), but with $\mathbf{W}_s$ representing the belief network's weights for that layer and $g(\cdot)$ representing the belief network's activation function.

While a separate learning algorithm can be applied to learn the belief network weights, in practice, the authors found that this resulted in the belief network weights becoming approximately equal to an affine function of the corresponding neural network weights [4]. So, an effective approach to determine belief network weights is setting them as

$$\mathbf{W}_s = \alpha\mathbf{W} + \beta \tag{5}$$

at each training iteration for some constants α and β. The output of each layer during testing is given by:

$$y = f(\mathbf{W}\mathbf{x}) \circ g(\mathbf{W}_s\mathbf{x}). \tag{6}$$

Fast dropout [5] provides a faster way to do dropout-like regularization, by interpreting dropout methods from a Bayesian perspective. The authors show that the outputs of layers with dropout can be seen as sampling from an underlying distribution, which can be approximated by a Gaussian. This distribution can then either be sampled from directly or its parameters can be used to propagate information about the entire dropout ensemble. This technique can lead to faster training procedures than standard dropout, where only one element of the ensemble of possible networks is sampled at once. One reason for this is that in standard dropout, for a given training sample, the fraction of neurons trained on the sample is p. To effectively use an entire training dataset to train all neurons requires passing each sample through the network multiple times. Fast dropout exposes all neurons to each training sample, which avoids this slowdown. Fast dropout can also be directly applied at test time, as opposed to the approximate averaging employed in standard dropout.

Another method inspired by a Bayesian understanding of dropout is variational dropout, as proposed by Kingma et al. [6] (not to be confused with work by Gal and Ghahramani [13]). The authors show that a variant of dropout that uses Gaussian multiplicative noise (proposed by Srivastava et al. [8]) can be interpreted as a variational method given a particular prior over the network weights and a particular variational objective. They then derive an adaptive dropout scheme that can automatically determine an effective dropout probability for an entire network, or for individual layers or neurons. This is a potential improvement over established ways of determining dropout rates, such as using

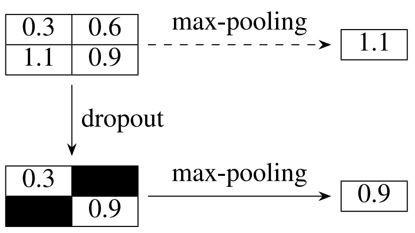

Figure 4: Max-pooling dropout in convolutional neural networks [12].

a fixed probability or grid search. Concrete dropout [20] is an alternative proposed method for automatically tuning dropout probabilities.

Variational dropout has also been applied to sparsify networks as a step in model compression. This application is described in section 6.

4 Convolutional layers

Naive dropout for convolutional neural networks (CNNs), defined as dropping pixels randomly in feature maps or in the input image, does not significantly reduce overfitting, mainly because adjacent pixels to the dropped pixels are highly correlated [21]. However, recently, many promising advances in using dropout as a regularization method for training CNNs have been proposed.

Pooling is a common operation in CNN topologies. The two main pooling methods are max-pooling and average-pooling. The max-pooling operator selects the maximum activation value from non-overlapping areas of an input feature map. The average-pooling operator computes the average of activation values in each area as the input value for the next layer. The average-pooling operator considers every feature value in the pooling window equally, while max-pooling always takes the strongest activation, meaning that other feature map values have no effect on its output.

Max-pooling dropout [12] is a method that retains the behaviour of max-pooling layers while probabilistically allowing other feature values to affect the output of a pooling layer. This operator masks a subset of feature values before performing the max-pooling operation. As Figure 4 shows, the max-pooling operator always pools the largest value in a given pooling window, while the max-pooling dropout method provides an opportunity for smaller feature values to affect activations in later layers. This technique can help the network to avoid overfitting as saturated activation values have less contribution in the network loss. At test time, the pooling operation becomes a linear sum over activations, where each activation is weighted by the probability that it would be selected as the output during training according to this dropout method.

Ideally, a pooling method will uphold task-related information while discarding irrelevant image details [12]. Naive CNN dropout does not have significant performance to prevent CNNs from overfitting, mainly because adjacent pixels to the dropped pixels are highly correlated [21]. Spatial dropout is a method that drops an entire feature map across a channel, which improves network robustness compared to not using dropout for join position prediction [21].

Maxout networks improve dropout by enhancing its model averaging capability. In a CNN, the maxout feature map is constructed by taking the maximum value across a number of affine feature maps and the dropout mask is elementwise multiplied prior to the multiplication by the weights. In this case, the inputs to the max operator are not dropped [22].

In [23], a dropout method is proposed where the dropout probability changes based on the training iteration. The probability of dropping neurons is drawn from a uniform or normal distribution. This approach is equivalent to adding noise to the output feature maps of each layer. This technique increases the robustness of the network to variations of images with noise [23]. The authors also propose max-drop, in which high activation values are selectively dropped. These values are selected across the feature map or the channels [23]. The reported results in [23] show performance of the proposed methods are comparable with spatial dropout [21].

Cutout is another dropout-based regularization and data augmentation technique for training CNNs [24], which applies a random square mask over a region of each input image. Unlike other common methods which apply dropout at the feature map level, this method directly applies to the input image. The main motivation behind cutout is removing visual features with high activation values in later layers of a CNN [24]. However, surprisingly, this masking approach on input images has equivalent performance and is cheaper to conduct.

4.1 Dropout and batch normalization

There is extensive debate around the efficacy of using dropout for training CNNs. It is widely believed that batch normalization is enough to discourage co-adaptation in CNNs and to prevent them from overfitting. In [25], eliminating dropout is shown to have negligible impact on the training of a CNN. This is followed by the introduction of batch normalization as a regularizer to reduce internal co-variant shift while discouraging the strength of dropout to prevent overfitting [26]. Batch normalization can also prevent the network from getting stuck in saturated modes [26].

We believe that there is significant potential for dropout methods in CNN training. While standard dropout has limited efficacy, the methods described above all improve upon it. Recent research has also looked at the mechanics of using dropout and batch normalization together. Cai et al. [27] examines using dropout alongside batch normalization in CNNs at neuron, channel, path, and layer levels. In this approach, dropout and batch normalization are reordered in convolutional building blocks to address the increase of variance from random deactivation of basic components such as neurons [27]. The authors claim that the failure of standard dropout, resulting in training instability, is due to the incorrect placement of dropout and batch normalization operations in the the convolutional layer. The conducted experiments on various datasets show that reordering them improves performance [27].

5 Recurrent layers

In general, feedforward dropout methods as described in section 3 can be applied to the feedforward connections of a network containing recurrent layers. Research has therefore focused on applying dropout methods to recurrent connections. Applying standard dropout to these connections results in poor performance [28], since the noise caused by dropout at each time step prevents the network from retaining long-term memory. However, methods that are specialized for recurrent layers have proved successful, and are commonly used in practice. Generally speaking, they apply dropout to recurrent connections in a way that can still preserve long-term memory.

Research into dropout in recurrent neural networks (RNNs) has focused on long short-term memory (LSTM) networks, although some proposed methods can be applied to RNNs in general. In an LSTM cell, for a given input $\mathbf{x}_t$, the input, forget, and output gates at time t are defined as

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}\right), \tag{7}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}\right), \tag{8}$$

and

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}\right), \tag{9}$$

respectively. The cell gate is defined as

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t, \tag{10}$$

where

$$\mathbf{g}_t = \tanh\left(\mathbf{W}_g \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}\right). \tag{11}$$

The hidden state is

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t), \tag{12}$$

where $\mathbf{W}$ represents learned weights, $\sigma(\cdot)$ represents a sigmoid activation function, and $\sigma(\cdot)$ and $\tanh(\cdot)$ are applied element-wise. For more information on LSTM networks, see [29].

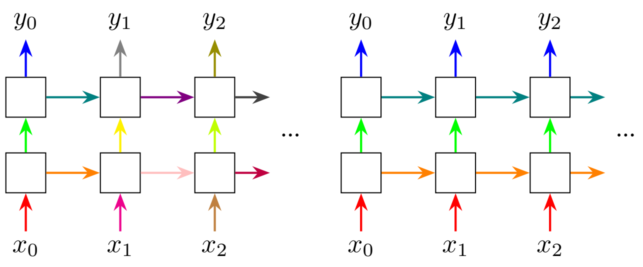

Figure 5: Comparison of per-step (left) versus per-sequence (right) sampling of dropout masks on an unrolled RNN. Horizontal connections are recurrent while vertical connections are feedforward. Different colours represent different dropout masks applied to the corresponding connection.

RNNdrop [30], proposed in 2015, provides a simple solution to better preserve memory when applying dropout. The key change is to generate a dropout mask for each input sequence, and keep it the same at every time step. This varies from the naive way of applying dropout to RNNs, which would generate new dropout masks for each input sample, regardless of which time sequence it was from. Generating masks on a per-sequence basis means that the elements in the network hidden state that are not dropped will persist throughout the entire sequence without ever being affected by dropout, which allows the network to maintain long-term memory. This is illustrated in Figure 5.

In particular, the authors proposed applying dropout to the hidden cell state. So, the only change from the original LSTM definition is the equation for $\mathbf{c}_t$, which becomes

$$\mathbf{c}_t = \mathbf{m} \circ (\mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t), \quad m_i \sim \text{Bernoulli}(p).$$

In 2016, an RNN dropout variant based on a Bayesian interpretation of dropout methods was proposed by Gal and Ghahramani [13]. The authors showed that if dropout is seen as a variational Monte Carlo approximation to a Bayesian posterior, then the natural way to apply it to recurrent layers is to generate a dropout mask that zeroes out both feedforward and recurrent connections for each training sequence, but to keep the same mask for each time step in the sequence. This is similar to RNNdrop in that masks are generated on a per-sequence basis, but the derivation leads to dropout being applied at a different point in the LSTM cell. Formally, the equations for $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, and $\mathbf{g}_t$ become:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i\left(\begin{bmatrix}\mathbf{x}_t\\ \mathbf{h}_{t-1}\end{bmatrix} \circ \mathbf{m}\right)\right) \tag{13}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f\left(\begin{bmatrix}\mathbf{x}_t\\ \mathbf{h}_{t-1}\end{bmatrix} \circ \mathbf{m}\right)\right) \tag{14}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o\left(\begin{bmatrix}\mathbf{x}_t\\ \mathbf{h}_{t-1}\end{bmatrix} \circ \mathbf{m}\right)\right) \tag{15}$$

$$\mathbf{g}_t = \tanh\left(\mathbf{W}_g\left(\begin{bmatrix}\mathbf{x}_t\\ \mathbf{h}_{t-1}\end{bmatrix} \circ \mathbf{m}\right)\right) \tag{16}$$

$$m_i \sim \text{Bernoulli}(p) \tag{17}$$

with the equations for $\mathbf{c}_t$ and $\mathbf{h}$ remaining the same as in the original LSTM. This dropout method has become one of the most widespread techniques for regularizing RNNs.

Recurrent dropout [14] is an alternative approach that can preserve memory in an LSTM while still generating different dropout masks for each input sample, as in standard dropout. This is done by only applying dropout to the part of the RNN that updates the hidden state and not the state itself. So, if an element is dropped, then it simply does not contribute to network memory, rather than erasing the hidden state. For an LSTM, the equations are the same as in the original LSTM except that the equation for $\mathbf{c}_t$ becomes

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t \circ \mathbf{m}_t, \quad m_{t,i} \sim \text{Bernoulli}(p). \tag{18}$$

6 Dropout methods for model compression

Standard dropout promotes sparsity in neural network weights [8]. This property means that dropout methods can be applied in compressing neural network models by reducing the number of parameters needed to perform effectively. Since 2017, several dropout-based approaches have been proposed for practical model compression.

In 2017, Molchanov et al. [9] proposed using variational dropout [6] (described in Section 3) to sparsify both fully connected and convolutional layers. This approach was shown to achieve large reductions in the number of parameters in standard convolutional networks while minimally affecting performance. This sparse representation can then be passed into existing methods that convert sparse networks into compressed models, as in [31]. A similar method was proposed by Neklyudov et al. [10], which uses a modified variational dropout scheme that promotes sparsity, but the resulting network is specifically structured in such a way that is easy to compress.

Developing further dropout methods for model compression has been an area of significant activity recently. Recently proposed approaches include targeted dropout [32], in which neurons are chosen adaptively to be dropped out in such a way that the network adapts to neural pruning, allowing it to be shrunk considerably without much loss in accuracy. Another recent proposal is Ising-dropout [11], which overlays a graphical Ising model on top of a neural network in order to identify less useful neurons, and drop them out in both training and inference. We expect to continue to see advances in applying dropout methods for model compression.

7 Monte Carlo dropout

In 2016, Gal and Ghahramani [7] proposed a Bayesian theoretical understanding of dropout that has since become widely accepted. They interpret dropout as a Bayesian approximation of a deep Gaussian process. A consequence of this interpretation is that applying standard dropout at test time, rather than scaling weights and using all neurons as described in Section 2, allows for the model uncertainty to be measured.

Roughly speaking, more a network output tends to vary when using dropout, the higher the model's output uncertainty is. To be precise, this means taking a input sample and running a neural network T times with standard dropout, using the same input but different randomly generated dropout masks each time. The sample means and variances of the output represent estimates of the the predictive mean and variance of the model. A proof is given in [7].

This approach provides a simple method for estimating the confidence of a neural network output, in addition to the usual point estimate output. Monte Carlo dropout has become widely applied for estimating model uncertainty.

8 Discussion

We have described a wide range of advances in dropout methods above. This section discusses ongoing research trends in broader terms.

The most common research direction in dropout methods has been improving dropout for regularization. It is generally accepted that standard dropout can regularize a wide range of neural network models, but there is room to achieve either faster training convergence or better final performance. The former concern is important since dropout reduces the exposure of neurons to each training sample, which can slow down training [5]. With neural networks becoming larger and more computationally intensive to train, techniques such as fast dropout [5] that reduce this effect are valuable. Improving how dropout affects the performance of trained networks is also an ongoing concern. Trying to drop neurons in a more intelligent or theoretically justified way than standard dropout has shown promise. Also, the growth of convolutional and recurrent neural networks in practice has prompted the development of specialized methods that perform better than standard dropout on specific kinds of neural networks. As new kinds of neural networks and neural network layers continue to be developed, there continue to be opportunities to design or improve on specialized dropout methods.

Other research into dropout methods looks to widen their applications beyond regularization. As discussed above, this includes the use of dropout for model compression, either on its own or in concert with existing model compression techniques. As with regularization, there are opportunities to develop improved methods that are specialized for particular kinds of networks or that use more advanced approaches for selecting neurons to drop. Monte Carlo dropout is another application of dropout methods: using them to measure model uncertainty. There is potential for further applications, given the broad ability of dropout methods to stochastically guide network training and operation.

A promising line of research has emerged into adversarial dropout methods [33–35]. These techniques either incorporate dropout methods into adversarial learning schemes, or apply ideas from adversarial learning to guide dropout procedures in a more effective way.

Finally, a substantial amount of theoretical analysis has been done to rigorously justify existing dropout methods. The growth of Bayesian interpretations of dropout methods over the last few years points to new opportunities in theoretical justifications of dropout and similar stochastic methods, which corresponds to a broader trend of Bayesian and variational techniques advancing research into deep neural networks.

In general, dropout methods have continually shown their utility and potential throughout deep learning, and we expect this trend to continue as deep neural networks continue to become more advanced and widely used.

References

[1] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[3] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural network using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning*. PLMR, 2013.

[4] L. J. Ba and B. Frey, "Adaptive dropout for training deep neural networks," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*. NIPS, 2013.

[5] S. Wang and C. Manning, "Fast dropout training," in *Proceedings of the 30th International Conference on Machine Learning*. PLMR, 2013.

[6] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 2575–2583.

[7] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on Machine Learning*. PLMR, 2016.

[8] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] 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.

[10] K. Neklyudov, D. Molchanov, A. Ashukha, and D. P. Vetrov, "Structured bayesian pruning via log-normal multiplicative noise," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 6775–6784.

[11] H. Salehinejad and S. Valaee, "Ising-dropout: A regularization method for training and compression of deep neural networks," *arXiv preprint arXiv:1902.08673*, 2019.

[12] H. Wu and X. Gu, “Towards dropout training for convolutional neural networks,” *Neural Networks*, vol. 71, no. C, pp. 1–10, 2015.

[13] Y. Gal and Z. Ghahramani, “A theoretically grounded application of dropout in recurrent neural networks,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS, 2016.

[14] S. Semeniuta, A. Severyn, and E. Barth, “Recurrent dropout without memory loss,” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, 2016.

[15] S. Merity, N. S. Keskar, and R. Socher, “Regularizing and optimizing LSTM language models,” *arXiv preprint arXiv:1708.02182*, 2017.

[16] G. Melis, C. Dyer, and P. Blunsom, “On the state of the art of evaluation in neural language models,” *arXiv preprint arXiv:1707.05589*, 2017.

[17] K. Żołna, D. Arpit, D. Suhubdy, and Y. Bengio, “Fraternal dropout,” *arXiv preprint arXiv:1711.00066*, 2018.

[18] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio, “An empirical analysis of dropout in piecewise linear networks,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[19] P. Baldi and P. J. Sadowski, “Understanding dropout,” in *Advances in Neural Information Processing Systems 26*, 2013, pp. 2814–2822.

[20] Y. Gal, J. Hron, and A. Kendall, “Concrete dropout,” in *Advances in Neural Information Processing Systems 30*, 2017, pp. 3581–3590.

[21] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks.” in *IEEE CVPR*. IEEE, 2015, pp. 648–656.

[22] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” *arXiv preprint arXiv:1302.4389*, 2013.

[23] S. Park and N. Kwak, “Analysis on the dropout effect in convolutional neural networks,” in *Asian Conference on Computer Vision*. Springer, 2016, pp. 189–204.

[24] T. DeVries and G. W. Taylor, “Improved regularization of convolutional neural networks with cutout,” *arXiv preprint arXiv:1708.04552*, 2017.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE CVPR*, 2016, pp. 770–778.

[26] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456.

[27] S. Cai, J. Gao, M. Zhang, W. Wang, G. Chen, and B. C. Ooi, “Effective and efficient dropout for deep convolutional neural networks,” *arXiv preprint arXiv:1904.03392*, 2019.

[28] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent neural network regularization,” *arXiv preprint arXiv:1409.2329*, 2015.

[29] H. Salehinejad, J. Baarbe, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent advances in recurrent neural networks,” *arXiv preprint arXiv:1801.01078*, 2017.

[30] T. Moon, H. Choi, H. Lee, and I. Song, “Rnndrop: A novel dropout for RNNs in ASR,” in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015.

[31] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[32] A. N. Gomez, I. Zhang, K. Swersky, Y. Gal, and G. E. Hinton, "Targeted dropout," in *2018 CDNNRIA Workshop at the 32nd Conference on Neural Information Processing Systems*. NeurIPS, 2018.

[33] S. Park, J. Park, S.-J. Shin, and I.-C. Moon, "Adversarial dropout for supervised and semi-supervised learning," 2018.

[34] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Adversarial dropout regularization," *arXiv preprint arXiv:1711.01575*, 2018.

[35] S. Park, K. Song, M. Ji, W. Lee, and I.-C. Moon, "Adversarial dropout for recurrent neural networks," *arXiv preprint arXiv:1904.09816*, 2019.

[36] S. J. Rennie, V. Goel, and S. Thomas, "Annealed dropout training of deep networks," in *IEEE Spoken Language Technology Workshop (SLT)*, 2014.

[37] S. Srinivas and R. V. Babu, "Generalized dropout," *arXiv preprint arXiv:1611.06791*, 2016.

[38] Y. Li and Y. Gal, "Dropout inference in bayesian neural networks with alpha-divergences," in *Proceedings of the 34th international conference on machine learning (ICML'17)*, 2017, pp. 2052–2061.

[39] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino, "Curriculum dropout," *arXiv preprint arXiv:1703.06229*, 2017.

[40] Z. Li, B. Gong, and T. Yang, "Improved dropout for shallow and deep learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS, 2016.

[41] A. Achille and S. Soatto, "Information dropout," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2897–2905, 2018.

[42] S. H. Khan, M. Hayat, and F. Porikli, "Regularization of deep neural networks with spectral dropout," *Neural Networks*, vol. 110, pp. 82–90, 2019.