International Journal of Engineering Sciences Paradigms and Researches (IJESPR)
Volume 47, Issue 01, Quarter 01 (January to March 2018)
An Indexed and Referred Journal with Impact Factor: 2.80
ISSN (Online): 2319-6564
www.ijesonline.com

A Stochastically Generated Deep Neural Network Using Random Graph Models

Monalisa Samal[1], Kaustav Das[2], D. K. Mohanty[3]

[1]Assistant Professor, Department of Electronics and Communication Engineering, Gandhi Institute for Technology (GIFT), Bhubaneswar
[2]Assistant Professor, Department of Electronics and Communication Engineering, Gandhi Engineering College, Bhubaneswar
[3]AssistantProfessor, School of Computer Engineering, KIIT University, Bhubaneswar

Publishing Date: 5h March, 2018

Abstract
A Deep Neural Network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. Recently, DNN has been extensively used for image recognition. The success of ResNets [12] and Dense Nets [17] is due in large part to their innovative wiring plans, however, the space of possible wirings is constrained and still driven by manual design despite being searched. In this paper, we proposed a stochastically generated DNN called Randwire with more diverse set of connectivity patterns. To do this, we first define the concept of a stochastic or random network generator that encapsulates the entire network generation process. Then, we use three different classical random graph models to generator and only wired graphs for networks. Experimental results show that the proposed networks have competitive accuracy on the ImageNet benchmark.

Keywords: *Optical communications, optical crosstalk, optical losses, photonic interconnection networks, simulation software, system analysis and design.*

Introduction

DNN can outperform many conventional machine learning methods, due to the increased amount of training data, more powerful computing resources as well as dramatically increased model parameters. Deep learning today descends from the connectionist approach to cognitive science a paradigm reflecting the hypothesis that how computational networks are wired is crucial for building intelligent ma- chines. Echoing this perspective, recent advances in computer vision have been driven by moving from models with chain-like wiring to more elaborate connectivity patterns, e.g., ResNet and DenseNet that are effective in large part because of how they are wired.

Advancing this trend, neural architecture search (NAS) has emerged as a promising direction for jointly searching wiring patterns and which operations to perform. NAS methods focus on search while implicitly relying on an important yet largely overlooked component that we call a network generator. The NAS network generator defines a family of possible wiring patterns from which networks are sampled subject to a learnable probability distribution. However, like the wiring patterns in ResNet and DenseNet, the NAS network generator is hand designed and the space of allowed wiring patterns is constrained in a small subset of all possible graphs. Given this perspective, we ask: What happens if we loosen this constraint and design novel net- work generators?

We explore this question through the lens of randomly wired neural networks that are sampled from stochastic network generators, in which a human-designed random process defines generation. To reduce bias from us the authors of this paper on the generators, we use three classical families of random graph models in graph theory. To define complete net- works, we convert a random graph into a directed acyclic graph (DAG) and apply as implemapping from nodes to heir functional roles (e.g., to the same type of convolution). The results are surprising: several variants of these random generators yield networks with competitive accuracy on

ImageNet [40]. The best generators, which use the WS model, produce multiple networks that outperform or are comparable to their fully manually designed counter- parts and the networks found by various neural architecture search methods. We also observe that the variance of ac- curacy is low for different random networks produced by the same generator, yet there can be clear accuracy gaps between different generators. These observations suggest that the network generator design is important.

We note that these randomly wired networks are not “prior free” even though they are random. Many strong priors are in fact implicitly designed into the generator, including the choice of a particular rule and distribution to control the probability of wiring or not wiring certain nodes together. Each random graph model has certain probabilistic behaviors such that sampled graphs likely exhibit certain properties (e.g., WS is highly clustered). Ultimately, the generator design determines a probabilistic distribution over networks, and *as* a result these networks tend to have certain properties. The generator design under lies the prior and thus should not be overlooked.

Our work explores a direction orthogonal to concurrent work on random search for NAS. These studies show that random search is competitive in “the NAS search space”, i.e., the “NAS network generator” in our perspective. Their results can be understood as showing that the prior induced by the NAS generator design tends to produce good models, similar to our observations. In contrast to, our work goes beyond the design of established NAS generators and explores different random generator designs.

Finally, our work suggests a new transition from designing an individual network to designing a network generatormay be possible, analogous to how our community has transitioned from designing features to designing a network that learns features. Rather than focusing primarily on search with a fixed generator, we suggest designing new network generators that produce new families of models for searching. The importance of the designed network generator (in NAS and elsewhere) also implies that machine learning has not been automated (c.f. “AutoML”) the underlying human design and prior shift from network engineering to network generator engineering.

Experiments

We conduct experiments on the ImageNet 1000-class classification task [40]. We train on the training set with ∼1.28M images and test on the 50K validation images.

Architecture Details

Our experiments span a small computation regime (e.g., MobileNet and ShuffleNet) and a regular computation regime (e.g., ResNet-50/101). Rand Wire nets in the se regimes, where N nodes and C channels determine network complexity.

We set N =32, and then set C to the nearest integer such that target model complexity is met: C=78 in the small regime, and C=109 or 154 in the regular regime.

Random Seeds

For each generator, we randomly sample 5 network instances (5 random seeds), train them from scratch, and evaluate accuracy for each instance. To emphasize that we perform no random search for each generator, we report the classification accuracy with “mean±std” for all 5 random seeds (i.e., we do not pick the best). We use the same seeds 1, . . ., 5 for all experiments.
Implementation details. We train our networks for 100 epochs, unless noted. We use a half-period-cosine shaped learning rate decay. The initial learning rate is 0.1, the weight decay is 5e-5, and the momentum is 0.9. We use label smoothing regularization with a coefficient of 0.1. Other details of the training procedure are the same.

Analysis Experiments

Random graph generators, paper compares the results of different generators in the small computation regime:

Each R and Wire net has ~580M FLOPs. It visualizes one example graph for each generator. The graph generator is specified by the random graph model (ER/BA/WS) and 20 its set of parameters: e.g., ER (0.2). We observe: 0 output degree of removed node

40 40

20 20

0 0

All random generators provide decent accuracy over all

1 5 9 13 17 21 25

1 2 3 5 7

1 3 5 7 9 11

5 random network instances; none of them fails to converge. ER, BA, and WS all have certain settings that yield mean accuracy of >73%, within a <1% gap from the best mean accuracy of 73.8% from WS (4,0.75).

Moreover, the variation among the random network in- stances is low. Almost all random generators input degree of removed edge's target node. We randomly remove one node (top) or remove one edge (bottom) from a graph after the network is trained, and evaluate the loss (Δ) in accuracy on ImageNet. From left to right are ER, BA, and WS generators. Red circle: mean; gray bar: median; orange box: interquartile range; blue dot: an individual damaged instance.

Have and standard deviation (std) of 0.2~0.4%. As a comparison, training the same instance of a ResNet-50 multiple times has a typical std of 0.1~0.2% [11]. The observed low variance of our random generators suggests that even with- out random search (i.e., picking the best from several random instances), it is likely that the accuracy of a network instance is close to the mean accuracy, subject to some noise.

On the other hand, different random generators may have a gap between the mean accuracies, e.g., BA(1) has 70.7% accuracyandis~3%lowerthanWS(4,0.75).This suggests that random generator design, including the wiring priors (BAvs.WS) and generation parameters, plays an important role in the accuracy of sampled network instances.

Figure 3 also includes a set of non-random generators: WS (K, P =0). “P =0” means no random rewiring. Interestingly, the results of WS(K,P=0) are all worse than their WS(K, P>0) counterparts for any fixed K in Figure3.

Graph damage. We explore graph damage by randomly removing on enodeoredgeanablative setting inspired by. Formally, given a network instance after training, were move one node or one edge from the graph and evaluate the validation accuracy without any further training.

When a node is removed, we evaluate the accuracy loss (Δ) vs. the output degree of that node (Figure 5, top). It is clear that ER, BA, and WS behave differently under such damage. For networks generated by WS, the mean degradation of accuracy is larger when the output degree of the removed node is higher. This implies that “hub” nodes in WS that send information to many nodes are influential.

When an edge is removed, we evaluate the accuracy loss vs. the input degree of this edge’s target node (bottom). If the input degree of an edge’s target node is smaller, removing this edge tends to change a larger portion of the target node’s inputs. This trend can be seen by the fact that the accuracy loss is generally decreasing along the x-axis in (bottom). The ER model is less sensitive to edge removal, possibly because in ER’s definition wiring of every edge is independent.

Conclusion

We explored stochastically wired neural networks driven by three classical random graph models from graph theory. The results were surprising: the mean accuracy of these models is competitive with hand-designed and optimized models from recent work on neural architecture search. Our exploration was enabled by the novel concept of a network generator. We hope that future work exploring new generator designs may yield new, powerful networks designs.

References

[1] Albert and Albert La. Statistical mechanics of complex networks. Reviews of modern physics, 74(1):47, 2002. 1, 2, 4,5

[2] Danielle Smith Bassett and ED Bullmore. Small-world brain networks. The neuroscientist, 12(6):512–523, 2006.2

[3] Danielle S Bassett and Olaf Sporns. Nature neuroscience, 20(3):353, 2017. 2

[4] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. Nature reviews neuroscience, 10(3):186, 2009.2

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target ask and hardware. ICLR, 2019.2

[6] Franc¸ ois Chollet. Xception: Deep learning with depth wise separable convolutions. In CVPR, 2017.4

[7] PaulErdo˝s and Alfre´dRe´nyi. On the evolution of random graphs.Publ.Math.Inst.Hung.Acad.Sci,5(1):17–60,1960. 1, 2, 4,5

[8] Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. Cognition, 28(1- 2):3–71, 1988.1

[9] Edgar Nelson Gilbert. Random graphs. The Annals of Math- ematical Statistics, 30(4):1141–1144, 12 1959.5

[10] Ross Girshick, IlijaRadosavovic, Georgia Gkioxari,PiotrDolla´r,andKaimingHe.Detectron,2018.8

[11] PriyaGoyal,PiotrDolla´r, RossGirshick, PieterNoord-huis, Lukasz Wesolowski, AapoKyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1hour.arXiv:1706.02677, 2017. 6, 7,8

[12] Kaiming He, Xiangyu Zhang, ShaoqingRen, and JianSun. Deep residual learning for image recognition. In CVPR, 2016. 1, 2, 3, 5, 6,8

[13] Kaiming He, Xiangyu Zhang, ShaoqingRen, and JianSun. Identity mappings in deep residual networks. In ECCV, 2016.4

[14] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012.8

[15] Sepp Hochreiter and Ju¨rgen Schmidhuber. Long short termmemory. Neural computation, 1997.2

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco An- dreetto, and Hartwig Adam. MobileNets: Efficient con- volutional neural networks for mobile vision applications. arXiv:1704.04861, 2017. 6,8

[17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kil- ian Q Weinberger. Densely connected convolutional networks. In CVPR, 2017. 1, 2, 3,6

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.4

[19] Manfred Kochen. The Small world. Ablex Pub., 1989. 2,5

[20] Alex Krizhevsky, IlyaSutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks.
In NIPS, 2012.1, 5