Learning Sparse Networks Using Targeted Dropout

Aidan N. Gomez, Ivan Zhang, Siddhartha Rao Kamalakara, Divyam Madaan, Kevin Swersky, Yarin Gal, Geoffrey E. Hinton

Coby Penso

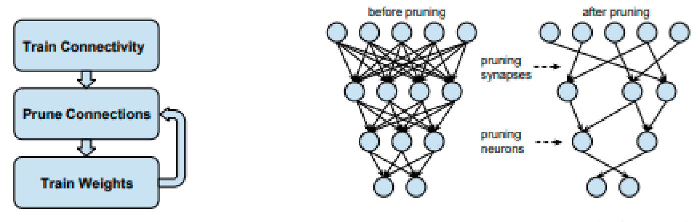
What we'll see

1. Neural Network Pruning
2. Dropout
3. Targeted Dropout
4. Related Work
5. Experiments and Results

Neural Network Pruning

Procedure of removing neurons or sub-networks with the goal of reducing the computational resources such as:

- memory complexity - size of the final network
- time complexity - inference time



Identifying Important Subnetworks

The result of targeted dropout is a reduction in the important subnetwork's dependency on the unimportant subnetwork, i.e Better performance after pruning.

Evaluation:

Observe the Taylor expansion of error after removing subnetwork *d*:

$$\Delta\epsilon = |\epsilon(\theta - d) - \epsilon(\theta)|$$

$$\Delta\epsilon = | - \nabla_{\theta}\epsilon^{T} d + \frac{1}{2} d^{T} H d + \Theta(||d||^{3})|$$

- Θ Parameters of the network
- d Deleted sub-network
- H Hessian matrix

Identifying Important Subnetworks

$$\Delta\epsilon = | - \nabla_\theta \epsilon^T d + \frac{1}{2} d^T H d + \Theta(||d||^3)|$$

Focus only on the middle term $\frac{1}{2} d^T H d$ since first term vanishes

Still a problem, H is intractable.

Solutions:

- **Weight independence**
  Sparse H with diagonal approximation - known as Optimal Brain Damage
- **First order approx.**
  Not really vanishes:
  Expectation - yes, Variance - informative $E[|\nabla_\theta \epsilon * d^T|]$
- **Zero order approx.**
  In practice works pretty well.

Magnitude-based pruning

Treat the top-k largest magnitude weights as important.

Notation:

$$\theta \in \Theta - \textit{Vector of NN parameters}$$

$$|\theta| - \textit{Number of parameters}$$

$$W \in \Omega_\theta - \textit{Weights connect one layer to another}$$

$$w_o \equiv W_{,o}$$

Unit Pruning:

$$W(\theta) = \left\{ \underset{w_o}{\text{argmax-k}} \, \|W_o\|_2 \,\middle|\, 1 \leq o \leq N_{col}(W),\ W \in \Omega_\theta \right\}$$

Weight Pruning:

$$W(\theta) = \left\{ \underset{w_{io}}{\text{argmax-k}} \, \|W_{io}\|_2 \,\middle|\, 1 \leq o \leq N_{col}(W),\ 1 \leq i \leq N_{row}(W),\ W \in \Omega_\theta \right\}$$

Dropout

We will consider two Bernoulli Dropout techniques:

- Dropout - Unit Dropout

$$Y = (X \odot M)W, \quad M \sim Bernoulli(1 - \alpha)$$

- DropConnect - Weight Dropout

$$Y = X(W \odot M), \quad M \sim Bernoulli(1 - \alpha)$$

Targeted Dropout

We hope to find optimal parameters Θ such that our loss $E(W(\Theta))$ is low, and at the same time $|W(\Theta)| <= k$.
A deterministic pruning selects the bottom $|\Theta|k$ elements and drop them out.

Add stochasticity into the process:

- Targeting proportion γ
- Drop probability α

Algorithm 1 Targeted Dropout - Training

1: **repeat**
2: Train for X epochs
3: Pick the $\gamma\,|\Theta|$ units/weights with lowest magnitude
4: Apply Dropout with probability α on them.
5: **until** Convergence

$$E(\mathit{Units\ To\ Keep}) = (1 - \gamma\alpha)|\Theta|$$

Known Pruning Techniques

We will cover the following techniques:

- L^1 Regularization
- L^0 Regularization
- Smallify
- Variational Dropout

These will be later used and compared to the suggested Targeted Dropout

L^1 Regularization

Bar-Ilan University

Using L^1 Regularization term in order to achieve sparsity in the weights.

L^1 Regularization drives and force more weights to be close to zero in magnitude.
Usually important against overfitting and being robust.
Here, used as a **sparsity and pruning** mechanism.

$$R(\theta) = \frac{1}{N}\left(\sum_{i=1}^{N} L(h(x_i;\theta), y_i)\right) + \beta||\theta||_1$$

In the Experminent the notation is L^1_β with β being the cost-balancing coefficient.

L^0 Regularization

L^0 Regularization encourage weights to become exactly zero. Use it as a regularization term in the objective function.

$$R(\theta) = \frac{1}{N}\left(\sum_{i=1}^{N} L(h(x_i;\theta), y_i)\right) + \lambda||\theta||_0, \quad ||\theta||_0 = \sum_{j=1}^{|\theta|} \mathbb{1}[\theta_j \neq 0]$$

Important to observe that it's non-differential.
In practice:

- Stochastic Gating Mechanism - reparameterization trick
- Approximate the expected L^0 regularized objective with smooth differential distribution

Bar-Ilan University

Combining the training and pruning in the same step, in order for the neuron to learn and adjust to the new architecture such that there is no need for post-training step.
The paper present a SwitchLayer which turns on/off neuron of the fly:

$$S_\beta(L(x))_{i,\ldots} = \beta_i L(x)_{i,\ldots} \; \forall i \in [1 \ldots c]$$

Algorithm 2 Smallify - Training

1: **for** $i = 1$ to N **do**
2: Train with SwitchLayer
3: Remove *off* neurons by some criteria
4: **end for**

Loss:

$$L_{SN}(x, y; \theta, \beta) = L(x, y; \beta) + \lambda ||\beta||_1 + \lambda_2 ||\theta||_p^p$$

Variational Dropout

Apply Gaussian dropout with trainable drop rates to the weights of the network and interprets the model as a variational posterior with a particular prior.

The authors note that the variational lower bound used in training favors higher drop probabilities and experimentally confirm that networks trained in this way do indeed sparsify.

Experiments and Results

Experiments

The following architectures covered in order to check the effect of Targeted Dropout method:

- ResNet - with CIFAR-10 and ImageNet
- Wide ResNet -With CIFAR-10 and ImageNet
- Transformer - with WMT English-German Translation

Toy example - single dense hidden layer with ten units and ReLUs, on CIFAR-10.

- Left - Unregularised
- Right - Targeted dropout, with $\gamma = 75\%$ and $\alpha = 50\%$.

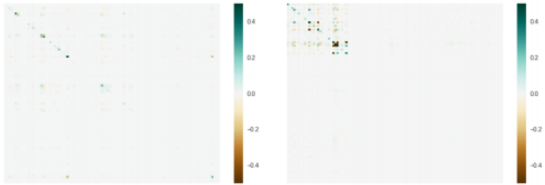


Figure 1: A comparison between a network without dropout (left) and with targeted dropout (right) of the matrix formed by $\theta^\top \odot \boldsymbol{H} \odot \theta$. The weights are ordered such that the last 75% are the weights with the lowest magnitude (those we intend to prune). The sum of the elements of the lower right hand corner approximates the change in error after pruning (Eqn. (3)). Note the stark difference between the two networks, with targeted dropout concentrating its dependence on the top left corner, leading to a much smaller error change after pruning (given in Table 1).

Regularisation	$\lvert\Delta\mathcal{E}\rvert$	Unpruned Accuracy	Pruned Accuracy
None	0.120698	38.11%	26.13%
Targeted Dropout	0.0145907	40.09%	40.14%

Table 1: Comparison of the change in loss ($\lvert\Delta\mathcal{E}\rvert$ of Equation (3)) for dense networks.

ResNet

ResNet-32 on CIFAR-10

Weight Dropout/Pruning

prune percentage	none	dropout $\alpha=0.25$	targeted $\alpha=0.5,\gamma=0.5$	targeted $\alpha=0.33,\gamma=0.75$	targeted $\alpha=0.66,\gamma=0.75$	targeted $\alpha=0.75,\gamma=0.90$	variational	$L^1_{0.1}$	$L^0_{0.1}$
0 %	93.71	93.62	93.03	89.88	92.64	92.53	92.09	92.80	88.83
10%	93.72	93.63	93.04	89.80	92.62	92.55	92.00	92.72	90.66
20%	93.77	93.66	93.02	89.93	92.63	92.48	92.02	92.84	88.64
30%	93.59	93.58	92.98	89.89	92.66	92.53	92.07	92.63	87.16
40%	93.09	93.45	93.03	89.75	92.70	92.63	92.12	92.80	85.31
50%	92.20	93.07	92.99	89.72	92.65	92.54	91.84	92.29	80.94
60%	90.46	90.81	92.66	89.84	92.70	92.55	91.48	91.20	69.48
70%	81.88	72.29	92.22	89.80	92.66	92.56	90.23	86.30	46.19
80%	32.02	19.84	84.03	85.80	91.86	92.54	83.44	63.00	23.71
90%	14.63	10.05	28.27	27.04	67.58	92.48	15.16	21.08	12.55

Unit Dropout/Pruning

prune percentage	none	dropout $\alpha=0.25$	targeted $\alpha=0.5,\gamma=0.5$	targeted $\alpha=0.33,\gamma=0.75$	targeted $\alpha=0.66,\gamma=0.75$	targeted $\alpha=0.90,\gamma=0.75$	variational	$L^1_{0.01}$	$L^0_{0.01}$
0 %	93.69	92.43	92.21	90.46	89.38	89.78	93.14	93.31	93.35
10%	90.05	67.52	91.96	88.44	89.48	90.18	92.91	91.03	83.01
20%	80.34	25.05	91.63	83.55	88.89	89.79	90.38	85.63	54.59
30%	59.94	13.47	91.30	69.82	88.84	89.88	86.38	72.19	21.34
40%	35.40	10.02	89.89	54.42	87.54	89.98	83.59	46.41	10.82
50%	12.63	9.97	88.41	28.88	84.86	90.05	65.79	26.72	15.04
60%	10.65	9.99	26.55	18.55	81.98	90.08	41.05	12.11	9.46
70%	11.70	10.01	17.41	17.84	75.47	90.03	19.36	11.81	10.02
80%	9.99	9.95	10.63	10.87	28.99	34.18	9.56	14.73	14.88
90%	9.85	9.98	9.30	10.29	9.97	10.04	10.41	10.22	9.98

Transformer

Bar-Ilan University

Robustness of **Targeted Dropout** to various of architectures,
for example - Transformer.

Weight Dropout/Pruning

prune percentage	none	targeted $\alpha=0.66, \gamma=0.75$	targeted $\alpha=0.66, \gamma=0.90$
0 %	26.01	26.52	25.32
10%	26.05	26.44	25.32
20%	25.90	26.48	25.19
30%	25.91	26.30	25.27
40%	25.81	26.20	24.97
50%	25.08	26.03	24.93
60%	23.31	25.62	24.27
70%	8.89	24.07	22.41
80%	0.24	12.39	10.57
90%	0.01	0.07	0.64

(a) Transformer model uncased BLEU score.

Weight Dropout/Pruning

prune percentage	none	targeted $\alpha=0.66, \gamma=0.75$	targeted $\alpha=0.66, \gamma=0.90$
0 %	62.29	58.31	57.41
10%	62.54	59.00	58.10
20%	62.21	59.39	58.52
30%	62.33	58.66	57.86
40%	61.81	59.39	58.67
50%	60.82	57.71	57.08
60%	58.13	58.42	57.96
70%	48.40	55.39	54.85
80%	25.80	47.09	46.63
90%	6.90	21.64	27.02

(b) Transformer model per-token accuracy.

Table 5: Evaluation of the Transformer Network under varying sparsity rates on the WMT newstest2014 EN-DE test set.

Scheduling the Targeting Proportion

Bar-Ilan University

Ramping targeted dropout - Anneals the targeting rate γ from zero to specific γ through the course of training.

Weight Dropout/Pruning

prune percentage	targeted $\alpha=0.66, \gamma=0.75$	smallify $\lambda=0.00001$
0 %	92.64	90.16
10%	92.62	90.13
20%	92.63	90.16
30%	92.66	90.06
40%	92.70	90.17
50%	92.65	90.20
60%	92.70	90.12
70%	92.66	90.10
80%	91.86	90.15
90%	67.58	90.16

prune percentage	smallify $\lambda=0.00001$	ramp targ $\alpha=0.99, \gamma=0.99$
90%	90.20	89.03
91%	90.33	89.16
92%	90.30	89.14
93%	90.27	89.03
94%	89.46	89.05
95%	89.41	89.05
96%	88.55	89.02
97%	86.35	89.05
98%	59.27	89.05
99%	13.83	88.97

prune percentage	ramp targ $\alpha=0.99, \gamma=0.99$
98.5%	89.03
98.6%	89.08
98.7%	89.00
98.8%	89.05
98.9%	88.99
99.0%	89.10
99.1%	88.35
99.2%	79.88
99.3%	77.35
99.4%	16.55

Unit Dropout/Pruning

prune percentage	targeted $\alpha=0.66, \gamma=0.75$	smallify $\lambda=0.0001$	ramp targ $\alpha=0.99, \gamma=0.90$
0 %	90.55	90.20	85.98
10%	90.83	90.33	86.12
20%	89.88	90.30	86.01
30%	87.35	90.27	86.10
40%	85.39	89.46	85.98
50%	80.84	89.41	86.13
60%	71.97	88.55	86.02
70%	55.98	86.35	86.08
80%	10.02	59.27	85.95
90%	10.07	13.83	85.99

Table 6: Comparing Smallify to targeted dropout and ramping targeted dropout. Experiments on CIFAR10 using ResNet32.

Random Pruning vs. Targeted Dropout

- Random-pruning: prune random subnetwork before training.
- Ramping Targeted Dropout

Type	Prune % 50%	75%	90%	99%
Random-prune	92.58	92.32	90.66	80.86
Ramping TD	**93.29**	**92.72**	**92.51**	**88.80**

(a) Comparison of weight-level pruning methods using ResNet-32 trained on CIFAR-10.

Type	Prune % 75%	85%	90%	95%
Random-prune	90.50	88.52	84.98	79.09
Ramping TD	**90.84**	**88.59**	**86.45**	**80.65**

(b) Comparison of unit-level pruning methods using ResNet-32 trained on CIFAR-10.

Type	Prune % 75%	85%	90%	95%
Random-prune	48.98 (0.62)	45.58 (1.25)	40.50 (2.03)	31.44 (1.64)
Ramping TD	**52.64** (0.61)	**49.20** (0.10)	**45.03** (0.83)	30.15 (1.72)

(c) Comparison of unit-level pruning methods using VGG-16 trained on CIFAR-100. Results are the average of five independent training runs followed by one standard deviation reported in brackets.

Table 7: Comparison between random pruning at the beginning of training and regularising with targeted dropout throughout the course of training, followed by post hoc pruning.

Questions?

Bibliography

- https://arxiv.org/pdf/1905.13678.pdf
- https://arxiv.org/pdf/1506.02626.pdf
- https://proceedings.neurips.cc/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf
- https://arxiv.org/pdf/1207.0580.pdf
- https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf
- http://proceedings.mlr.press/v28/wan13.pdf
- https://arxiv.org/pdf/1611.06440.pdf
- https://arxiv.org/pdf/1701.05369.pdf
- https://arxiv.org/pdf/1506.02557.pdf
- https://arxiv.org/pdf/1806.03723.pdf