EnergyNet: Energy-based Adaptive Structural Learning of Artificial Neural Network Architectures

Gus Kristiansen*  
Google Research  
New York, NY 10011  
gusatb@google.com

Xavi Gonzalvo  
Google Research  
New York, NY 10011  
xavigonzalvo@google.com

Abstract

We present EnergyNet, a new framework for analyzing and building artificial neural network architectures. Our approach adaptively learns the structure of the networks in an unsupervised manner. The methodology is based upon the theoretical guarantees of the energy function of restricted Boltzmann machines (RBM) of infinite number of nodes. We present experimental results to show that the final network adapts to the complexity of a given problem.

1 Introduction

Despite the huge success of deep learning during the last years as a powerful framework, choosing the right architecture still remains a practical problem. This presents several problems. On the one hand, from the point of view of the optimization problem there is no guarantee of the optimality of the learning objective [Cortes et al., 2016b]. On the other hand, large-scale hyperparameter tuning requires large amounts of data and resources (e.g. random search [Bergstra et al., 2011]).

In this paper, we attempt to alleviate the problem of choosing the right network architecture. We interpret the general structure of a neural network (NN) as an effective generative model for unsupervised learning. We then introduce EnergyNet as a framework for adapting the structure and complexity of the network to the difficulty of the particular problem at hand with no pre-defined architecture. The adaptation process is divided into two stages. Firstly, the architecture of the neural network is predicted. Starting from a simple single layer neural network, we add more neurons as required. Once the architecture of the network is estimated it works as a regular feedforward NN by adding a supervised output layer and training with gradient descent.

Automatic design of NN architectures has been studied before. Fahlman and Lebiere [1990] presented a system with a minimal network, then automatically trained and added new hidden units. Genetic algorithms have also been proposed [Vonk et al., 1995, Rolls and Stringer, 2000, Arifovic and Gencay, 2001, Saemi et al., 2007, Stanley et al., 2003, Stanley and Miikkulainen, 2002, Miikkulainen et al., 2017]. Additionally, smaller networks has been used to train larger networks [Chen et al., 2015, Ha et al., 2016]. Finally, reinforcement learning has also been used for searching different architectures [Zoph and Le, 2016].

Recently Cortes et al. [2016a] proposed the ADANet framework that adaptively learns NN architectures. The major difference of our work with that framework is as follows. Firstly, in our case the architecture of the network has no output layer connections. Secondly, the architecture of the network is now independent of the learned weights.

This paper is organized as follows. Section 2 presents the theory of the framework. We describe the model complexity to balance the size of the networks and the theoretical properties of the iRBMs.

*Work done as a member of the Google Brain Residency program [g.co/brainresidency]

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
Section 2.4 describes the properties of the DBNs. Section 3 describes the algorithm and Section 4 concludes with experimental results.

2 Preliminaries and Theory

Let $\mathcal{X}$ denote the input space. We assume that training and test points are drawn i.i.d. according to some distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$. Given any $x \in \mathcal{X}$, we denote by $\Phi(x) \in \mathbb{R}^{n_0}$, the feature representation of $x$.

2.1 Artificial Neural Networks

The standard description of a modern feedforward network is a network of layers of nodes, where each layer is mapped to the layer above it via a linear mapping composed with a component-wise nonlinear transformation. To make this description precise, we define a neural network following [Cortes et al., 2016b]. Let $l$ denote the number of layers in a network. For each $k \in [l]$, denote by $n_k$ the maximum number of nodes in layer $k$.

Let $1 \leq p \leq \infty$ and $k \geq 1$. Then define the set $\mathcal{H}_k^{(p)}$ to be the family of functions at layer $k$ of the network in the following way:

$$\mathcal{H}_k^{(p)} = \left\{ x \mapsto \sum_{j=1}^{n_k-1} u_j (\varphi_{k-1} \circ h_j)(x) : u \in \mathbb{R}^{n_k-1}, \|u\|_p \leq \Lambda_k, h_j \in \mathcal{H}_k^{(p)} \right\}, \quad (\forall k > 1) \quad (1)$$

where $\mathcal{H}_k^{(p)}$ refers to the case of input features, $\Lambda_k > 0$ is a hyperparameter and where $\varphi_k$ is an activation function (e.g. Rectified Linear Unit (ReLU), see [Goodfellow et al., 2016] for more). The choice of norm $p$ here is left to the learner and will determine both the sparsity of the network and the accompanying learning guarantee of the resulting model.

Since neural networks are built as compositions of layers, it is natural from the theoretical standpoint to first analyze the complexity of any layer in terms of the complexity of its previous layer. Results for ADANET [Cortes et al., 2016b] demonstrate that this can indeed be done, and that the empirical Rademacher complexity of any intermediate layer $k$ in the network is bounded by the empirical Rademacher complexity of its input times a term that depends on a power of the size of the layer:

**Definition 1** (Complexity of a neural network). Let $r_\infty = \max_{j \in [1,n_1], i \in [1,m]} |\Phi(x_i)|$, and $\frac{1}{p} + \frac{1}{q} = 1$. Then for any $k \geq 1$, the empirical Rademacher complexity of $\mathcal{H}_k^{(p)}$ for a sample $S$ of size $m$ can be upper bounded as follows:

$$\hat{R}_S(\mathcal{H}_k^{(p)}) \leq 2^{k-1} r_\infty \left( \prod_{j=1}^{k} \Lambda_{n_{j-1}^{1/p}} \right) \sqrt{\frac{2 \log(2n_0)}{m}}. \quad (2)$$

2.2 Restricted Boltzmann Machines (RBM)

The RBM is a two layer Markov Random Field, where the observed binary stochastic visible units $v \in \{0, 1\}^D$ have pairwise connections to the binary stochastic hidden units $h \in \{0, 1\}^K$. There are no pairwise connections within the visible units, nor within the hidden ones [Hinton, 2002].

In an RBM model, each configuration $(v,h) \in \mathcal{V} \times \mathcal{H}$ has an associated energy value defined by the following function:

**Definition 2** (RBM energy function). Restricted Boltzmann machine is an energy-based model, in which we define the energy for a state $(v, h)$ as:

$$E(v, h; \theta) = -v^T b^v - h^T b^h - v^T W h,$$  \quad (3)

where $\theta = \{W, b^v, b^h\}$ are the parameters of the model.

**Definition 3** (Probability distribution of an RBM). For $m$ data samples, the probability distribution of the RBM of a visible vector is obtained by marginalizing over all configurations of hidden vectors:

$$P_\theta(v) = \frac{1}{Z} \sum_{h' \in \mathcal{H}} \exp(-E(v, h'; \theta))$$
where $h_i$ are the hidden nodes of the $i$-th layer, $Z$ is the partition function used for normalization:

$$Z = \sum_{v' \in V} \sum_{h' \in H} \exp(-E(v, h; \theta))$$

### 2.3 Infinite Restricted Boltzmann Machines (iRBM)

iRBM can be implemented using a random variable [Côté and Larochelle, 2015] or via Frank-Wolfe optimization [Ping et al., 2016]. We follow the former approach where the order of a hidden unit is taken into account by introducing a random variable $z$ that can be seen as the effective number of hidden units participating to the energy. Hidden units are selected starting from the left and the selection of each hidden unit is associated with an incremental cost in energy.

The model via the latent variable $z$ allows for different observations having been generated by a different number of hidden units. Specifically, for a given $v$, the conditional distribution over the corresponding value of $z$ is:

$$P(z|v) = \frac{N(z)}{\sum_{z'=1}^{K=\infty} N(z')} = \frac{N(z)}{Z(v)}$$

where $N(z) = \exp(-F(v, z))$ and $K$ is the maximum number of hidden nodes, which in this case is infinite. The free energy, $F()$ is defined as:

$$F(v, z) = -v^T b_v - \sum_{i=0}^z \left( \text{soft}_+ (W_i v + b_i^h) - \beta_i \right)$$

where $\beta_i$ is an energy penalty for selecting each $i$-th hidden unit.

The denominator (i.e. the normalization factor) needs to be defined for the infinite $K$. In this case $Z(v)$ must be divided into two parts, before the maximum number of nodes at a particular time step, $n$, and the rest:

$$Z(v) = \sum_{z=1}^{n} N(z) + \sum_{z=n+1}^{\infty} N(z) = \sum_{z=1}^{n} N(z) + \frac{a}{a-1} N(z)$$

where $a$ is the result of a geometric series of the infinite term [Côté and Larochelle, 2015].

### 2.4 Deep Belief Networks (DBN)

There are two possible ways of putting together multiple RBMs: Deep Boltzmann Machines (DBM) [Salakhutdinov and Hinton, 2009] and Deep Belief Networks (DBN) [Hinton et al., 2006]. Both are probabilistic graphical models consisting of stacked layers of RBMs. The difference is in how these layers are connected [Salakhutdinov and Hinton, 2009]. In a DBN the connections between layers are directed. Therefore, the first two layers form an RBM (an undirected graphical model), then the subsequent layers form a directed generative model. In a DBM, the connection between all layers is undirected, thus each pair of layers forms an RBM. If multiple layers are learned in a greedy, layer-by-layer way, the resulting composite model is a DBN [Bengio et al., 2007].

As described in Section 3 the algorithm proposed in this paper requires the use of the log-likelihood of a DBN. Due to the presence of the partition function, exact maximum likelihood learning in RBMs and DBNs are intractable. In [Salakhutdinov and Murray, 2008, Neal, 1998] an Annealed Importance Sampling (AIS) is used to efficiently estimate the partition function of an RBM and to estimate a lower bound on the log-probability of a DBN. We opted for a more efficient version presented by [Theis et al., 2011]. This still requires an estimation of the partition function for the top RBM but uses samples drawn from the densities of each layer in a feed-forward manner.

### 3 Algorithm

Our System forms a DBN by stacking multiple iRBMs trained layer by layer. Since an iRBM is an extension of an RBM, the training of a DBN remains unchanged. After each layer is trained...
using the contrastive divergence algorithm applying the specifics of iRBMs (see Section 2.3) it is converted back into a regular RBM in order to compute the log-likelihood.

The minimum description length (MDL) criterion (see Rissanen, 1984) formalizes the idea that for a set of hypotheses \( H \) and data set \( X \), we should try to find the hypothesis or a combination of hypotheses in \( H \) that compresses \( X \) most. The idea is applied successfully to the problems of model selection and overfitting (Grünwald, 2005).

For a sequence of \( m \) data points \( X \) we want to find a model defined by a set of parameters \( \theta = [\theta_1, \ldots, \theta_L] \) that can efficiently maximize \( P_{\theta}(x) \) for the data \( x \). The description length \( M_j(x) \) for data \( x \) of an underlying model \( j \) is given by,

\[
M_j(x) = -\log \left( P_{\hat{\theta}^{(j)}}(x) \right) + \ell(\hat{\theta}^{(j)})
\]

where \( -\log \left( P_{\hat{\theta}^{(j)}}(x) \right) \) represents the maximum likelihood estimate of model \( j \) which is identical to the negative of the log-likelihood (Section 2.4); and \( \ell(\hat{\theta}^{(j)}) \) is the complexity of the model. Since the ultimate goal is to treat the DBN as a regular DNN, the complexity of the regular DNN is defined as the Rademacher complexity of Equation (2).

Appendix A describes in detail the general algorithm. In summary, the algorithm assumes there is a maximum number of layers \( T \). For every time step \( t \) for a network \( U_{t+1} \) we will train a new candidate layer \( H_{t+1} \). In order for the candidate layer to become part of the new network \( U_{t+1} \) it must maximize the log-likelihood taking into account Equation (7).

4 Experimental results

In this section we present the performance of architectures generated by ENERGYNET on a number of datasets: mnist, cifar-10, german and diabetes. Note that convolutional neural networks are often a more natural choice for image classification problems such as cifar-10 (60,000 images evenly categorized in 10 different classes [Krizhevsky, 2009]. However, the goal of our experiments was a proof-of-concept showing that the structural learning approach we propose is very competitive with traditional approaches for finding efficient architectures.

| Dataset | Best architecture | Accuracy |
|---------|-------------------|----------|
|         | DNN | EnergyNet | DNN | EnergyNet |
| mnist   | 2048, 512, 1024 | 1525, 239 | 0.9862 ± 0.0003 | 0.9873 ± 0.0002 |
| cifar-10| 256, 1024         | 101, 257 | 0.6024 ± 0.0031 | 0.5856 ± 0.0011 |
| german  | 16, 32            | 122      | 0.8677 ± 0.010  | 0.8851 ± 0.0079 |
| diabetes| 32, 16            | 184      | 0.8433 ± 0.011  | 0.8593 ± 0.0078 |

In Table 1 we compare ENERGYNET to a regular framework for defining DNN. The regular framework uses the following hyperparameter ranges: learning rate in [0.0001, 0.5], batch size in [16, 64, 128] and ReLu activations. The search for hidden layer architectures is performed using a combination of 30 networks with a predefined number of nodes using a Gaussian process bandits algorithm [Snoek et al., 2012].

The ENERGYNET framework fixes the architecture with a maximum number of layers of 10, dual norm of 2, a constant \( \beta = 1.01 \) and the same learning rate of 0.01 for every dataset. Appendix A describes how the growth in every layer is controlled via \( \Gamma \). This is fixed to \( \Gamma = 0.1 \) for all datasets but mnist which is set to \( \Gamma = 0.8 \). Once the architecture is fixed, a DNN is trained and tuned using the same range of hyperparameters as in the regular framework.

5 Conclusions

We presented ENERGYNET: a new framework for unsupervisedly analyzing the architecture of artificial neural networks. Our method optimizes for reconstruction performance, and it explicitly and automatically addresses the trade-off between network architecture and data modeling. We presented
experimental results showing that ENERGYNET can efficiently learn DNN architectures that achieve comparable results with baseline methods. ENERGYNET generates DNNs with fewer number of parameters. This is caused by the greedy construction of the model layer by layer with finer granularity in the total number of nodes in each layer. Our framework is independent of the type of input features so it can be used for other purposes as well (e.g. as a weak learner in ADANET [Cortes et al., 2016a]).

References

J. Arifovic and R. Gençay. Using genetic algorithms to select architecture of a feedforward artificial neural network. Physica A: Statistical Mechanics and its Applications, 289(3):574 – 594, 2001. ISSN 0378-4371. doi: https://doi.org/10.1016/S0378-4371(00)00479-9. URL http://www.sciencedirect.com/science/article/pii/S0378437100004799

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-Wise Training of Deep Networks. pages 153–160, 2007. URL http://www.iro.umontreal.ca/~lisa/pointeurs/BengioNips2006All.pdf

J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In NIPS, pages 2546–2554, 2011.

T. Chen, I. J. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. CoRR, abs/1511.05641, 2015. URL http://arxiv.org/abs/1511.05641

C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang. AdaNet: Adaptive Structural Learning of Artificial Neural Networks. In Proceedings of Adaptive and Scalable Nonparametric Methods in Machine Learning, 2016a.

C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang. AdaNet: Adaptive Structural Learning of Artificial Neural Networks. In Proceedings of The 1st International Workshop on Efficient Methods for Deep Neural Networks, 2016b.

M. Côté and H. Larochelle. An infinite restricted boltzmann machine. CoRR, abs/1502.02476, 2015. URL http://arxiv.org/abs/1502.02476

S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, Advances in Neural Information Processing Systems 2, pages 524–532. Morgan-Kaufmann, 1990.

I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.

P. Grünwald. A tutorial introduction to the minimum description length principle. In Advances in Minimum Description Length: Theory and Applications. MIT Press, 2005.

D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. CoRR, abs/1609.09106, 2016. URL http://arxiv.org/abs/1609.09106

G. E. Hinton. Training products of experts by minimizing contrastive divergence. Neural Computation, 14(8):1771–1800, 2002.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. Neural Comput., 18(7):1527–1554, July 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL http://dx.doi.org/10.1162/neco.2006.18.7.1527

A. Krizhevsky. Learning multiple layers of features from tiny images. Master’s thesis, University of Toronto, 2009.

R. Miikkulainen, J. Z. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving deep neural networks. CoRR, abs/1703.00548, 2017. URL http://arxiv.org/abs/1703.00548

R. M. Neal. Annealed Importance Sampling. ArXiv Physics e-prints, Mar. 1998.
A Algorithm

In the following section we present the details of the algorithm introduced in Section 3.

The algorithm runs for $T$ time steps in order to create a maximum of $T$ layers (see Algorithm 1, function BUILD_NETWORK). At time $t$ there exist a neural network $U_t$ (see Figure 1). A candidate layer $h_{t+1}$ is trained as a new iRBM model. This iRBM is designed to have an infinite number of nodes. This requires that a finite number of neurons $n$ is selected (see TRAIN NEW LAYER).

The description length of a fitted model $M_t$ is the sum of two parts. The first part of the description length represents the fit of the model to the data; as the model fits better, this term shrinks. The second part of the description length represents the complexity of the model. This part encodes the parameters of the model itself; it grows as the model becomes more complex.
Algorithm 1 Constructs a network where $\gamma$ is the average growth of a layer on each training step and $\Gamma$ is a threshold. A layer stops growing when the number of nodes converges.

function $\text{TRAINNEWLAYER}(v)$

\[
n \leftarrow 0
\]
\[
c \leftarrow 0
\]
while $\gamma > \Gamma$ do
\[
\text{if ADDNODE() then}
\]
\[
n \leftarrow n + 1
\]
end if
\[
\gamma \leftarrow n/c
\]
\[
c \leftarrow c + 1
\]
end while

end function

function $\text{BUILDNETWORK}$

\[
U \leftarrow \{\}
\]
for $t < T$ do
\[
\text{if ADDNODE() then}
\]
\[
n \leftarrow n + 1
\]
end if
\[
\gamma \leftarrow n/c
\]
\[
c \leftarrow c + 1
\]
end while

end function

Figure 1: Network construction at time $t$. Output connections are marked with dotted lines because they do not exist at construction time.