JMLR: Workshop and Conference Proceedings 44 (2015) 49-59 NIPS 2015

The 1st International Workshop "Feature Extraction: Modern Questions and Challenges"

Stage-wise Training: An Improved Feature Learning Strategy for Deep Models

Elnaz Barshan EBARSHAN@UWATERLOO.CA
Department of System Design Engineering
University of Waterloo
Waterloo, Canada

Paul Fieguth PFIEGUTH@UWATERLOO.CA
Department of System Design Engineering
University of Waterloo
Waterloo, Canada

Editor: Afshin Rostamizadeh

Abstract

Deep neural networks currently stand at the state of the art for many machine learning applications, yet there still remain limitations in the training of such networks because of their very high parameter dimensionality. In this paper we show that network training performance can be improved using a stage-wise learning strategy, in which the learning process is broken down into a number of related sub-tasks that are completed stage-by-stage. The idea is to inject the information to the network *gradually* so that in the early stages of training the "coarse-scale" properties of the data are captured while the "finer-scale" characteristics are learned in later stages. Moreover, the solution found in each stage serves as a prior to the next stage, which produces a regularization effect and enhances the generalization of the learned representations. We show that decoupling the classifier layer from the feature extraction layers of the network is necessary, as it alleviates the diffusion of gradient and over-fitting problems. Experimental results in the context of image classification support these claims.

Keywords: Deep Learning, Feature Extraction, Stage-wise Training

1. Introduction

In recent years we have witnessed the exceptional success of "learned" features using deep models, in contrast to hand-crafted features, in many number of applications including image classification (Krizhevsky and Hinton (2009); Farabet et al. (2013)), speech recognition (Dahl et al. (2012); Deng et al. (2013)), language modeling (Arisoy et al. (2012); Mikolov (2012)) and information retrieval (Huang et al. (2013); Salakhutdinov and Hinton (2009)). The power of these deep models is rooted in their nonlinear multi-layer architecture which allows them to learn complicated input-output relationships and to extract high-level abstract features.

The origin of deep models, and more specifically deep neural networks, dates back to more than twenty years ago. However, training these networks for general use was impractically slow. The significant growth in computational power (particularly in GPUs and distributed computing) and access to large labeled data sets (e.g., ImageNet (Deng

et al. (2009)) paved the way for the popularity of deep models. Moreover, effective methods were proposed for dealing with the two major obstacles in training deep nets:

Overfitting stems from the very large number of parameters present in deep networks. The key, then, is to employ regularization techniques to limit the degrees of freedom in the parameter space. Among the classic regularizers for neural networks we find early-stopping, Tikhonov regularization (i.e., L2 weight decay) (Tikhonov (1943)) and lasso (i.e., L1 regularization) (Tibshirani (1996)). Specialized regularizers for deep neural networks include convolutional weight-sharing (LeCun et al. (1998); Lee et al. (2009)), dropout (Srivastava et al. (2014)) and drop-connect (Wan et al. (2013)).

The diffusion of gradients is caused by the progressive vanishing of the error as it is back-propagated to earlier layers of the network. The most widely used methods to address this problem are layer-wise pre-training (Hinton et al. (2006); Bengio et al. (2007)), piecewise linear activation functions (Nair and Hinton (2010)) and transfer learning (Caruana (1995); Bengio (2012)).

Despite the remarkable advances in this area, the training of deep networks remains challenging and continues to motivate further developments. In particular, our research described in this paper focuses on the issues of overfitting and gradient-diffusion into two questions of interest:

1. How should computational resources be distributed between learning the classifier and the feature extractors?

2. Should all of the information (i.e., constraints) be fed to the network at once? Or is it preferable for the network to be guided step-by-step towards learning more discriminative features through a gradual, strategic presentation of the information?

One of the structural properties of deep networks is that all of the feature extraction layers and the classifier layer are trained at the same time. Recently, Yosinski et al. (2014) showed that the feature extractors at successive layers are co-adopted and there is a complicated interaction between them. Therefore, training of feature extractor layers can not be factorized and it is important to learn them at the same time. However the same limitation does not necessarily apply to the classifier layer, leading to a question whether the classifier and feature extraction layers should be trained at the same time. Although in general coupled training of the feature extractors and classifier results in a better performance (Gönen (2014); Mohri et al. (2015)), it is not clear to what extent these two steps should be co-adapted. In this paper we show that for deep nets, training indeed benefits from treating the classifier layer separately, and that the problem of diffusion of gradient can be addressed by preventing the complex co-adaptation of the feature extraction layers with the classification layer.

The second question that we raise in this paper is whether we can increase the generalization of the learned features by the gradual injection of information to the network during training. In particular, given an image classification problem, our goal is to develop stage-wise learning, whereby the network is first steered in the direction of capturing discriminative information related to coarse-scale structures (i.e., shape features). Then, by

gradually increasing the level of detail presented at the input, the learned feature extractors are fine-tuned to grasp the discriminative information related to the fine-scale image characteristics (i.e., appearance features).

Motivated by these questions, we propose a stage-wise training framework for representation learning using deep networks in which overfitting can be avoided through stage-wise evolution of the information fed to the network, and whereby gradient diffusion can be addressed by decoupling the feature extraction layers from the classifier layer across successive training stages.

2. Related Works

2.1 Unsupervised Layer-wise Pre-training

Layerwise Pre-training (Hinton et al. (2006)) played a significant role in revitalizing deep nets. As it comes from its name, the main idea behind this method is to train only one layer of the network at a time, starting from the first layer. After training each layer, its computed output is considered as the input for training the next layer. This layer-wise pre-training strategy is usually performed in an unsupervised way because of two reasons: 1) cheap access to abundant unlabeled data 2) avoiding overfitting due to the large number of parameters per layer. The pre-trained weights are used to initialize the network for a fine-tuning stage where all of the layers are trained together.

The optimization explanation for the effectiveness of layer-wise pre-training is that, this method initializes the network at a location where it is more likely to converge to a good local minimum (Bengio et al. (2007)). In addition, initializing the network parameters acts as a regularizer in the sense that it restricts the parameters to the regions corresponding to the input distribution (Erhan et al. (2010)).

Due to its layer-wise nature, this initialization method does not take into account the interactions between successive layers of the network.

2.2 Transfer Learning

The goal of transfer learning for deep nets (Bengio (2012)) is to use the related information in a *base* dataset to initialize the parameters of a model on a *target* dataset. Assume that our target task is to learn a deep net N_T on the target dataset D_T. Given a base dataset D_B with similar general properties to D_T, one can train a deep net N_B on that and *transfer* the learned representations from N_B to N_T. The transfered features are used to initialize some layers of N_T which can be kept frozen or fine-tuned using D_T, depending on the size of D_B and D_T and their common characteristics. Since the specificity of the learned features to the target task increases as we move towards the top layers of the network, this method is used to initialize the early layers of the network[1]. Transfer learning is used to prevent overfitting specially when D_B is much larger than D_T. A thorough study of this method for deep nets can be found in (Yosinski et al. (2014)).

It is important to note that, the applicability of transfer learning is limited to tasks that are supported with a related and large base dataset.

1. Another reason is that early layers suffer more from the diffusion of gradients problem.

3. Stage-wise Training

Neural networks are discriminative models focused on minimizing the prediction error with respect to some architectural priors, such as the number of layers, the number of neurons at each layer, and the type of the activation function. To distinguish between different classes, these networks aim at learning features that are *unique* to each class, as opposed to the *typical* features of a class (Nguyen et al. (2014)). However, this training strategy can produce counter-intuitive results, for example how a deep network trained on the ImageNet dataset mistakenly classified a black and yellow striped pattern as a school bus (Nguyen et al. (2014)). The reason for such a mistake is that in decision making no priority is assigned to the coarse properties of the data compared relative to its detailed characteristics.

To address this problem, we propose a stage-wise training framework in which the information in the training data is presented to the network *gradually*. In this way, at the early stages of training the network has access to only a subset of the data, specifically the coarse-scale properties of the data, such that the network learns to perform prediction by extracting features at that coarse scale. Then, during following stages, finer information is provided to the network and the learned feature extractors from previous stages are permitted to evolve to perform better predictions. In other words, the learned feature extractors at each stage act as a prior for feature leaning at the next stage (see figure 1).

3.1 Definition of a Training Stage

Let $s \in \{1, 2, ..., S\}$ be the current stage of training. The training set in stage s is denoted by $T_s = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n_s} \subseteq \mathcal{X}_s \times \mathcal{Y}_s$, where $\mathcal{X}_s \subset \mathbb{R}^{p_s}$ and $\mathcal{Y}_s \subset \{0, 1\}^c$ are the input and output domains[2]. At stage s, the learning algorithm $A(.,.)$ takes the training set and the initial value of the parameters W_s^{init} as input, and outputs the learned parameters W_s:

$$W_s = A(T_s, W_s^{init}), \tag{1}$$

where A is, in principle, any state of the art learning strategy currently established in the research literature.

3.2 Connection between Successive Stages

Assuming that the network is randomly initialized at the first stage,

$$\mathbf{W}_1^{init} := \text{random} \tag{2}$$

a natural way of connecting successive stages would be as follows:

$$\mathbf{W}_s^{init} := \mathbf{W}_{s-1} \text{ for all } s \in \{2, ..., S\}. \tag{3}$$

However, if we consider the multi-layer structure of the network and the properties of the error back-propagation algorithm, some layers of the network require special treatment. Training of neural networks is a gradient-based optimization procedure and the gradient of the objective function is back-propagated through the layers of the network. In multi-layer networks, this training strategy usually suffers from the problem of gradient diffusion:

2. Note that training instances of different stages can be from different domains

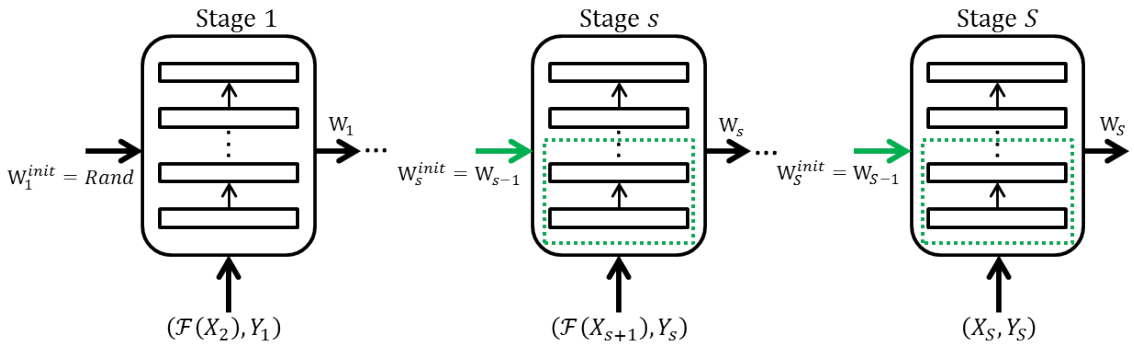


Figure 1: Schematic diagram of stage-wise training with information evolution. Note that only the feature extractor layers (and not the classifier) are initialized using the previous stage.

the back-propagated gradients vanish quickly as the depth of the network increases. Consequently the top layers learn faster than the more distant earlier layers of the network. Furthermore the last layer of the network alone, the classifier layer, has sufficient degrees of freedom (free parameters) to model the entire labeled data by itself. Consequently, the classifier layer is more prone to overfitting than the feature extraction layers.

It follows, then, that in any sort of stage-wise training framework it is important to use the previous stage to initialize the feature extraction layers, and not the classifier layer, to avoid overfitting at the classifier layer.

3.3 Stage-wise Information Evolution

In the proposed stage-wise framework, the training information passed to each stage should be evolved gradually. Considering the definition of a training stage, this objective can be met in a number of ways:

1. The evolution of the input domain $\mathcal{X}_s$,
2. The evolution of the output domain $\mathcal{Y}_s$, or
3. The evolution of training set T_s.

In this paper we focus on the first way, i.e., evolution of the input domain.

Assume that X_S is the original representation of the input data; that is, our target is to arrive at the usual training data at the final stage S. We therefore require a stage-to-stage mapping operator $\mathcal{F}$ which projects X_s, the input data at stage s, to lower-dimensional X_{s-1}:

$$\begin{aligned} X_{s-1} &:= \mathcal{F}(X_s) \\ \mathcal{F} &: \mathcal{X}_s \mapsto \mathcal{X}_{s-1} \quad \text{where} \quad p_s > p_{s-1} \end{aligned} \tag{4}$$

It is important to note that this information evolution method should be used with a *convolutional* weight-sharing scheme for all of feature extraction layers so that the number

of parameters becomes independent of the input dimension. The parameter count independence is essential to allow learned parameters to be associated and connected between successive stages using the method discussed in section 3.3.

In the case of image data, a choice for mapping $\mathcal{F}$ is the sub-sampling operation and a stage-wise evolution of the input image can be obtained through sub-sampling the original input samples with an increasing ratio. Therefore, at the early stages of training, the network is focused on capturing coarse-scale image characteristics (i.e., *shape* features) through considering a wider context around each pixel. As we move forward toward the final stages, given more detailed information, the learned feature extractors are fine-tuned to detect discriminative information in the fine-scale image structures (i.e., *appearance* features).

4. Experiments

We first conduct a set of experiments to understand how the proposed stage-wise training framework for multi-layer neural network improves feature extraction. Then, we examine the performance of a multi-stage trained network for image classification compared to an equivalent single-stage trained counterpart.

For the experiments the standard CIFAR10 dataset (Krizhevsky and Hinton (2009)) is used. This dataset consists of 32×32 colored images of ten classes of objects, each having 50000 training samples and 10000 test samples. For the purpose of information evolution, each training image is sub-sampled with 5 different increasing ratios (i.e., $S = 5$)[3].

For the stage-wise training, the architecture of the network is kept unchanged during training stages. We use a convolutional neural network (CNN) with two feature extraction layers (i.e., hidden layers) before the classification layer. The hyper-parameters of this CNN are similar to that of (Hinton et al. (2012)) for CIFAR10: 64 filters of size 5×5 at each convolutional layer followed by max-rectifying nonlinearity and pooling of size 3×3 with stride 2. The max and average pooling functions are used for the first and second layer, respectively.

4.1 Analysis

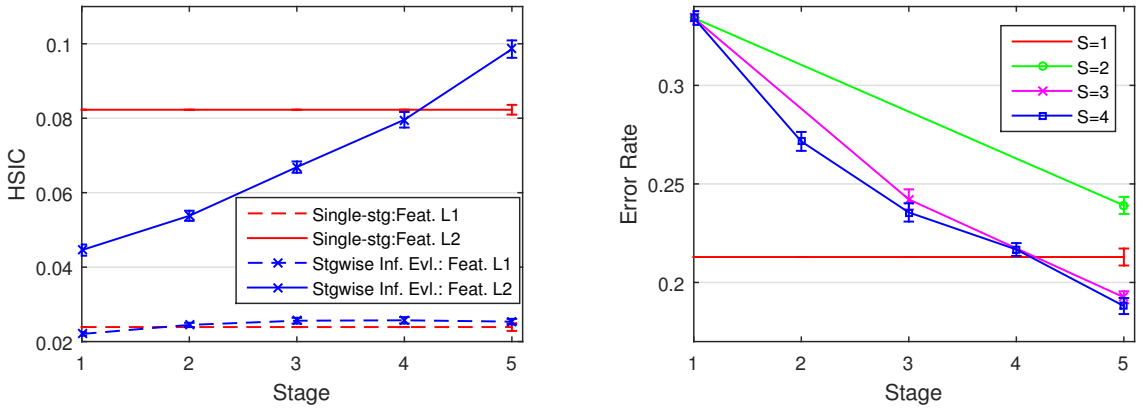
Given the fact that different input data is used at different stages, we study how the classification problem solved at each training stage is related to that of the other stages.

Table 1 shows the connection between the learned models at different stages with evolved inputs. In this table, entry at row i and column j refers the to "what ratio of the test samples that are correctly classified (with high confidence) at stage i are classified correctly (with high confidence) at stage j?"[4]. Considering the entries above the main diagonal, most (i.e., more than %92) of the samples that are recognized confidently at each scale are also classified correctly in the subsequent stages. The entries below the main diagonal show that the learned network at each stage is an improved version of the learned model at its previous stage. This result is interesting in the sense that, despite the difference in the size of the input data at each stage and consequently, different width of the context used for

3. The size of each input training image at stages 1, 2, ... , 5 are 14×14, 16×16, 20×20, 23×23 and 32×32, respectively.
4. The confidence of a prediction comes from the output of the soft-max layer for the predicted class.

		Confidently Recog. at				
		Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Given that Confidently Recog. at	Stage 1	1	0.957±0.002	0.956±0.003	0.957±0.003	0.954±0.003
	Stage 2	0.608±0.004	1	0.953±0.003	0.949±0.002	0.938±0.003
	Stage 3	0.477±0.004	0.748±0.004	1	0.950±0.002	0.930±0.004
	Stage 4	0.426±0.004	0.665±0.004	0.848±0.003	1	0.929±0.004
	Stage 5	0.386±0.003	0.597±0.005	0.754±0.003	0.844±0.003	1

Table 1: Connection between the learned models at different stages with evolved inputs. The entry at row i and column j indicates "what ratio of the test samples that are correctly classified (with high confidence) at stage i, are classified correctly (with high confidence) at stage j?". Note that despite the difference in the input data, the models learned at successive stages are closely connected.

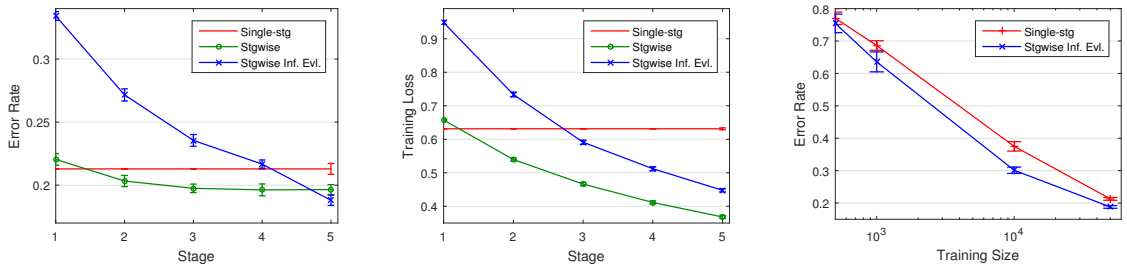


(a) (b)

Figure 2: (a) Dependency of the extracted features at each layer of the network to the true class label across different stages of training. Observe how this dependency increases during stage-wise training for the extracted features at both layers of the network. (b) Performance of the learned features as a function of the level of information presented to the network during stage-wise training. Notice how the gradual information evolution during multiple training stages leads to a smaller error rate.

feature extraction, there is a close relationship between the problems that we are solving at successive stages.

In the next experiment, we factorize the effect of the learned feature extractors from the classifier. In other words, we directly compare the statistical dependence between the learned features (at each layer of the network) and the output variable for different training strategies. In order to measure this dependence, we use the Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al. (2005)). Assume that we are given n samples belonging to c different classes. Each sample is represented by a d-dimensional vector, which are the extracted features at a specific layer of the network. Using HSIC, we can compute the dependence between the extracted features $Z_{d\times n}$ and their corresponding labels $Y_{c\times n}$ as



(a) (b) (c)

Figure 3: Classification performance of three methods (i.e., single-stage, stage-wise without information evolution and stage-wise with information evolution) in terms of (a) test error (b) training loss. Observe the superior performance of the stage-wise training strategy on test data despite the better performance of its counterparts on the training loss, which demonstrates the regularization effect of stage-wise training. Also, figure (c) shows that the gap between the performance of the stage-wise framework is more significant for smaller training sizes.

$$HSIC(Z,Y) := (n-1)^{-2}\mathbf{tr}(HKHL) \tag{5}$$

where K is a kernel of Z (e.g., Z^TZ), L is a kernel of Y and $H = I - n^{-1}\mathbf{e}\mathbf{e}^T$ (the centering matrix)[5]. In this experiment we use a linear kernel for both Z and Y and normalize HSIC to have a maximum value of 1 (i.e., multiplying (5) by $\frac{(n-1)^2}{\|K\|_F\|L\|_F}$). Figure 2a shows the computed HSIC for the extracted features from the test set using the trained network at different stages. This figure indicates that during stage-wise training, the dependence between the learned features and the class labels increases. Moreover, compared to single-stage training, stage-wise training of the network improves the quality of the extracted representations at all of the network layers in terms of their dependence to the class labels.

Finally, we study the effect of the *speed* of the information evolution on the performance of the learned features. We wonder how many training stages is required for a certain amount of information evolution to take place. As explained in the beginning of this section, each input image is represented in 5 different levels of details such that level 5 corresponds to the original input representation. In this experiment we compare and contrast three different training scenarios: (i) Single-stage training using the entire information (e.g., level 5) (ii) Stage-wise training with a high speed of information evolution (e.g. jumping from stage 1 to 5) (iii) Stage-wise training with a slow speed of information evolution (e.g. stage 1 to 2, ...,4 to 5). As figure 2b shows, it is important to increase the amount of training information fed to the network gradually and during multiple stages.

4.2 Classification Results

In this section, we evaluate the classification performance of the learned representations using the proposed stage-wise training framework. Three different training strategies are

5. $\mathbf{e}$ is a vector of all ones.

compared: 1) conventional single-stage training 2) stage-wise training without information evolution (i.e., the level of details in training images are the same for all stages) 3) stage-wise training with information evolution (i.e., the level of details in training images increases at successive stages). The architecture of the trained networks using all of these three methods are the same and is as discussed at the beginning of this section. Figure 3a demonstrates the superior performance of the stage-wise training strategy on test data classification compared to the conventional single-stage training. Furthermore, considering the training loss of these methods shown in figure 3b, evolution of information during stage-wise training acts as a regularizer and improves the generalization of the learned features. Note that these results are obtained using a simple base-line two-layer network and can be applied to the state of the art models without loss of generality.

A good regularizer should improve the generalization performance of the learned model on small data sets. Considering this fact, we explore the performance of the model as a function of the number of training samples. The result of this experiment is depicted in figure 3c. It can be observed that the gap between the performance of the stage-wise model and its single-stage counterpart is more significant for small training sets, confirming the fact that it is more important to regularize well for small datasets.

5. Conclusion

In this paper we proposed a stage-wise training framework with information evolution for feature extraction using deep neural networks. In this framework, the network is steered towards a good solution through a sequence of related learning stages, where the amount of information provided to the network is gradually increased during these stages. More specifically, at the early stages only coarse-scale properties were provided to the network; then, using the solution found at the previous stage, as a prior for the next learning stage, fine-scale learning takes place at the successive stages. In principle, stage-wise training acts as a regularizer.

Moreover, we showed that the classifier layer of the network requires a special treatment. In fact, due to the problem of diffusion of gradient in deep models, the classifier can overfit while the early layers are not still learned. This problem is alleviated in a stage-wise framework, where the feature extraction layers are initialized using the previous stage while the classifier layer is randomly initialized at the beginning of each stage .

The experimental analysis showed that the proposed framework improves the accuracy of image classification for CIFAR10 data set. The most important reason for such an improvement is the regularization effect of the stage-wise training. In addition, it was shown that the statistical dependence between the learned features and the class labels increases stage-by-stage, eventually becoming larger than that of single-stage training. This shows that the extracted features using stage-wise model are more discriminative. Nevertheless, we believe that the real power of the proposed method emerges on higher-dimensional inputs (e.g., larger images), the subject of future work.

References

Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28. Association for Computational Linguistics, 2012.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. *Unsupervised and Transfer Learning Challenges in Machine Learning*, 7:19, 2012.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

Rich Caruana. Learning many related tasks at the same time with backpropagation. *Advances in neural information processing systems*, pages 657–664, 1995.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603. IEEE, 2013.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.

Mehmet Gönen. Coupled dimensionality reduction and classification for supervised and semi-supervised multilabel learning. *Pattern recognition letters*, 38:132–141, 2014.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic learning theory*, pages 63–77. Springer, 2005.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 1(4):7, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

Tomáš Mikolov. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 2012.

Mehryar Mohri, Afshin Rostamizadeh, and Dmitry Storcheus. Foundations of coupled nonlinear dimensionality reduction. *arXiv preprint arXiv:1509.08880v2*, 2015.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *arXiv preprint arXiv:1412.1897*, 2014.

Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Andrey Nikolayevich Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198, 1943.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.