Greedy Deep Transform Learning

Jyoti Maggu
IIIT-Delhi
jyotim@iiitd.ac.in

Angshul Majumdar
IIIT-Delhi
angshul@iiitd.ac.in

***Abstract*— We introduce deep transform learning – a new tool for deep learning. Deeper representation is learnt by stacking one transform after another. The learning proceeds in a greedy way. The first layer learns the transform and features from the input training samples. Subsequent layers use the features (after activation) from the previous layers as training input. Experiments have been carried out with other deep representation learning tools – deep dictionary learning, stacked denoising autoencoder, deep belief network and PCA-Net (a version of convolutional neural network). Results show that our proposed technique is better than all the said techniques, at least on the benchmark datasets (MNIST, CIFAR-10 and SVHN) compared on.**

Keywords*— *deep learning, greedy learning, transform learning

I. Introduction

Transform learning is an analysis formulation; the transform is learnt such that it analyses the signal to generate coefficients. It has been introduced only recently [1-4]. It has mainly been applied for solving inverse problems like denoising and reconstruction. There are only a few studies where it has been used for generating features for solving machine learning problems [5-7]; where the transform was learnt on the training data and the learnt transform was being used on the test data for feature generation.

Transform learning is the analysis equivalent of dictionary learning. In dictionary learning, a basis (dictionary) is learnt so as to regenerate / synthesize the data from the coefficients / features. Dictionary learning has its roots in matrix factorization [8] / sparse coding [9] – it was used for analyzing early vision. Its popularity has resurfaced in the last decade with the success of KSVD [10, 11] (mostly for solving inverse problems in signal processing) and related techniques [12-14] (for solving supervised problems in machine learning).

The success of deep learning is popular knowledge today. The main idea in deep learning is to learn abstract representations. Shallow representation learning tools like autoencoder [15] and restricted Boltzmann machine (RBM) [16] are layered on each other to build deeper architectures. For example nesting one autoencoder into another leads to stacked autoencoders (SAE) [17]; stacking RBMs one after the other (loosely speaking) leads to deep belief network (DBN) [18]. Usually, such deep architectures are used for classification problems leading to deep neural networks, but this is not mandatory. It can be used for other problems where class information is not required; for example in clustering [19] or reconstruction [20].

Dictionary learning, although well known in the machine learning community, has been traditionally overlooked as a candidate for building deep architectures. It is only in recent times, layers of single level dictionaries have been stacked one after the other leading to the framework of deep dictionary learning (DDL) [21-25]; it has been shown in there that deep dictionary learning indeed is a more powerful tool than conventional ones like SAE and DBN.

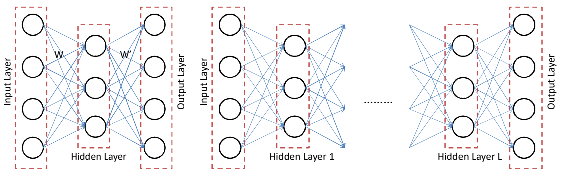
This work introduces deep transform learning. The first layer proceeds like the usual – it learns a transform and corresponding features from the training samples. From the second and subsequent layers, the features (after activation) from the previous layer acts as training samples. We follow the popular greedy learning paradigm in deep learning [26].

The rest of the paper is organized into several sections. The following section discusses some relevant prior studies. The proposed formulation is given in section 3. The experimental results are described in section 4. The conclusions of this work are discussed in section 5.

II. Literature Review

A. *Deep Representation Learning*

Deep learning is a vast topic today. We will only be discussing some studies relevant to this paper. This includes SAE, DBN and DDL.

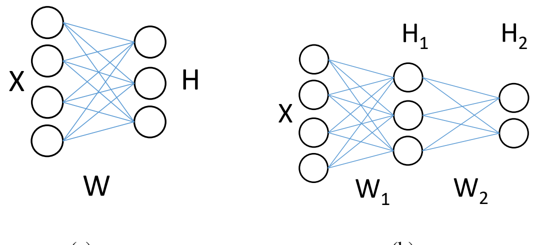


(a) (b)
Fig. 1. (a) Autoencoder. (b) Stacked Autoencoder.

First we discuss the autoencoder. It is an unsupervised neural network where the input and the output are the same (Fig. 1a). An encoder (*W*) learns to map the input data to the representation and a decoder (*W'*) generates the input (output) from the learnt representation. The main idea here is that the representation should be learnt such that, the information content is preserved (usually in the Euclidean sense).

Stacked autoencoders are built by nesting one autoencoder inside the other (Fig. 1b). Solving this deep architecture is complex. Therefore a greedy paradigm is followed. First the outermost autoencoder is solved, using the training samples as input. Once, the representation from the first layer is obtained, it is used as input for the second layer. This continues into the deeper layers.

Traditionally autoencoders have been used for pre-training deep neural networks for classification. The decoder portions of all the autoencoders are removed and the targets are attached to the deepest representation layer. The entire deep neural network is fine-tuned by backpropagating errors.

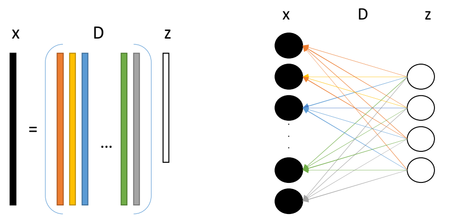


(a) (b)
Fig. 2. (a) RBM. (b) Deep Boltzmann Machine.

Another popular representation learning tool used commonly in deep learning (especially in speech processing) is RBM. It is an undirected graphical model (Fig. 2a). It learns a representation from the training data such that cosine similarity between the projection of the data and the features is high. In a sense, RBM is maintaining the information content of the inputs in terms of cosine similarity. RBM has a probabilistic formulation.

Deep Boltzmann Machine (DBM) is formed by stacking one RBM after another (Fig. 2b); this too is an undirected graph. A variation of the DBM is the deep belief network (DBN). The representation from the deepest layer can be used as it is, but most often it is used for pre-training deep neural networks. The targets are attached to the deepest representation layer and fine-tuned by backpropagating errors.

The usual interpretation of dictionary learning is different. It learns a basis for representing the data (Fig. 3a). The columns of the dictionary are called 'atoms'. The atoms can be interpreted in a different manner. Instead of thinking of them of them as basis for representation, we can think of them as connections between the input and the representation layer. To showcase the similarity, we have kept the color scheme intact in Fig. 3b.



(a) (b)
Fig. 3. (a) Dictionary Learning. (b) Neural Network Interpretation.

Unlike the usual neural network, which is directed from the input to the representation, the dictionary learning can be viewed as a network that points in the reverse direction – from representation to the input. This is what is called 'synthesis framework' in signal processing. The dictionary is learnt so that the features (along with the dictionary) can synthesize / generate the data.

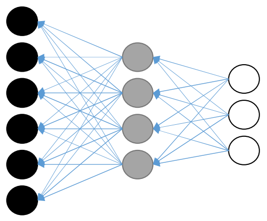


Fig. 4. Deep Dictionary Learning

Given the neural network interpretation, one can easily extend it to deeper layers. For the first layer, a dictionary is learnt to represent the data. In the second layer, the representation from the first layer acts as input and it learns a second dictionary to represent the features from the first level. This concept can be further extended to deeper layers. This constitutes deep dictionary learning (Fig. 4).

B. *Transform Learning*

Transform learning is relatively new. Hence, we discuss it in detail. Transform learning analyses the data by learning a transform / basis to produce coefficients. Mathematically this is expressed as,

$$TX = Z \tag{1}$$

Here T is the transform, X is the data and Z the corresponding coefficients.

The following transform learning formulation was proposed [1, 2] –

$$\min_{T,Z} \|TX - Z\|_F^2 + \lambda\left(\|T\|_F^2 - \log\det T\right) + \mu\|Z\|_1 \tag{2}$$

The factor $-\log\det T$ imposes a full rank on the learned transform; this prevents the degenerate solution (*T=0, Z=0*). The additional penalty $\|T\|_F^2$ is to balance scale; without this $-\log\det T$ can keep on increasing producing degenerate results in the other extreme.

In [1, 2], an alternating minimization approach was proposed to solve the transform learning problem. This is given by –

$$Z \leftarrow \min_{Z} \|TX - Z\|_F^2 + \mu\|Z\|_1 \tag{3a}$$

$$T \leftarrow \min_{T} \|TX - Z\|_F^2 + \lambda\left(\varepsilon\|T\|_F^2 - \log\det T\right) \tag{3b}$$

Updating the coefficients (3a) is straightforward. It can be updated via one step of soft thresholding. This is expressed as,

$$Z \leftarrow signum(TX) \cdot \max\left(0, abs(TX) - \mu\right) \tag{4}$$

Here ⊙ indicates element-wise product.

In the initial paper on transform learning [1], a non-linear conjugate gradient based technique was proposed to solve the transform update. In the more refined version [2], with some linear algebraic tricks they were able to show that a closed form update exists for the transform.

$$XX^T + \lambda\varepsilon I = LL^T \tag{5a}$$

$$L^{-1}XZ^T = USV^T \tag{5b}$$

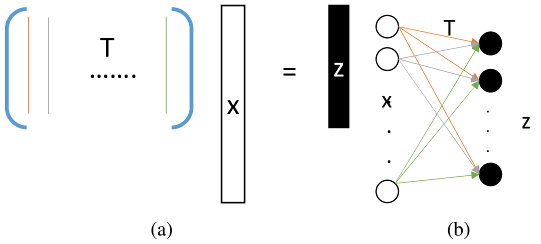
$$T = 0.5R\left(S + (S^2 + 2\lambda I)^{1/2}\right)Q^T L^{-1} \tag{5c}$$

The first step is to compute the Cholesky decomposition; the decomposition exists since $XX^T + \lambda\varepsilon I$ is symmetric positive definite. The next step is to compute the full SVD. The final step is the update step. The proof for convergence of such an update algorithm can be found in [3].

There are only a handful of papers on this topic. Theoretical aspects of transform learning are discussed in [1-3]. In [4] it is used to solve inverse problems. Exactly the same formulation has been dubbed as 'analysis sparse coding' when applied to feature generation [5].

III. Proposed Deep Transform Learning

Transform learning is the analysis version of dictionary learning. Just as we showed the neural network interpretation of dictionary learning, we will show the same for transform learning.

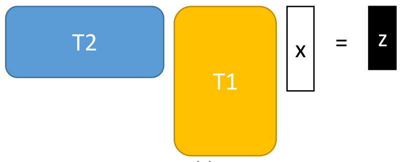


(a) (b)

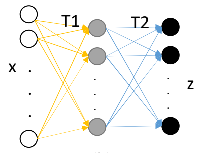
Fig. 5. (a) Transform Learning. (b) Neural Network Interpretation.

The usual depiction of transform learning is shown in Fig. 5a. The transform (*T*) operates on the data (*x*) to generate the feature (*z*). Instead of interepreting the columns of the transform as basis vectors, we can think of them as connections from each element of the input vector to elements in the feature vector. We have kept the color scheme same for ease of interpretation (Fig. 5b). Note that since transform learning is an analysis framework, the data points to the usual direction – from input to the features.

Our proposed deep transform learning is the multi-layer extension of the shallow one. It can be thought of as application of multiple levels of transforms to generate the coefficients. The schematic explanation is shown in Fig. 6.



(a)



(b)

Fig. 6. (a) Deep Transform Learning. (b) Neural Network Interpretation.

Mathematically this is expressed as follows –

$$T_N(\varphi...(T_2(\varphi(T_1X))) = Z \tag{6}$$

Here φ denotes the activation function; without which all the transforms will collapse into a single one.

Following the greedy learning paradigm [25, 26], we solve (6) one layer at a time. With the substitution $\varphi(T_{N-1}...(T_2(\varphi(T_1X)) = Z_{N-1}$, (6) can be expressed as,

$$T_N Z_{N-1} = Z \tag{7}$$

where $\varphi(T_{N-1}...(T_2(\varphi(T_1X)) = Z_{N-1}$. This can be alternately expressed as,

$$T_{N-1}...(T_2(\varphi(T_1X)) = \varphi^{-1}(Z_{N-1}) \tag{8}$$

With the substitution $\varphi(T_{N-2}...(T_2(\varphi(T_1X)) = Z_{N-2}$, we have for the next layer,

$$T_{N-1}Z_{N-2} = \varphi^{-1}(Z_{N-1}) \tag{9}$$

Continuing the substitution in this fashion till the final layer, we have

$$T_1X = \varphi^{-1}(Z_1) \tag{10}$$

Note that for all the layers, it is easy to invert the activation φ since they operate element-wise.

We start solving for the different layers of transforms in backward direction; starting from (10); this is easily solved using the standard transform learning formulation.

$$\min_{T_1,Z_1} \|T_1X - Z_1\|_F^2 + \lambda\left(\|T_1\|_F^2 - \log\det T_1\right) \tag{11}$$

Note that, we have dropped the sparsity penalty. Such penalties are required for inverse problems, but we are not aware of any theoretical or intuitive necessity for such terms for learning problems. Removing the sparsity penalty makes the updates for the coefficients even easier; in each iteration it is just $Z_1 = T_1X$.

Once Z_1 is solved for, it acts as the input for the second layer for solving T_2 and Z_2; as shown below –

$$\min_{T_2,Z_2} \|T_2Z_1 - Z_2\|_F^2 + \lambda\left(\|T_2\|_F^2 - \log\det T_2\right) \tag{12}$$

Now Z_2 acts as an input for the third layer and so on. This is continued till the final layer of transform (T_N). This completes the training process. The advantage of such a greedy training paradigm is that for each level, we only need solving a shallow transform learning problem which has algorithms with convergence guarantees.

During testing, the objective is to generate the test feature (z_{test}) given the input test sample (x_{test}). This is expressed as,

$$T_N(\varphi ...(T_2(\varphi(T_1 x_{test}))) = z_{test} \quad (17)$$

The multiple layers of transforms have already been learnt during the training phase. Therefore during testing, one simply needs to apply them one after the other.

IV. Experimental Results

We experiment on the MNIST, CIFAR-10 and SVHN datasets; these are well known benchmarking datasets in deep learning [27]. Owing to limitations in space we are delving into the dataset description. MNIST does not require any pre-processing. Standard pre-processing [28] is performed on the CIFAR-10 and SVHN.

In the first set of experiments, we want to show that the accuracy of deep transform learning indeed improves when one goes deeper; the number of basis are halved in each layer. To elucidate, we have used a simple nearest neighbor (NN) classifier. The results are shown in Table I for levels 1, 2 and 3.

TABLE I Classification with NN

Dataset	Level 1	Level 2	Level 3
MNIST	97.27	97.66	97.94
CIFAR-10	81.12	81.89	82.60
SVHN	91.97	92.68	93.00

It has already been shown in [6] that a single layer of transform learning yields better results than other single layer representation learning tools including dictionary learning, autoencoder and restricted Boltzmann machine. Therefore it is expected that by going deeper we will improve upon their deeper counterparts.

We compare our proposed technique with Deep Belief Network (DBN) [18], Stacked Denoising Autoencoder (SDAE) [17] and Deep Dictionary Learning (DDL) [21]. Trained models for all the tools are available for these popular datasets; these models are assumed to yield the best possible features. All of them are three layered.

Our proposed robust dictionary learning is also three layered architecture. We follow the simple rule of thumb where the number of nodes /basis are halved in each layer. The representation from the deepest layer is used to train the classifiers. We use a tanh activation function. We have only one parameter λ; for all the layers it has been fixed at 0.1. This value was obtained on a validation dataset (CIFAR-100).

We choose to use two non-parametric classifiers – Nearest Neighbor (Table II), Sparse Representation based Classifier (SRC) (Table III) [29] and a parametric classifier – Support Vector Machine with RBF kernel (Table IV). The SVM was tuned via grid search. We test all the tools on the same classifiers in order to analyse the representation capacity of different techniques. In all cases, our proposed method yields the best results.

TABLE II Classification with NN

Dataset	Proposed	SDAE	DBN	DDL
MNIST	**97.94**	97.33	97.05	97.75
CIFAR-10	**82.60**	78.62	73.96	81.09
SVHN	**93.00**	91.11	88.29	92.26

TABLE III Classification with SRC

Dataset	Proposed	SDAE	DBN	DDL
MNIST	**98.96**	98.33	98.43	98.81
CIFAR-10	**85.06**	79.32	75.02	83.75
SVHN	**94.55**	92.05	90.11	93.62

TABLE IV Classification with SVM

Dataset	Proposed	SDAE	DBN	DDL
MNIST	**98.94**	97.05	98.44	98.64
CIFAR-10	**85.55**	78.90	74.30	84.96
SVHN	**94.42**	92.60	89.70	93.81

We have also compared with the PCA-Net architecture [30] of convolutional neural network. Since it has an inbuilt classifier, we could not compare it in any of the tables. The results for PCA-Net are 98.88 for MNIST, 83.76 for CIFAR-10 and 92.24 for SVHN. Except for the simple NN classifier, our proposed method (with SRC and SVM) does better than PCA-Net.

The testing and training times are shown in Table V. We have only shown it for the MNIST, since the other datasets are of similar size. The results show that our proposed method has a very short training time; even shorter than deep dictionary learning. This is because our method converges very faster – 20 iterations per layer. The testing time is at par with other deep learning tools (requiring matrix vector products); it is much shorter than DDL.

TABLE V Timing in Seconds

Mode	Proposed	SDAE	DBN	DDL
Training	25	120408	30071	107
Testing	50	61	50	79

V. Conclusion

This work introduces a new deep learning tool – deep transform learning. It has been compared with existing unsupervised deep learning methods – SDAE, DBN, DDL and PCA-Net. In all cases (at least for the benchmark datasets experimented on) we improve upon them.

This is the first work introducing deep transform learning. Therefore it is not fair to compare its performance against advanced regularized deep learning tools like sparse autoencoder [31], contractive autoencoder [32], sparse DBN [33], or their supervised counterparts [34, 35]. Neither is it fair to compare against powerful stochastic regularization techniques like DropOut [36] and DropConnect [37]. Comparison will only be fare when we will incorporate these deterministic / stochastic regularization techniques into the deep transform learning framework; which we intend to pursue in the future.

References

[1] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms", IEEE Transactions on Signal Processing, Vol. 61 (5), pp. 1072-1086, 2013.

[2] S. Ravishankar, B. Wen and Y. Bresler. "Online sparsifying transform learning-Part I: Algorithms", IEEE Journal of Selected Topics in Signal Processing, Vol. 9 (4), pp. 625-636, 2015.

[3] S. Ravishankar and Y. Bresler, "Online Sparsifying Transform Learning-Part II: Convergence Analysis", IEEE Journal of Selected Topics in Signal Processing, Vol. 9 (4), pp. 637-646, 2015.

[4] S. Ravishankar and Y. Bresler, "Efficient Blind Compressed Sensing using Sparsifying Transforms with Convergence Guarantees and Application to MRI," SIAM Journal on Imaging Sciences, 2015.

[5] S. Shekhar, V. M. Patel and R. Chellappa, "Analysis sparse coding models for image-based classification," IEEE ICIP, 2014.

[6] J. Maggu and A. Majumdar, “Alternate Formulation for Transform Learning”, ACM ICVGIP, 2016

[7] J. Maggu and A. Majumdar, “Robust Transform Learning”, IEEE ICASSP 2017.

[8] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization", Nature 401 (6755), pp. 788–791, 1999.

[9] B. Olshausen and D. Field, "Sparse coding with an overcomplete basis set: a strategy employed by V1?", Vision Research, Vol. 37 (23), pp. 3311-3325, 1997.

[10] M. Aharon, M. Elad and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation", IEEE Transactions on Signal Processing, Vol. 54 (11), pp. 4311-4322, 2006.

[11] M. Elad and M. Aharon, "Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries," IEEE Transactions on Image Processing, Vol.15 (12), pp. 3736-3745, 2006.

[12] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. "Supervised dictionary learning". Advances in Neural Information Processing Systems, 2009.

[13] K. Huang and S. Aviyente. "Sparse representation for signal classification". Advances in Neural Information Processing Systems, 2007.

[14] Z. Jiang, Z. Lin and L. S. Davis, "Learning A Discriminative Dictionary for Sparse Coding via Label Consistent K-SVD", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, pp. 2651-2664, 2013

[15] G. E. Hinton and R. S. Zemel. "Autoencoders, minimum description length, and Helmholtz free energy." Advances in neural information processing systems, 1994.

[16] R. Salakhutdinov, A. Mnih and G. E. Hinton, “Restricted Boltzmann machines for collaborative filtering”, ACM ICML, 2007.

[17] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P. A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion", Journal of Machine Learning Research, Vol. 11, pp. 3371-3408, 2010.

[18] G. E. Hinton, S. Osindero and Y. W. Teh, “A fast learning algorithm for deep belief nets”, Neural computation, Vol. 18(7), 1527-1554, 2006.

[19] F. Tian, B. Gao, Q. Cui and E. Chen and T.-Y. Liu, “Learning Deep Representations for Graph Clustering”, AAAI, 2014.

[20] C. Dong, C. C. Loy, K. He and X. Tang, “Image super-resolution using deep convolutional networks”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 38(2), pp. 295-307, 2016.

[21] S. Tariyal, A. Majumdar, R. Singh and M. Vatsa, “Deep Dictionary Learning”, IEEE ACCESS, 2016.

[22] V. Singal and A. Majumdar, "Majorization Minimization Technique for Optimally Solving Deep Dictionary Learning", Neural Processing Letters, (accepted).

[23] I. Manjani, S. Tariyal, M. Vatsa, R. Singh, A. Majumdar, Detecting Silicone Mask based Presentation Attack via Deep Dictionary Learning, IEEE Transactions on Information Forensics and Security, (accepted).

[24] A. Majumdar and R. K. Ward, “Robust Greedy Deep Dictionary Learning for ECG Arrhythmia Classification”, IEEE IJCNN, 2017.

[25] V. Singhal, H. Agrawal, S. Tariyal and A. Majumdar, “Discriminative Robust Deep Dictionary Learning for Hyperspectral Image Classification”, IEEE Transactions on Geosciences and Remote Sensing

[26] Y. Bengio, P. Lamblin, P. Popovici and H. Larochelle, “Greedy Layer-Wise Training of Deep Networks”, NIPS, 2007.

[27] http://deeplearning.net/datasets/

[28] G.E. Hinton, N. Srivastave, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012

[29] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry and Y. Ma, "Robust face recognition via sparse representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31(2), pp. 210-227, 2009.

[30] T. H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng and Y. Ma, "PCANet: A Simple Deep Learning Baseline for Image Classification?," IEEE Transactions on Image Processing, vol. 24, no. 12, pp. 5017-5032, 2015.

[31] K. Cho, "Simple sparsification improves sparse denoising autoencoders in denoising highly noisy images", ACM ICML, 2013.

[32] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, “Contractive Auto-Encoders: Explicit Invariance During Feature Extraction”, ACM ICML, 2011

[33] H. Lee, C. Ekanadham and A. Y. Ng, “Sparse deep belief net model for visual area V2”, NIPS, 2008.

[34] A. Majumdar, M. Vatsa and R. Singh, "Face Recognition via Class Sparsity based Supervised Encoding", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016

[35] A. Sankaran, G. Sharma, R. Singh, M. Vatsa and A. Majumdar, "Class Sparsity Signature based Restricted Boltzmann Machines", Pattern Recognition, 2016.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", Journal of Machine Learning Research, Vol. 15 (1), pp. 1929-1958, 2014.

[37] L. Wan, M. Zeiler, S. Zhang, Y. LeCun and R. Fergus, Regularization of neural networks using dropconnect. ACM ICML, 2013.