

Efficient Deep Feature Learning and Extraction via StochasticNets

Mohammad Javad Shafiee[†], Parthipan Siva[‡], Paul Fieguth[†], Alexander Wong[†]

[†]VIP Research Group, University of Waterloo, Waterloo, ON, Canada

[‡]Aimetis Corporation, Waterloo, ON, Canada

{mjshafiee, pfieguth, alexander.wong}@uwaterloo.ca

[‡]parthipan.siva@aimetis.com

Abstract

Deep neural networks are a powerful tool for feature learning and extraction. One area worth exploring in feature extraction using deep neural networks is efficient neural connectivity formation for faster feature learning and extraction. Motivated by findings of stochastic synaptic connectivity formation in the brain as well as the brain's uncanny ability to efficiently represent information, we propose the efficient learning and extraction of features via StochasticNets, where sparsely-connected deep neural networks can be formed via stochastic connectivity between neurons. Experimental results show that features learned using deep convolutional StochasticNets, with fewer neural connections than conventional deep convolutional neural networks, can allow for better or comparable classification accuracy than conventional deep neural networks. Finally, it was also shown that significant gains in feature extraction speed can be achieved in embedded applications using StochasticNets.

1. Introduction

Deep neural networks are a powerful tool for feature learning and extraction given their ability to represent and model high-level abstractions in highly complex data. Deep neural networks have shown considerable capabilities in producing features that enable state-of-the-art performance for handling complex tasks such as speech recognition [6, 13], object recognition [14, 21, 23, 26], and natural language processing [1, 5]. Recent advances in improving the performance of deep neural networks for feature learning and extraction have focused on areas such as network regularization [30, 31], activation functions [10, 11, 15], and deeper architectures [26, 28, 32], where the goal is to learn more representative features with respect to increasing task accuracy.

Despite the power capabilities of deep neural networks for feature learning and extraction, they are very rarely employed on embedded devices such as video surveillance

cameras, smartphones, and wearable devices. This difficult migration of deep neural networks into embedded applications for feature extraction stems largely from the fact that, unlike the highly powerful distributed computing systems and GPUs that are often leveraged for deep learning networks, the low-power CPUs commonly used in embedded systems simply do not have the computational power to make deep neural networks a feasible solution for feature extraction.

Much of the focus on migrating deep neural networks for feature learning and extraction in embedded systems have been to design custom embedded processing units dedicated to accelerating deep neural networks [8, 18, 12]. However, such an approach greatly limits the flexibility of the type of deep neural network architectures that can be used. Furthermore, such an approach requires additional hardware, which adds to the cost and complexity of the embedded system. On the other hand, improving the efficiency of deep neural networks for feature learning and extraction is much less explored, with considerably fewer strategies proposed so far [3]. In particular, very little exploration has been conducted on efficient neural connectivity formation for efficient feature learning and extraction, which can hold considerable promise in achieving highly efficient deep neural network architectures that can be used in embedded applications.

One way to address this challenge is to draw inspiration from the brain which has an uncanny ability to efficiently represent information. In particular, we are inspired by the brain's procedure where develops synaptic connectivity between neurons. Recently, in a pivotal paper [16], data of living brain tissue from Wistar rats was collected and used to construct a partial map of a rat brain. Based on this map, Hill et al. came to a very surprising conclusion. The synaptic formation, of specific functional connectivity in neocortical neural microcircuits, was found to be stochastic in nature. This is in sharp contrast to the way deep neural networks are formed, where connectivity is largely deterministic and pre-defined.

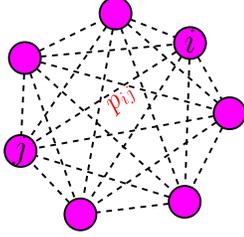


Figure 1. An illustrative example of a random graph. All possible edge connectivity between the nodes in the graph may occur independently with a probability of p_{ij} .

Motivated by findings of random neural connectivity formation and the efficient information representation capabilities of the brain, we proposed the learning of efficient feature representations via StochasticNets [25], where the key idea is to leverage random graph theory [7, 9] to form sparsely-connected deep neural networks via stochastic connectivity between neurons. The connection sparsity, in particular for deep convolutional networks, allows for more efficient feature learning and extraction due to the sparse nature of receptive fields which require less computation and memory access. We will show that these sparsely-connected deep neural networks, while computationally efficient, can still maintain the same accuracies as traditional deep neural networks. Furthermore, the StochasticNet architecture for feature learning and extraction presented in this work can also benefit from all of the same approaches used for traditional deep neural networks such as data augmentation and stochastic pooling to further improve performance.

The paper is organized as follows. First, a review of random graph theory is presented in Section 2. The theory and design considerations behind forming StochasticNet as a random graph realizations are discussed in Section 3. Experimental results where we train deep convolutional StochasticNets to learn abstract features using the CIFAR-10 dataset [20], and extract the learned features from images to perform classification on the SVHN [24] and STL-10 [4] datasets is presented in Section 5. Finally, conclusions are drawn in Section 6.

2. Review of Random Graph Theory

The underlying idea of deep feature learning via StochasticNets is to leverage random graph theory [7, 9] to form the neural connectivity of deep neural networks in a stochastic manner such that the resulting neural networks are sparsely connected yet maintaining feature representation capabilities. As such, it is important to first provide a general overview of random graph theory for context. In random graph theory, a random graph can be defined as the probability distribution over graphs [2]. A number of different random graph models have been proposed in literature.

A commonly studied random graph model is that pro-

posed by [9], in which a random graph can be expressed by $\mathcal{G}(n, p)$, where all possible edge connectivity are said to occur independently with a probability of p , where $0 < p < 1$. This random graph model was generalized by [19], in which a random graph can be expressed by $\mathcal{G}(\mathcal{V}, p_{ij})$, where \mathcal{V} is a set of vertices and the edge connectivity between two vertices $\{i, j\}$ in the graph is said to occur with a probability of p_{ij} , where $p_{ij} \in [0, 1]$. Therefore, based on this generalized random graph model, realizations of random graphs can be obtained by starting with a set of n vertices $\mathcal{V} = \{v_q | 1 \leq q \leq n\}$ and randomly adding a set of edges between the vertices based on the set of possible edges $\mathcal{E} = \{e_{ij} | 1 \leq i \leq n, 1 \leq j \leq n, i \neq j\}$ independently with a probability of p_{ij} . A number of realizations of a random graph are provided in Figure 2 for illustrative purposes. It is worth noting that because of the underlying probability distribution, the generated realizations of the random graph often exhibit differing edge connectivity.

Given that deep neural networks can be fundamentally expressed and represented as graphs \mathcal{G} , where the neurons are vertices \mathcal{V} and the neural connections are edges \mathcal{E} , one intriguing idea for introducing stochastic connectivity for the formation of deep neural networks is to treat the formation of deep neural networks as particular realizations of random graphs, which we will describe in greater detail in the next section.

3. StochasticNets: Deep Neural Networks as Random Graph Realizations

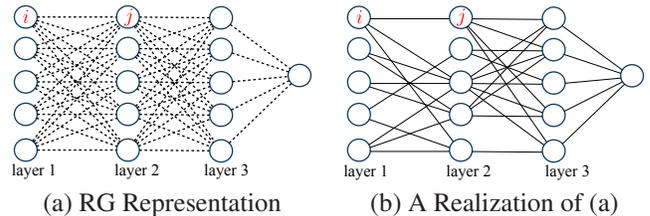


Figure 3. Example random graph representing a section of a deep feed-forward neural network (a) and its realization (b). Every neuron i may be connected to neuron j with probability $p(i \rightarrow j)$ based on random graph theory. To enforce the properties of a general deep feed-forward neural network, there is no neural connectivity between nodes that are not in the adjacent layers. As shown in (b), the neural connectivity of nodes in the realization are varied since they are drawn based on a probability distribution.

Let us represent a deep neural network as a random graph $\mathcal{G}(\mathcal{V}, p(i \rightarrow j))$, where \mathcal{V} is the set of neurons $\mathcal{V} = \{v_i | 1 \leq i \leq n\}$, with v_i denoting the i^{th} neuron and n denoting the total number of neurons, in the deep neural network and $p(i \rightarrow j)$ is the probability that a neural connection occurs between neurons v_i and v_j . It is worth noting that since neural networks are directed graphs, to this end the probability $p(\cdot)$ is represented in a directed way from

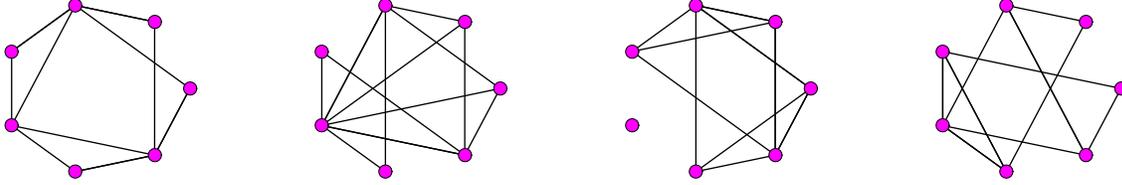


Figure 2. Realizations of a random graph with 7 nodes. The probability for edge connectivity between all nodes in the graph was set to $p_{i,j} = 0.5$ for all nodes i and j . Each diagram demonstrates a different realization of the random graph.

source node to the destination node. As such, one can then form a deep neural network as a realization of the random graph $\mathcal{G}(\mathcal{V}, p(i \rightarrow j))$ by starting with a set of neurons \mathcal{V} , and randomly adding neural connections between the set of neurons independently with a probability of $p(i \rightarrow j)$ as defined above.

While one can form practically any type of deep neural network as a realization of random graphs, an important design consideration for forming deep neural networks as random graph realizations is that different types of deep neural networks have fundamental properties in their network architecture that must be taken into account and preserved in the random graph realization. Therefore, to ensure that fundamental properties of the network architecture of a certain type of deep neural network is preserved, the probability $p(i \rightarrow j)$ must be designed in such a way that these properties are enforced appropriately in the resultant random graph realization. Let us consider a general deep feed-forward neural network. First, in a deep feed-forward neural network, there can be no neural connections between non-adjacent layers. Second, in a deep feed-forward neural network, there can be no neural connections between neurons on the same layer. Therefore, to enforce these two properties, $p(i \rightarrow j) = 0$ when $l(i) \neq l(j) + 1$ where $l(i)$ encodes the layer number associated to the node i . An example random graph based on this random graph model for representing general deep feed-forward neural networks is shown in Figure 3(a), with an example feed-forward neural network graph realization is shown in Figure 3(b). Note that the neural connectivity for each neuron may be different due to the stochastic nature of neural connection formation.

Deep convolutional neural networks (CNNs) are popular network structures which are mostly used for computer vision applications. Here, the experiments are evaluated based on CNNs. The neural connectivity formation for the deep convolutional StochasticNet realizations is achieved via acceptance-rejection sampling, and can be expressed by:

$$e_{i \rightarrow j} \text{ exists where } \left[p(i \rightarrow j) \geq T \right] = 1 \quad (1)$$

where $e_{i \rightarrow j}$ is the neural connectivity from node i to node j , $[\cdot]$ is the Iverson bracket [17], and T encodes the sparsity of neural connectivity in the StochasticNet. While it is

possible to take advantage of any arbitrary distribution to construct deep convolutional StochasticNet realizations, for the purpose of this paper two different spatial neural connectivity models were explored for the convolutional layers: i) uniform connectivity model:

$$p(i \rightarrow j) = \begin{cases} U(0, 1) & j \in R_i \\ 0 & \text{otherwise.} \end{cases}$$

and ii) a Gaussian connectivity model:

$$p(i \rightarrow j) = \begin{cases} \mathcal{N}(i, \sigma) & j \in R_i \\ 0 & \text{otherwise.} \end{cases}$$

where the mean is at the center of the receptive field (i.e., i) and the standard deviation σ is set to be a third of the receptive field size. In this study, R_i is defined as a 5×5 spatial region around node i in convolutional layers and it has the size of next fully connected in the fully connected hidden layers. The neural connectivity of 100% is equivalent to a dense 5×5 receptive field used for ConvNets.

4. Feature Learning via Deep Convolutional StochasticNets

As one of the most commonly used types of deep neural networks for feature learning is deep convolutional neural networks [22], let us investigate the efficacy of efficient feature learning can be achieved via deep convolutional StochasticNets. Deep convolutional neural networks can provide a general abstraction of the input data by applying the sequential convolutional layers to the input data (e.g. input image). The goal of convolutional layers in a deep neural network is to extract discriminative features to feed into the classifier such that the fully connected layers play the role of classification in deep neural networks. Therefore, the combination of receptive fields in convolutional layers can be considered as the feature extractor in these models. The receptive fields' parameters must be trained to find optimal parameters leading to most discriminative features.

However learning those parameters is not possible every time due to the computational complexity or lack of enough training data such that general receptive fields (i.e., convolutional layers) without learning is desirable. On the other

hand, the computational complexity of extracting features is another concern which should be addressed. Essentially, extracting features in a deep convolutional neural network is a sequence of convolutional processes which can be represented as multiplications and summations and the number of operations is dependent on the number of parameters of receptive fields. Motivated by those reasons, sparsifying the receptive fields while maintaining the generality of them is highly desirable.

To this end, we want to sparsify the receptive field motivated by the StochasticNet framework to provide efficient deep feature learning. As explained in previous section, in addition to the design considerations for $p(i \rightarrow j)$ presented in the previous section to enforce the properties of deep feed-forward neural networks, additional considerations must be taken to preserve the properties of deep convolutional neural networks, which is a type of deep feed-forward neural network.

Specifically, the neural connectivity for each randomly realized receptive field K (i.e., K is the stochastic realization of R in a convolutional layer) in the deep convolutional StochasticNet is based on a probability distribution, with the neural connectivity configuration thus being shared amongst different small neural collections for a given randomly realized receptive field. An example of a realized deep convolutional StochasticNet is shown in Figure 4. As seen, the neural connectivity for randomly realized receptive field $K_{l,1}$ is completely different from randomly realized receptive field $K_{l,2}$. The response of each randomly realized receptive field leads to an output in new channel in layer $l + 1$.

A realized deep convolutional StochasticNet can then be trained to learn efficient feature representations via supervised learning using labeled data. One can then use the trained StochasticNet for extracting a set of abstract features from input data.

4.1. Relationship to Other Methods

While a number of stochastic strategies for improving deep neural network performance have been previously introduced [30], [3] and [27], it is very important to note that the proposed StochasticNets is fundamentally different from these existing stochastic strategies in that StochasticNets main significant contributions deals primarily with the formation of neural connectivity of individual neurons to construct efficient deep neural networks that are inherently sparse prior to training, while previous stochastic strategies deal with either the grouping of existing neural connections to explicitly enforce sparsity [3], or removal/introduction of neural connectivity for regularization during training. More specifically, StochasticNets is a realization of a random graph formed prior to training and as such the connectivity in the network are inherently sparse, and are permanent and

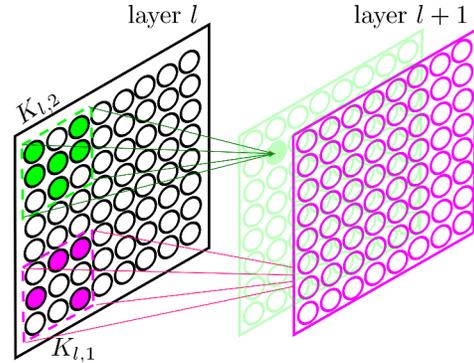


Figure 4. Forming a deep convolutional StochasticNet. The neural connectivity for each randomly realized receptive field (in this case, $\{K_{l,1}, K_{l,2}\}$) are determined based on a probability distribution, and as such the configuration of each randomly realized receptive field may differ. It can be seen that the shape and neural connectivity for receptive field $K_{l,1}$ is completely different from receptive field $K_{l,2}$. The response of each randomly realized receptive field in layer l leads to an output in layer $l + 1$. Only one layer of the formed deep convolutional StochasticNet is shown for illustrative purposes.

do not change during training. This is very different from Dropout [27] and DropConnect [30] where the activations and connections are temporarily removed during training and put back during test for regularization purposes only, and as such the resulting neural connectivity of the network remains dense. There is no notion of dropping in StochasticNets as only a subset of possible neural connections are formed in the first place prior to training, and the resulting network connectivity of the network is sparse.

StochasticNets are also very different from HashNets [3], where connection weights are randomly grouped into hash buckets, with each bucket sharing the same weights, to explicitly sparsifying into the network, since there is no notion of grouping/merging in StochasticNets; the formed StochasticNets are naturally sparse due to the formation process. In fact, stochastic strategies such as HashNets, Dropout, and DropConnect can be used in conjunction with StochasticNets. Figure 5 demonstrates the difference of StochasticNet compared to other methods in network model, network formation, training and testing situations.

5. Experimental Results

To investigate the efficacy of efficient feature learning via StochasticNets, we form deep convolutional StochasticNets and train the constructed StochasticNets using the CIFAR-10 [20] image dataset for generating generic features. The StochasticNet formation in this study is implemented via MatConvNet framework [29] and all experi-

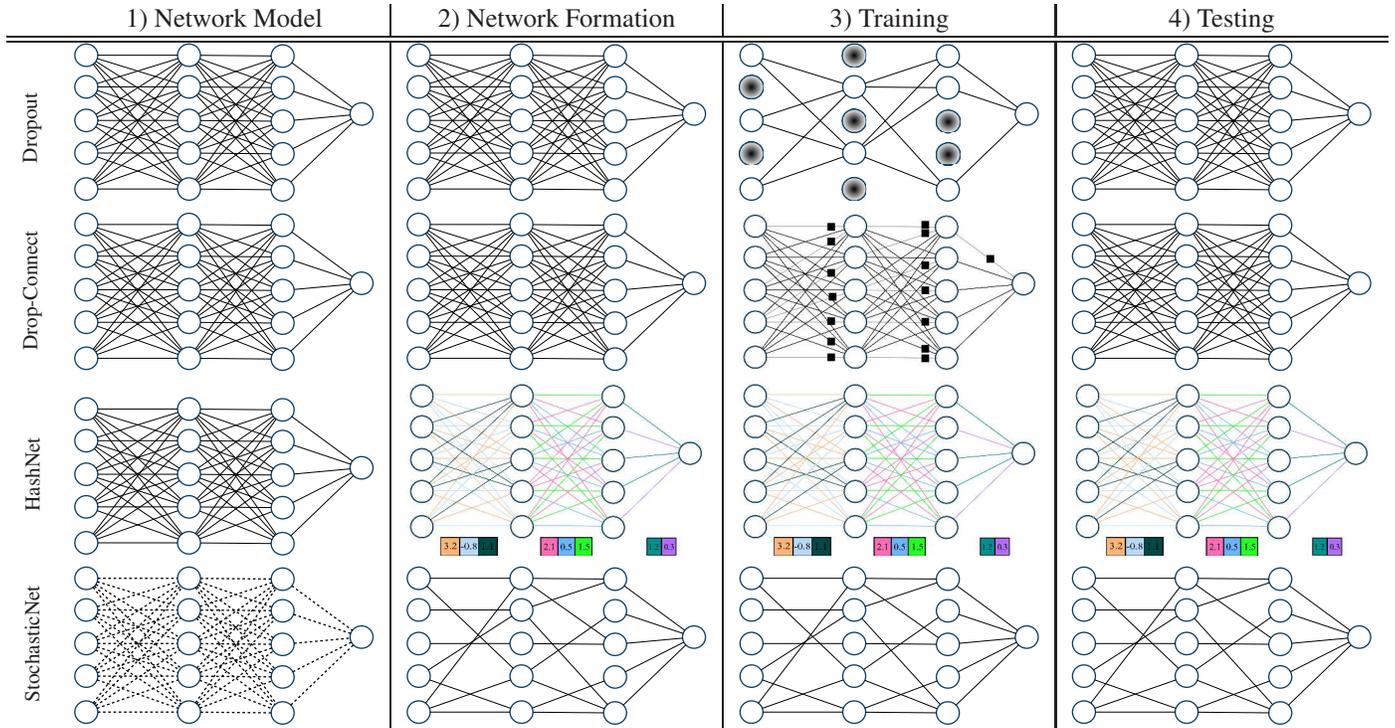


Figure 5. Visual comparison among different related methods to StochasticNet. Constructing a deep neural network can be divided into 4 steps. 1) Network Model: dropout, drop-connect and Hashnet methods start with the conventional convNet where as stochastic starts with a random graph. 2) Network Formation: dropout and drop-connect approaches utilize the formed network from previous step while Hashent groups edges to have same weight (shown in same color) and stochastic net samples the constructed random graph. 3) Training: dropout and drop-connect methods drop activation function results or the weight connectivities randomly to regularize the training network while the Hashnet and StochasticNet obtain the same configuration as previous step (i.e., Network Formation). 4) Testing: in this step dropout, drop-connect and Hashnet approximate all network parameters and get the output based on the complete network structure while StochasticNet outputs by the sparse trained network.

ments were done via MatConvNet framework. Based on the trained StochasticNets, features are then extracted for the SVHN [24] and STL-10 [4] image datasets and image classification performance using these extracted deep features within a neural network classifier framework are then evaluated in a number of different ways. It is important to note that the main goal is to investigate the efficacy of feature learning via StochasticNets and the influence of stochastic connectivity parameters on feature representation performance, and not to obtain maximum absolute classification performance; therefore, the performance of StochasticNets can be further optimized through additional techniques such as data augmentation and network regularization methods.

Datasets

The CIFAR-10 image dataset [20] consists of 50,000 training images categorized into 10 different classes (5,000 images per class) of natural scenes. Each image is an RGB image that is 32×32 in size. The MNIST image dataset [22] consists of 60,000 training images and 10,000 test images

of handwritten digits. Each image is a binary image that is 28×28 in size, with the handwritten digits are normalized with respect to size and centered in each image. The SVHN image dataset [24] consists of 604,388 training images and 26,032 test images of digits in natural scenes. Each image is an RGB image that is 32×32 in size. Finally, the STL-10 image dataset [4] consists of 5,000 labeled training images and 8,000 labeled test images categorized into 10 different classes (500 training images and 800 training images per class) of natural scenes. Each image is an RGB image that is 96×96 in size. The images were resized to 32×32 to have the same network configuration for all experimented datasets for consistency purposes. Note that the 100,000 unlabeled images in the STL-10 image dataset were not used in this paper.

StochasticNet Configuration

The deep convolutional StochasticNets used in this paper are realized based on the LeNet-5 deep convolutional neural network [22] architecture, and consists of three convo-

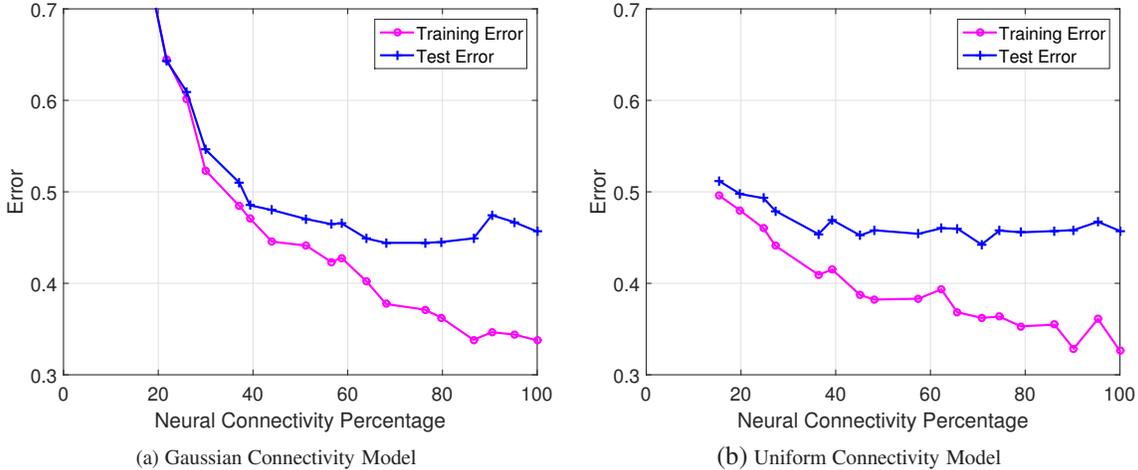


Figure 6. Training and test error versus the number of neural connections for the STL-10 dataset. Both Gaussian and uniform neural connectivity models were evaluated. Note that neural connectivity percentage of 100 is equivalent to ConvNet, since all connections are made. As seen a StochasticNet with fewer neural connectivity compared to the ConvNet provides better than or similar accuracy to the ConvNet.

lutional layers with 32, 32, and 64 receptive fields for the first, second, and third convolutional layers, respectively, and one hidden layer of 64 neurons, with all neural connections being randomly realized based on probability distributions. For comparison purposes, the conventional ConvNet used as a baseline is configured with the same network architecture as explained above using 5×5 receptive fields.

5.1. Number of Neural Connections

An experiment was conducted to illustrate the impact of the number of neural connections on the feature representation capabilities of StochasticNets. Figure 6 demonstrates the training and test error versus the number of neural connections in the network for the STL-10 dataset. The neural connection probability is varied to achieve the desired number of neural connections for testing its effect on feature representation capabilities.

Figure 6 demonstrates the training and testing error vs. the neural connectivity percentage relative to the baseline ConvNet, for two different spatial neural connectivity models: i) uniform connectivity model, and ii) Gaussian connectivity model. It can be observed that classification using the features from the StochasticNet is able to achieve the better or similar test error as using the features from the ConvNet when the number of neural connections in the StochasticNet is fewer than the ConvNet. In particular, classification using the features from the StochasticNet is able to achieve the same test error as using the features from the ConvNet when the number of neural connections in the StochasticNet is half that of the ConvNet. It can be also observed that, although increasing the number of neural connections resulted in lower training error, it does not exhibit

reductions in test error, and as such it can be observed that the proposed StochasticNets can improve the handling of over-fitting associated with deep neural networks while decreasing the number of neural connections, which in effect greatly reduces the number of computations and thus resulting in more efficient feature learning and feature extraction. Finally, it is also observed that there is a noticeable difference in the training and test errors when using the Gaussian connectivity model when compared to the uniform connectivity model, which indicates that the choice of neural connectivity probability distributions can have a noticeable impact on feature representation capabilities.

5.2. Comparisons with ConvNet for Feature Learning

Motivated by the results shown in Figure 6, a comprehensive experiment were done to investigate the efficacy of feature learning via StochasticNets on CIFAR-10 and utilize them to classify the SVHN and STL-10 image datasets. Deep convolutional StochasticNet realizations were formed with 75% neural connectivity using the Gaussian connectivity model as well as the uniform connectivity model when compared to a conventional ConvNet. The performance of the StochasticNets and the ConvNets was evaluated based on 25 trials and the reported results are based on the best of the 25 trials in terms of training error. Figure 7 and Figure 8 shows the training and test error results of classification of models realized based on the uniform connectivity and the Gaussian connectivity models which use learned deep features from CIFAR-10 using the StochasticNets and ConvNets on the SVHN and STL-10 datasets. It can be observed that, in the case where the uniform connectivity

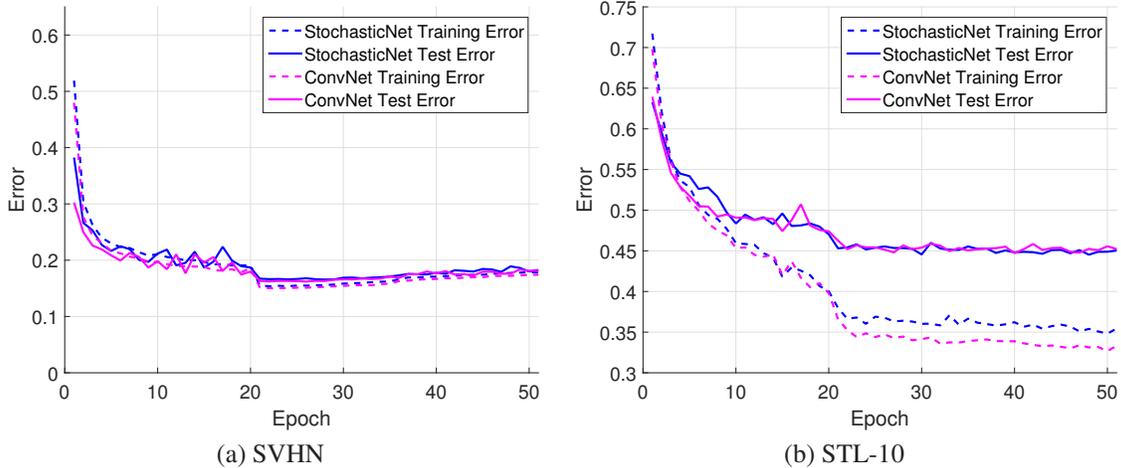


Figure 7. **Uniform distribution realization:** Comparison of classification performance between deep features extracted with a standard ConvNet and that extracted with a StochasticNet containing 75% neural connectivity as the ConvNet. The StochasticNet is realized based on a uniform connectivity model. The StochasticNet results in a 0.5% improvement in relative test error for the STL-10 dataset, as well as provides a smaller gap between the training error and test error. The StochasticNet using a uniform connectivity model achieved the same performance as the ConvNet for the SVHN dataset. The results are reposted based on 25 trials.

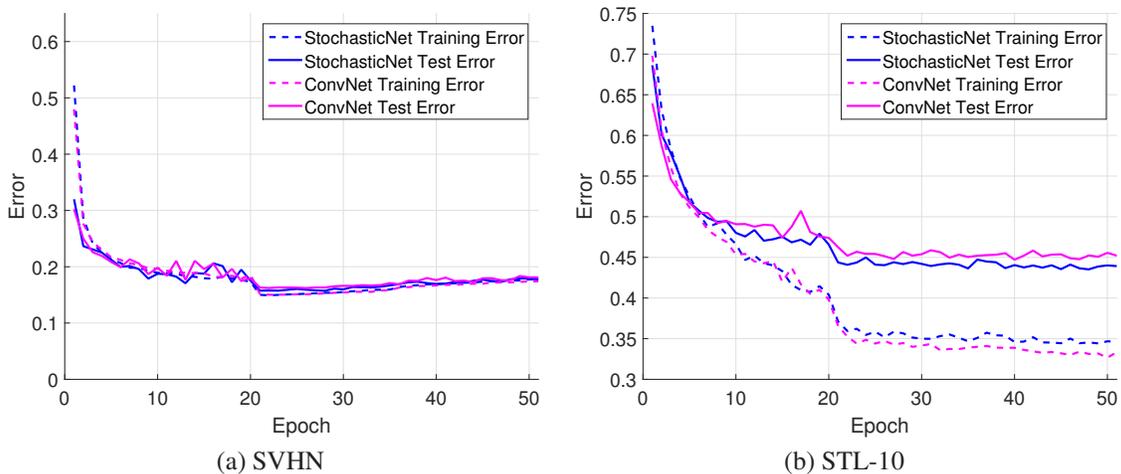


Figure 8. **Gaussian distribution realization:** Comparison of classification performance between deep features extracted with a standard ConvNet and that extracted with a StochasticNet containing 75% neural connectivity as the ConvNet. The StochasticNet is realized based on a Gaussian connectivity model. The StochasticNet results in a 4.5% improvement in relative test error for the STL-10 dataset, as well as provides a smaller gap between the training error and test error. A 1% relative improvement is also observed for the SVHN dataset. The reported performance were computed based on 25 trials.

model is used, the test error for classification using features learned using StochasticNets, with just 75% of neural connections as ConvNets, is approximately the same as ConvNets for both the SVHN and STL-10 datasets (with $\sim 0.5\%$ test error reduction for STL-10). It can also be observed that, in the case where the Gaussian connectivity model is used, the test error for classification using features learned using StochasticNets, with just 75% of neural connections as ConvNets, is approximately the same ($\sim 1\%$ relative test error reduction) as ConvNets for the SVHN dataset. More

interestingly, it can also be observed that the test error for classification using features learned using StochasticNets, with just 75% of neural connections as ConvNets, is reduced by $\sim 4.5\%$ compared to ConvNets for the STL-10 dataset. Furthermore, the gap between the training and test errors of classification using features extracted using the StochasticNets is less than that of the ConvNets, which would indicate reduced overfitting in the StochasticNets.

These results illustrate the efficacy of feature learning via StochasticNets in providing efficient feature learning and

extraction while preserving feature representation capabilities, which is particularly important for real-world applications where efficient feature extraction performance is necessary.

5.3. Relative Feature Extraction Speed vs. Number of Neural Connections

Previous sections showed that StochasticNets can achieve good feature learning performance relative to conventional ConvNets, while having significantly fewer neural connections. We now investigate the feature extraction speed of StochasticNets, relative to the feature extraction speed of ConvNets, with respect to the number of neural connections formed in the constructed StochasticNets. To this end, the convolutions in the StochasticNets are implemented as a sparse matrix dot product, while the convolutions in the ConvNets are implemented as a matrix dot product. For fair comparison, both implementations do not make use of any hardware-specific optimizations such as Streaming SIMD Extensions (SSE) because many industrial embedded architectures used in applications such as embedded video surveillance systems do not support hardware optimization such as SSE.

As with Section 5.1, the neural connection probability is varied in both the convolutional layers and the hidden layer to achieve the desired number of neural connections for testing its effect on the feature extraction speed of the formed StochasticNets. Figure 9 demonstrates the relative feature extraction time vs. the neural connectivity percentage, where relative time is defined as the time required during the feature extraction process relative to that of the ConvNet. It can be observed that the relative feature extraction time decreases as the number of neural connections decrease, which illustrates the potential for StochasticNets to enable more efficient feature extraction.

Interestingly, it can be observed that speed improvements are seen immediately, even at 90% connectivity, which may appear quite surprising given that sparse representation of matrices often suffer from high computational overhead when representing dense matrices. However, in this case, the number of connections in the randomly realized receptive field is very small relative to the typical input image size. As a result, the overhead associated with using sparse representations is largely negligible when compared to the speed improvements from the reduced computations gained by eliminating even one connection in the receptive field. Therefore, these results show that StochasticNets can have significant merit for enabling the feature representation power of deep neural networks to be leveraged for a large number of industrial embedded applications.

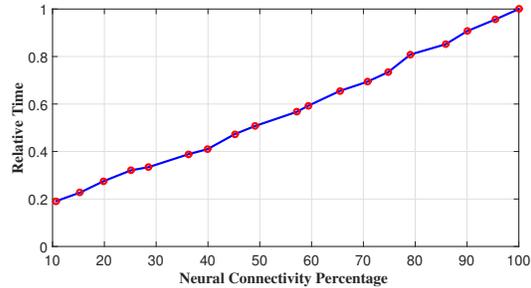


Figure 9. Relative feature extraction time versus the number of neural connections. Note that neural connectivity percentage of 100 is equivalent to ConvNet, since all connections are made.

6. Conclusions

In this paper, we proposed the learning of efficient feature representations via StochasticNets, where sparsely-connected deep neural networks are constructed by way of stochastic connectivity between neurons. Such an approach facilitates for more efficient neural utilization, resulting in reduced computational complexity for feature learning and extraction while preserving feature representation capabilities. The effectiveness of feature learning via StochasticNet was investigated by training StochasticNets using the CIFAR-10 dataset and using the learned features for classification of images in the SVHN and STL-10 image datasets. The StochasticNet features were then compared to the features extracted using a conventional convolutional neural network (ConvNet). Experimental results demonstrate that classification using features extracted via StochasticNets provided better or comparable accuracy than features extracted via ConvNets, even when the number of neural connections is significantly fewer. Furthermore, StochasticNets, with fewer neural connections than the conventional ConvNet, can reduce the over fitting issue associating with the conventional ConvNet. As a result, deep feature learning and extraction via StochasticNets can allow for more efficient feature extraction while facilitating for better or similar accuracy performances.

References

- [1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research (JMLR)*, 3, 2003. 1
- [2] B. Bollobás and F. R. Chung. *Probabilistic combinatorics and its applications*, volume 44. American Mathematical Soc., 1991. 2
- [3] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. *arXiv preprint arXiv:1504.04788*, 2015. 1, 4
- [4] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Inter-*

- national conference on artificial intelligence and statistics*, pages 215–223, 2011. 2, 5
- [5] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International conference on Machine learning (ICML)*. ACM, 2008. 1
- [6] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1), 2012. 1
- [7] P. Erdos and A. Renyi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959. 2
- [8] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello. Hardware accelerated convolutional neural networks for synthetic vision systems. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2010. 1
- [9] E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, pages 1141–1144, 1959. 2
- [10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, 2010. 1
- [11] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2011. 1
- [12] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello. A 240 g-ops/s mobile coprocessor for deep neural networks. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2014. 1
- [13] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al. Deepspeech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014. 1
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*. Springer, 2014. 1
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015. 1
- [16] S. L. Hill, Y. Wang, I. Riachi, F. Schürmann, and H. Markram. Statistical connectivity provides a sufficient foundation for specific functional connectivity in neocortical neural microcircuits. *Proceedings of the National Academy of Sciences*, 109(42), 2012. 1
- [17] K. Iverson. A programming language. In *Proceedings of the May 1-3, 1962, spring joint computer conference*. ACM, 1962. 3
- [18] J. Jin, V. Gokhale, A. Dundar, B. Krishnamurthy, B. Martini, and E. Culurciello. An efficient implementation of deep convolutional neural networks on a mobile coprocessor. In *International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2014. 1
- [19] I. Kovalenko. The structure of a random directed graph. *Theory of Probability and Mathematical Statistics*, 6:83–92, 1975. 2
- [20] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009. 2, 4, 5
- [21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, 2012. 1
- [22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998. 3, 5
- [23] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2. IEEE, 2004. 1
- [24] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 2011. 2, 5
- [25] M. J. Shafiee, P. Siva, and A. Wong. Stochasticnet: Forming deep neural networks via stochastic connectivity. *IEEE Access*, (99), 2016. 2
- [26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1
- [27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 2014. 4
- [28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 1
- [29] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. 4
- [30] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning (ICML)*, pages 1058–1066, 2013. 1, 4
- [31] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013. 1
- [32] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *arXiv preprint arXiv:1505.06798*, 2015. 1