ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 26-28 April 2017, i6doc.com publ., ISBN 978-287587039-1. Available from http://www.i6doc.com/en/.

Random projection initialization for deep neural networks

Piotr Iwo Wójcik and Marcin Kurdziel [*]

AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland

Abstract. In this work we propose to initialize rectifier neural networks with random projection matrices. We focus on Convolutional Neural Networks and fully-connected networks with pretraining. Our results show, that in convolutional networks a well designed random projection initialization can perform better than the current state-of-the-art He's initialization. Specifically, in our evaluation, initialization based on the Subsampled Randomized Hadamard Transform consistently outperformed He's initialization on several evaluated image classification datasets.

1 Introduction

The choice of initial neural network parameters, i.e., weights and biases, can significantly affect the speed of convergence and the quality of the found solution. This issue, related to backpropagation through deep networks, prompted a vigorous research on network initialization. Glorot and Bengio [1], for example, proposed initialization schemes for sigmoid and hyperbolic tangent activation functions. Their initialization was one of the factors that made it possible to successfully train deep networks from scratch [2]. A weight initialization scheme for rectifier linear units (ReLU) was given in [3] - the so called *He's initialization*. He's initialization is currently the state-of-the-art for practical applications in ReLU networks. Comparatively less work has been published on initialization of weights as a starting point for generative pretraining. Weights before pretraining are typically densely initialized with random values drawn from a zero-mean normal distribution with small standard deviation [4].

In this work we study initialization of deep neural networks with random projection (RP) matrices. In particular, we investigate several RP initialization schemes in Convolutional Neural Networks (CNNs) and in fully-connected networks with pretraining. We focus on rectifier networks, as the ReLU transfer function is currently the most popular choice for efficient training of deep architectures. Our results show that using RP matrices as initial weights in CNNs

[*]This research is supported by the Polish National Science Centre grant no. DEC-2013/09/B/ST6/01549 "Interactive Visual Text Analytics (IVTA): Development of novel, user-driven text mining and visualization methods for large text corpora exploration." This research was carried out with the support of the "HPC Infrastructure for Grand Challenges of Science and Engineering" Project, co-financed by the European Regional Development Fund under the Innovative Economy Operational Programme. This research was supported in part by PL-Grid Infrastructure.

can improve performance over the current state-of-the-art initialization scheme. Pretrained networks, however, do not benefit from RP initialization.

2 Initializing weights with random projection matrices

In random projection, a dataset $\mathbf{A} \in \mathbb{R}^{n \times d}$ is projected from a high-dimensional space $\mathbb{R}^d$ into a lower-dimensional space $\mathbb{R}^k$ using a random matrix $\mathbf{R} \in \mathbb{R}^{d \times k}$: $\tilde{\mathbf{A}} = \mathbf{AR}$. The simplest construction for the matrix $\mathbf{R}$ is the Gaussian random matrix, whose entries are drawn from a zero-mean normal distribution [5]. Note, that this construction is related to network initialization with small, fixed-variance Gaussian weights, which historically was a popular choice for initial weights. To see this, consider an untrained network layer with d input units and k output units, $k \leq d$, whose weight matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$ has been initialized from $N(0, \sigma^2)$ and the biases have been set to zero. The layer receives an input vector $\mathbf{I} \in \mathbb{R}^d$ and computes the output $\mathbf{O} \in \mathbb{R}^k$: $\mathbf{O} = f(\mathbf{IW})$, where f is the activation function. The computation realized by this layer is similar to performing a Gaussian RP of the input vector using $\mathbf{W}$ as the random projection matrix $\mathbf{R}$. The only difference is the application of an element-wise activation function f, as matrices $\mathbf{W}$ and $\mathbf{R}$ are constructed in exactly the same manner: they consist of random numbers drawn from a Gaussian distribution with a small variance. For nonlinear activations that are well approximated by a linear function near zero, e.g., sigmoid or hyperbolic tangent, applying f has little impact on the result. In case of ReLU applying f zeroes-out the negative entries of $\mathbf{IW}$. However, the resultant vector $\mathbf{O}$ still contains on average half of the random projection coordinates.

A network consisting of l hidden layers performs l consecutive Gaussian random projections, resulting in a k-dimensional data representation, where k is the size of the last hidden layer. As the Gaussian RP satisfies the Johnson–Lindenstrauss Lemma [5], unless k is small, feeding data through a Gaussian-initialized network yields an output that roughly preserves the structure of the original data. This is supported by recent findings that random untrained weights can perform surprisingly well in certain convolutional network architectures [6]. We argue that this observation may, to some extent, be attributed to the structure-preserving nature of an embedding realized by untrained networks with Gaussian weights.

While the above argument was presented for the Gaussian initialization, it also holds for other random projection schemes. This motivates us to explore the prospects of initializing the weights in neural networks with more intricate random matrices. In particular, we consider four RP matrix constructions: Achlioptas' [7], Li's [8], Subsampled Randomized Hadamard Transform [9] (SRHT) and a construction based on Count Sketch [10].

When initializing CNNs, following [11], we only initialize weights in the convolutional layers, and set the biases to zero. Consider a weight matrix of size: $c_{out} \times c_{in} \times k \times k$, where c_{in} and c_{out} are the number of input and output activation maps, respectively, and k is the size of the convolution kernel. In RP-based initialization we use $n = c_{in} \times k \times k$ as the data dimensionality and c_{out} as the projection dimensionality. Therefore, before training, the convolutional layer can be seen as performing RP of the input volume into a c_{out}-dimensional space. We normalize the weight matrices to the same standard deviation as in He's initialization, i.e., to $\sqrt{2/n}$. The exception is the Count Sketch initialization, where, due to its sparsity, we multiply the weights with a scale factor γ chosen with validation experiments. In SRHT initialization we set the sparsity parameter [9] to $q = \frac{log^2 N}{n}$. For the number of projected examples N we assume the number of training examples in a single epoch.

We also investigated initialization in fully-connected networks with generative pretraining. For the pretraining phase we employ stacked Restricted Boltzmann Machines (RBMs) [12]. Random projection initialization is performed before the pretraining phase, and thus serves as the starting point for the Contrastive Divergence algorithm. When initializing the weights in an RBM with RP matrix elements, we use the number of visible units as the data dimensionality and the number of hidden units as the projection dimensionality. Similarly to CNN initialization, we leave the biases set to zero. Following [4], after initialization we normalize the weight matrices to zero mean and a small standard deviation. The Count Sketch initialization is, again, an exception - because of its sparsity we scale the weights by a constant factor chosen with validation experiments. In SRHT initialization we set the sparsity parameter q as in CNNs.

3 Experiments

For the evaluation of RP initialization in CNNs we employed three datasets, namely CIFAR-10, CIFAR-100 and SVHN. The evaluation was carried out using Residual Networks with stochastic depth [11]. Note, however, that RP initialization proposed herein is not tailored for this specific network architecture, but can be used in any rectifier network. Our aim is not to improve beyond the current state-of-the-art results on CIFAR or SVHN but to find out whether modern CNN architectures can benefit from RP initialization. We use ResNets with stochastic depth as an example of such architecture. For our experiments we employed the best-performing architectures and hyper-parameter sets from [11], i.e., 110-layer ResNets for the CIFAR experiments and a 152-layer ResNet for the SVHN experiments. With the reference He's initialization these models achieved state-of-the-art performance on CIFAR-10 and CIFAR-100 with standard data augmentation, and the second best published result on SVHN. To account for the random nature of weight initialization, we trained ten network instances for each initialization method, using different random number generator seeds. Afterwards, we carried out statistical tests to assess the confidence of RP initialization schemes outperforming the He's initialization. Specifically, for each network instance we averaged the test errors from the last 100 (CIFAR) or 10 (SVHN) epochs, and compared the averages obtained with the He's initialization against averages obtained using RP initialization. For the comparison we employed the Wilcoxon-Mann-Whitney two-sample rank-sum test. All experiments

ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 26-28 April 2017, i6doc.com publ., ISBN 978-287587039-1. Available from http://www.i6doc.com/en/.

were carried out using an implementation of ResNets with stochastic depth by Yu Sun[1].

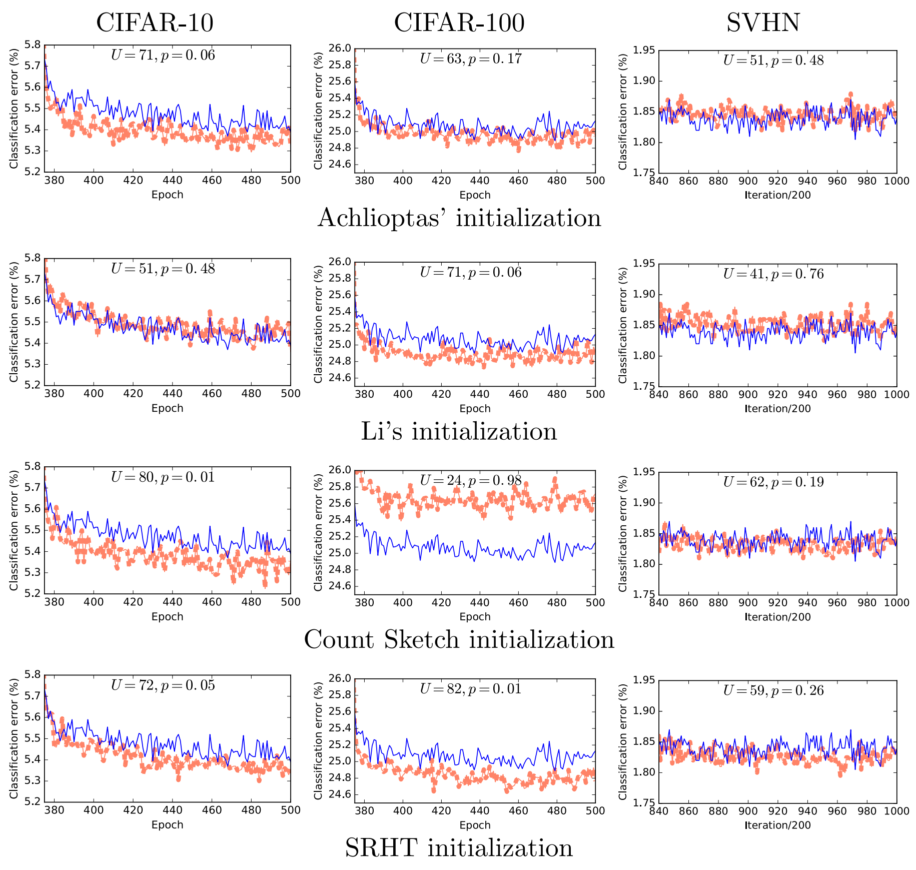

Fig. 1: Median test error on CIFAR and SVHN for different RP initializations (dashed orange line) and the reference He's initialization (solid blue line).

Performance of the evaluated initialization schemes is reported in Fig. 1. The plots report median test accuracy, the significance level of the hypothesis that the evaluated RP initialization outperforms He's initialization and the value of the U statistic in the Wilcoxon-Mann-Whitney test. Following [11], we report the test error in each epoch for CIFAR, and after every 200 iterations for the SVHN experiments. The SRHT initialization outperformed the reference initialization on all datasets. The other initialization schemes yielded mixed results.

Random projection initialization in pretrained networks was evaluated on MNIST and Jittered-Cluttered NORB datasets. For the MNIST experiments we employed one of the network architectures from [13], namely binary input layer followed by two hidden layers with 1000 ReLU units and a 10-way softmax.

[1]Available at `https://github.com/yueatsprograms/Stochastic_Depth`

For the NORB experiments we used the best performing network architecture reported in [14], i.e., two hidden layers with 4000 and 2000 ReLU units, respectively, followed by a 6-way softmax. Inputs in this network were modeled with Gaussian units. Evaluated networks were pretrained with the Contrastive Divergence algorithm and finetuned with error backpropagation. Learning hyperparameters for these networks were selected with experiments on validation sets.

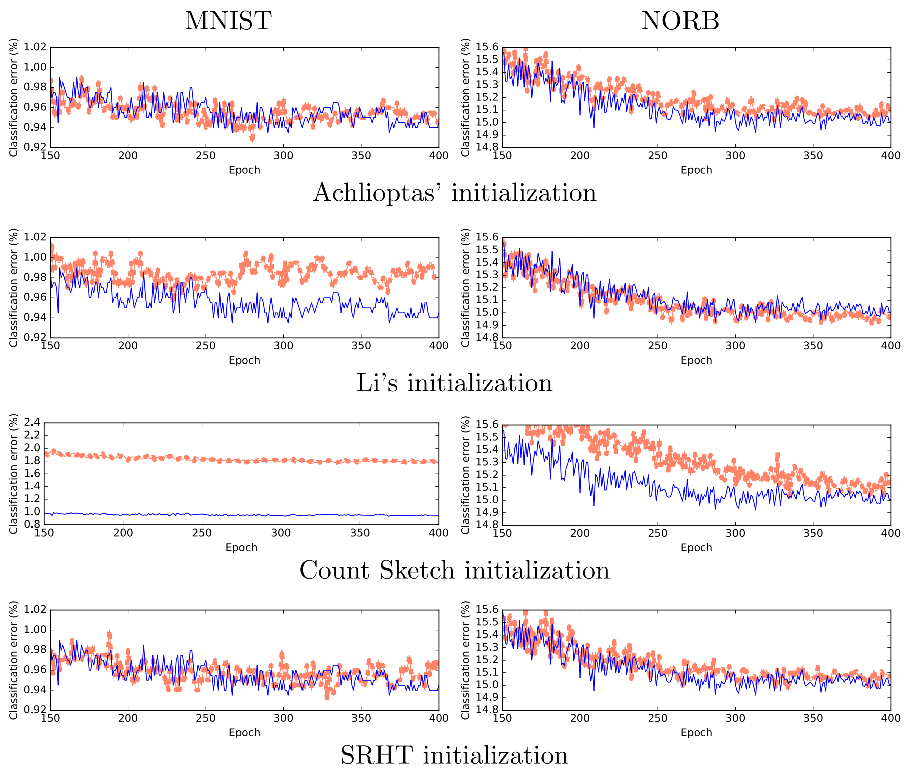

Fig. 2: Median test error on MNIST and NORB for different RP initializations (dashed orange line) and the reference Gaussian initialization (solid blue line).

For each dataset and initialization scheme we trained ten network instances with different random number seeds. Results from these experiments are reported in Fig. 2. In each case we report median test error as a function of the finetuning epoch. The standard Gaussian initialization serves as the baseline result. Unlike CNNs, pretrained networks did not benefit from RP initialization.

4 Conclusions

In this work we explored the viability of initializing rectifier networks with different random projection matrices. Our results show that RP initialization can

ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 26-28 April 2017, i6doc.com publ., ISBN 978-287587039-1. Available from http://www.i6doc.com/en/.

be a viable alternative to initialization schemes currently employed in Convolutional Neural Networks. In particular, the initialization based on Subsampled Randomized Hadamard Transform outperformed the reference He's initialization in residual networks with stochastic depth, the current state-of-the-art convolutional architecture. Sparse RP initializations, i.e., Li's, Achlioptas' and Count Sketch, yielded results that were inconsistent among different benchmarks. In pretrained networks random projection initialization yielded results usually close to the reference.

References

[1] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[2] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275, 2011.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[4] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In Grégoire Montavon, Geneviève B Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 599–619. Springer Berlin Heidelberg, 2012.

[5] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[6] Andrew Saxe, Pang W Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1089–1096, 2011.

[7] Dimitris Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281. ACM, 2001.

[8] Ping Li, Trevor J Hastie, and Kenneth W Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296. ACM, 2006.

[9] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 557–563. ACM, 2006.

[10] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse johnson-lindenstrauss transform. In *Proceedings of the forty-second annual ACM symposium on Theory of computing*, pages 341–350. ACM, 2010.

[11] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. *arXiv preprint arXiv:1603.09382*, 2016.

[12] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[13] Nitish Srivastava. Improving neural networks with dropout. Master's thesis, University of Toronto, 2013.

[14] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010.