SNIP: single-shot network pruning based on connection sensivity

Reproduction and analysis by Szymon Mikler, University of Wroclaw (extended abstract)

1 Description of SNIP pruning method

One can describe neural network's loss sensitivity on zeroing a weight w_j as $\widehat{s_j} = |L(w_j) - L(0)|$. We will call it the **exact salience** from now on. Intuitively, if this sensitivity is small, then removing the parameter shouldn't prevent neural network from learning, since it didn't affect the loss function anyway. Although, one usually wants to remove more than one parameter and, since they are dependent on each other, in order get the least loss affecting set of η parameters to prune, one needs to check every possible combination of η parameters, thus make this many forward passes in a network. Since this is computationally impossible, we are assuming that influence of w_j on $\Delta L = |L(w_j) - L(0)|$ is independent from every other weight than w_j and this is the first of two assumptions we make.

Since computing $\widehat{s_j}$ for every weight w_j separately is also computationally expensive – it requires m forward passes to compute (for m being number of parameters in the network) – we will assume that $|L(w_j) - L(0)|$ corresponds to $|L(w_j) - L(w_j - \varepsilon)|$ for infinitesimally small ε.

This second assumption is the one that describes the SNIP method. With it we can define a salience s_j of a weight w_j as $s_j = \frac{\partial L(w_j \cdot c_j)}{\partial c_j}$ for c_j being an indicator variable equal to 1. Such salience calculation can be done in modern frameworks very efficiently, for all weights at once. Salience can be then standardized to show parameters' importances in percentages. After evaluating the saliences, one can set the indicator c_j to 0 for selected connections.

2 Training algorithm

1. Randomly initialize weights, usually with Xavier initialization (either uniform or normal),
2. One-shot prune the network to a given sparsity level κ. In other words, set to 0 all the parameters w_j for which the salience $s_j \le s_\kappa$, where s_κ is κ-th salience in terms of magnitude out of all from the network,
3. Train the network with standard training techniques and with any optimizer.

The fact that we can use any training techniques is crucial advantage here. We don't need to adjust our training schedule. Implementation details, if needed, can be found in code[1].

3 Simplification and generalization

This is a simplification of the original SNIP salience formula: $\frac{\partial L}{\partial c_i} = \frac{\partial L}{\partial w_i} \cdot w_i$. Derivation of it is very simple and it is based on the fact that saliences are equal to 1 before the pruning. Here we show the most important part:

$$\frac{\partial L}{\partial c_i} = \frac{\partial L}{\partial (c_i \cdot w_i)} \cdot \frac{\partial (c_i \cdot w_i)}{\partial c_i} = \frac{\partial L}{\partial w_i} \cdot w_i$$

There is a good reason to derive this formula - this notation of SNIP method is crucial for efficient implementation, since $\frac{\partial L}{\partial w_i}$ are vanilla gradients, the same that are calculated during each backward pass. Also, looking at this equation, it is possible to derive a higher order approximations of the exact salience by using higher order polynomials (or other functions). It turns out that SNIP approximation is a linear approximation (since it assumes a constant derivative $\frac{\partial L}{\partial w_i}$ for $w \in [0, w_i]$). We can derive the formula for SNIP by assuming the loss function $L(w_i)$ is a linear function with respect to the each of the parameters. In the same way, we can derive formulas for higher order SNIP - by assuming the loss function is in polynomial relation with the parameters. Modern computers are able to calculate the diagonal approximation of the Hessian matrix nearly as quick as the first order gradients and this allows us to calculate higher

[1] `https://github.com/gahaalt/ICLR-SNIP-pruning`

order approximation of the exact salience. This is why we suggest the following extension of the method:

Let w_0 be the initial value of parameter w. Now let $L(w) = aw^2 + bw + c$, then $|L(w_0) - L(0)| = |aw_0^2 + bw_0|$. We can derive $\frac{\partial L(w)}{\partial w} = 2aw + b$, then $b = \frac{\partial L(w)}{\partial w} - 2aw$. Since we calculated the derivative value in w_0 only, $b = \frac{\partial L}{\partial w}(w_0) - 2aw_0$. Also $\frac{\partial^2 L(w)}{\partial w^2} = 2a$, from which we derive parameter $a = \frac{1}{2}\frac{\partial^2 L}{\partial w^2}(w_0)$. Finally, estimated salience is:

$$|L(w_0) - L(0)| = |aw_0^2 + bw_0| = |\frac{1}{2}\frac{\partial^2 L}{\partial w^2}(w_0) \cdot w_0 + \frac{\partial L}{\partial w}(w_0) - 2aw_0| = |\frac{\partial L}{\partial w}(w_0) - \frac{1}{2}\frac{\partial^2 L}{\partial w^2}(w_0) \cdot w_0|$$

Given the loss function is quadratic with respect to the parameters, calculated salience will be equal to the exact salience. Similar calculations can be performed for any differentiable function, but higher order polynomials will require calculating higher order Hessians. For the original SNIP salience, the estimated salience will be equal to the exact one under stronger assumption - when the loss function is linear with respect to each of the parameters.

4 SNIP vs exact salience

This is an empirical comparison between SNIP and the exact salience $|\Delta L|$ that SNIP, as stated at the beginning, tries to approximate. The tests were conducted on rather small architectures. The results are interesting - they indicate that SNIP approximates the exact salience almost perfectly.

It is worth to mention, even though exact salience is much more computationally expensive to calculate than SNIP, it is possible to compute it for big networks like VGG in matter of minutes. It is possible, that if the network architecture was more complicated, SNIP wouldn't be as good of an approximation of it as it is. Nevertheless, for simple architectures it seems as using higher order approximations of $|\Delta L|$ is unnecessary.

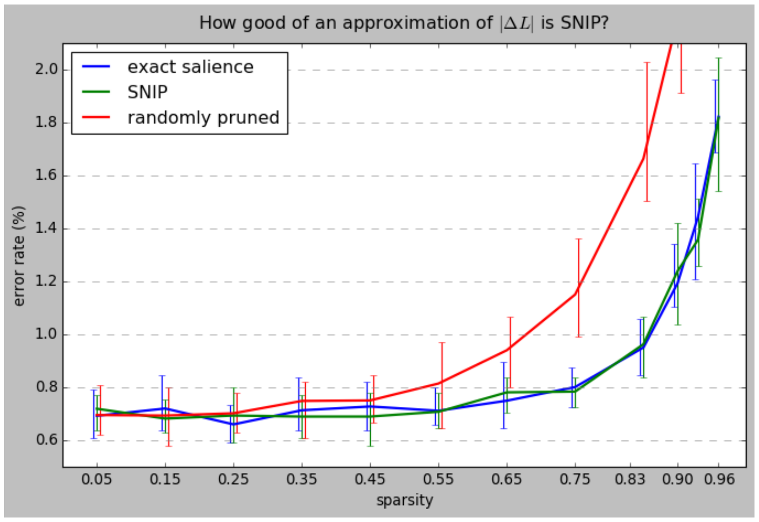

Figure 1: Comparison between exact salience criterion and SNIP

5 Inner working of SNIP

In the case of simple feedforward network, our results indicates that SNIP actually knows how many parameters in which layer the network needs for learning to proceed. One could use such statistics at the end of the training to conclude which layers are overparametrizied and have no effect on the loss. As we saw, the exact salience is approximated well by SNIP method, thus it should give very accurate results.

It is worth to mention that these structures stay consistent. In other words, parameters quantities on layers stay roughly the same when SNIP procedure is run for the second time with other set of randomly initialized parameters. During the whole analysis, pruning results were examined on many levels, from whole network statistics, to heatmaps of separate kernels in convolutional layers. Probability distribution of the remaining parameters was also analyzed. We believe that based on these observations it is possible to derive simple rules of thumb, allowing to build sparse networks as good as these pruned by SNIP.

layer	**params**	**50% sparse**	**95% sparse**
conv 1	150	143 (04.7%)	106 (29.3%)
bias conv 1	6	6 (00.0%)	3 (50.0%)
conv 2	2400	2118 (11.8%)	666 (72.3%)
bias conv 2	16	16 (00.0%)	6 (62.5%)
full 1	48000	22815 (52.5%)	1214 (97.5%)
bias full 1	120	94 (21.7%)	22 (81.7%)
full 2	10080	5022 (50.2%)	768 (92.4%)
bias full 2	84	66 (21.4%)	41 (51.2%)
full 3	840	563 (33.0%)	253 (69.9%)
bias full 3	10	10 (00.0%)	7 (30.0%)
result	0.65%	0.55%	1.33%

Figure 2: LeNet-5 layer sparsities statistics after pruning with SNIP with weights' saliences evaluated on whole batch of MNIST dataset and validation accuracies for comparison.