RANDOMOUT: USING A CONVOLUTIONAL GRADIENT NORM TO WIN THE FILTER LOTTERY

Joseph Paul Cohen & Henry Z. Lo & Wei Ding
Department of Computer Science
University of Massachusetts Boston
Boston, MA 02125, USA
{joecohen,henryzlo,ding}@cs.umb.edu

ABSTRACT

Convolutional neural networks are sensitive to the random initialization of filters. We call this The Filter Lottery (TFL) because the random numbers used to initialize the network determine if you will "win" and converge to a satisfactory local minimum. This issue forces networks to contain more filters (be wider) to achieve higher accuracy because they have better odds of being transformed into highly discriminative features at the risk of introducing redundant features. To deal with this, we propose to evaluate and replace specific convolutional filters that have little impact on the prediction. We use the gradient norm to evaluate the impact of a filter on error, and re-initialize filters when the gradient norm of its weights falls below a specific threshold. This consistently improves accuracy across two datasets by up to 1.8%. Our scheme RANDOMOUT allows us to increase the number of filters explored without increasing the size of the network. This yields more compact networks which can train and predict with less computation, thus allowing more powerful CNNs to run on mobile devices.

1 INTRODUCTION

When training a convolutional neural network LeCun & Bengio (1995) LeCun et al. (2015) with different random seeds for initialization (*ceteris paribus*), we observe that the seed affects both the quality of the learned convolutional filters and the amount of generalization error on the testing set. We call this issue *The Filter Lottery* because the random numbers used to initialize the network determine if you will "win" and converge to a satisfactory local minimum.

The accuracy obtained by varying random seeds on a simple convolutional neural network is shown in Figure 1. By simply changing the random initialization seed, we observe high variation in testing accuracy. Although most random seeds result in an acceptable local error minimum, the variance around that minimum is noticeable and sometimes the accuracy is as low as random chance (5% in our study). These results are to be expected because we are minimizing a non-convex loss function which we expect to have many local minimums or saddle points that cause convergence behavior similar to that of local minimum Dauphin et al. (2014).

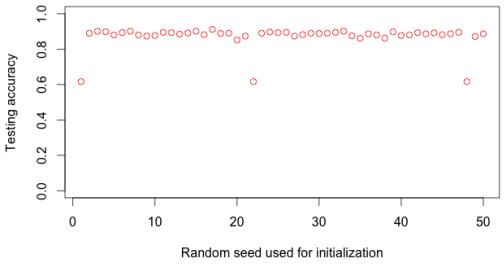

Figure 1: The testing accuracy of the network is plotted while varying nothing but the random seeds used to initialize the network.

We suspect this is due to the network not learning the filters needed to extract the most discriminative features. We have observed that varying the random seed can change how the filters will converge. With ReLU activations, later layers will reduce the contribution of underperforming filters on the predictions. These filters have been "abandoned" by the later layers because they contribute little or negatively to minimizing the error. Making the network wider (by adding more filters) increases performance. We believe that this is because the addition of these filters allows the network to

successfully capture more discriminative features of the dataset. This would be unnecessary if all filters were utilized instead of being abandoned.

We propose the method RANDOMOUT that ranks and replaces filters if they have been abandoned by the network. If a filter's contribution to the objective is insignificant then we reinitialize it with random values and continue learning. This allows the weights to learn a completely different filter which will give the network another chance to reach an acceptable local minimum. Our scheme allows us to increase the number of filters explored without increasing the size of the network. This yields more compact networks which can train and predict with less computation, thus allowing more powerful CNNs to run on mobile devices.

2 METHOD

We use the convolutional gradient norm (CGN) Lo et al. (2015) to determine how much impact a filter has on the overall classifiacation loss function by taking the derivative of the loss function with respect each weight in the filter. We take derivatives w.r.t. overall classification error because our desire is to determine the influence on every class and not just a specific one.

$$CGN(k) = \sum_i \frac{\partial L}{\partial w_i^k}$$

Here we sum the derivative of each weight w_i in filter k to produce an aggregate number representing that filter's impact on the loss function L. When $CGN(k)$ is low and overall network error is low, filter k has learned converged successfully. This filter will not be modified in the future and there is no need because the overall network performance is good. In contrast when the CGN of a specific filter is low but the overall error is high then we say that the network has abandoned this filter. That is, the filter will not change much as the network continues to train. This is undesirable as overall error is high because we are most likely lacking an important filter so we should randomize it. However, abandoned filters may not always be bad filters if they happen to be very easy to learn and then only cause a very small error. However but randomizing these will result in the same filter or a more needed filter to reduce the overall error. Because the overall error is high then the network might be better off finding a different filter that can better reduce the overall error.

This approach calls for two hyperparameters, a threshold τ and a "% of epochs active" $\mathcal{P}$. During training each filter k is checked at regular intervals to see if $|CGN(k)| < \tau$ and if so filter k is randomized. The motivation for the threshold is that the CGN is hardly ever 0, because learning rates are fractional so update rules only approach 0, but will become very close when the network has stopped learning a filter. For example, in our study $\tau = 10^{-8}$ yields good results. It is also necessary to consider the number of epochs from the start of training should the filters be randomized; we refer to this as % of epochs active or $\mathcal{P}$. The motivation for this hyperparameter is that performing this randomization may damage the network and it will need time to retrain itself.

3 EXPERIMENTS

In order to explore how this approach can improve the performance of a network we evaluate it on a simple convolutional network shown in Figure 2 presented in Cohen et al. (2016) and explained below. We train and test the network on a well known Martian Crater dataset Bandeira et al. (2010). We made these decisions because the network and dataset are sufficiently challenging enough to make our point while small enough to perform the training of >14,000 networks (>2 million epochs) required for this analysis. We use two subsets of the data (the East and West regions) for our experiments which are split 50/50 into a training

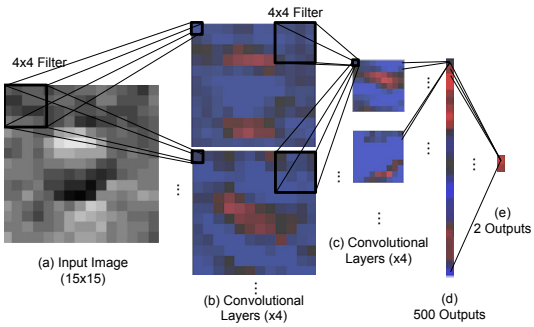

Figure 2: Crater Convolutional Neural Network (CNN) architecture computation graph.

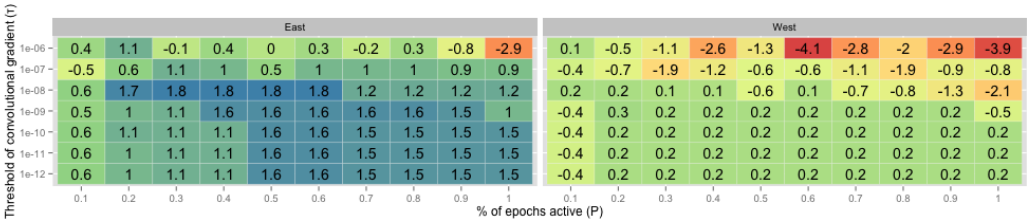

Figure 3: Resulting accuracy gain of when using two hyperparameters of extscRandomOut threshold and % of epochs are varied. Each cell value in the heatmap is the mean gain of of 50 different random seeds when using RANDOMOUT.

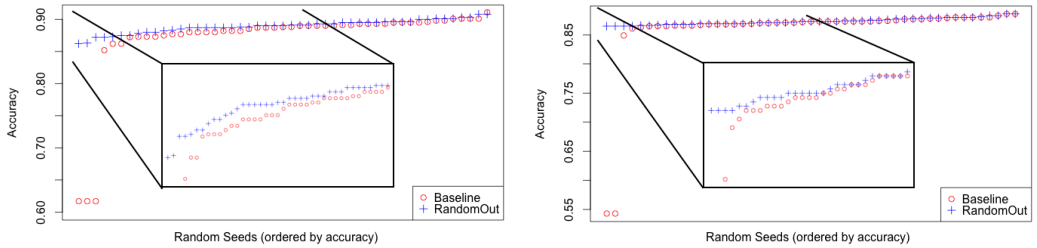

Figure 4: Evaluation of 50 random seeds sorted by accuracy and shown side by side when using RANDOMOUT and when not. Left: the East region with RANDOMOUT was run with $\tau = 1 \times 10^{-8}$ with $\mathcal{P} = 60\%$. Right: the West region with RANDOMOUT was run with $\tau = 1 \times 10^{-12}$ with $\mathcal{P} = 90\%$.

and test set. The East region contains 458 positive and 765 negative labeled crater examples while the West region contains 1121 positive and 1385 negative crater examples.

The CNN is implmented in Deeplearning4j Team (2015) has two convolutional layers, followed by a fully connected layer, then softmax (see Figure 2). The input is 15x15 grayscale image. Each crater candidate example is scaled to this size. Layers (b) and (c) are convolutional layers with 20 stride-1 4x4 filters. We did not use pooling as it did not perform well. Layer (d) is a fully connected layer where each element is a ReLU. Layer (e) has just two outputs; each corresponding to a class. A softmax regression is used so that the results can be interpreted as a probability distribution between the classes (craters and non-craters). The initial weights throughout the network are initialized using the Glorot & Bengio (2010) scheme and trained using gradient decent.

We evaluate RANDOMOUT on 50 random seeds used for initialization over 150 training epochs. We vary τ and $\mathcal{P}$ and study the resulting test accuracy in Figure 3. We can observe that generally for a fixed $\mathcal{P}$ a lower threshold τ value results in a higher average gain in network accuracy. This is because these have been correctly identified filters as being abandoned and brought them back to life to improve the network. We also find that increasing $\mathcal{P}$, provided a low threshold τ, yields a higher gain. We understand this to mean that the lower the threshold is the lower the risk of randomizing a filter that has learned an important feature. In Figure 4 we compare the best hyperparameters found for RANDOMOUT with the baseline networks. Note that the baseline numbers shown for the East region are those in Figure 1 and that RANDOMOUT recovered all networks that failed to converge to an acceptable accuracy. We draw the readers attention to the effectiveness of our method in recovering from all bad seeds and increasing accuracy overall. It is also signifcant to note that these improvements are not brittle and just for specific hyperparameters but can be observed over large areas of the parameter spaces as shown in the green and blue cells in Figure 3.

ACKNOWLEDGMENTS

We thank Michael Wick of UMass Amherst and Swami Iyer of UMass Boston for their feedback. This work is partially funded by a grant from the National Science Foundation Graduate Research Fellowship Program (grant number: DGE-1356104). This work utilized the supercomputing facilities managed by the Research Computing Department at the University of Massachusetts Boston.

REFERENCES

L. Bandeira, W. Ding, and T. F Stepinski. Automatic Detection of Sub-km Craters Using Shape and Texture Information. In *Proceedings of the 41st Lunar and Planetary Science Conference*, March 2010.

Joseph Paul Cohen, Henry Z. Lo, Tingting Lu, and Wei Ding. Crater Detection via Convolutional Neural Networks. In *Lunar and Planetary Institute Science Conference Abstracts*, volume 47, 2016.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems 27*, pp. 2933–2941. Curran Associates, Inc., 2014.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Henry Z. Lo, Joseph Paul Cohen, and Wei Ding. Prediction gradients for feature extraction and analysis from convolutional neural networks. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pp. 1–6, May 2015.

Deeplearning4j Development Team. Deeplearning4j: Open-source distributed deep learning for the JVM, 2015. URL `http://deeplearning4j.org`.