How Should Intelligence Be Abstracted in AI Research ...
AAAI Technical Report FS-13-02

Sports Video Classification from Multimodal Information Using Deep Neural Networks

Devendra Singh Sachan, Umesh Tekwani, Amit Sethi

Dept of Electronics and Electrical Engineering, IIT Guwahati, India
sachan.devendra@gmail.com, umesh.tekwani@gmail.com, amitsethi@iitg.ernet.in

Abstract

The work presents a methodology for classification of sports videos using both audio and visual information by applying deep learning algorithms. We show a methodology to combine multiple deep learning architectures through higher layers. Our method learns two separate models trained on audio and visual part of the data. We have trained the model for the audio part of the multimedia input using two stacked layers of CRBMs forming a CDBN. We also train two layered ISA network to extract features from video part of the data. We then train deep stacked autoencoder over both audio and visual features with discriminative fine tuning. Our results show that by combining both audio and visual features we get better accuracy as compared to single type of features.

Introduction

In machine learning the algorithms learn from training data and predict the outcome for a new test case. Machine learning algorithms are applied in various fields such as object recognition from image, search engines, spam classification, voice recognition and document indexing. Mostly people have used hand-engineered features for training a particular learning algorithm, but designing hand-engineered features requires a lot of domain expertise and it does not scale well to other applications. Another problem with learning algorithms is that they require large amounts of supervised labeled data to perform well which is very difficult and expensive to obtain.

In order to address the above problems, in our work we have used unsupervised feature learning algorithms (RBM, ISA, sparse deep autoencoders). These feature learning algorithms are directly applied on raw data with little preprocessing and are also applicable to other domains. To learn rich representations, we only need large amounts of unlabelled data which is easy to obtain and is freely available on the web.

Feature extraction using unsupervised learning and building deep networks have become very popular in the areas of object recognition, action recognition, speech and speaker classification, natural language processing(NLP), scene labeling etc. But most of these tasks use only one mode of information such as image content, speech or text data. Recently, there has been work done by Srivastava et al(2012) which uses both text and image features. There are various applications of unsupervised feature learning algorithms of which our problem is to classify sport video events using both video and audio information. We show that by using both video and audio data for the task of sports action classification we get much better results as compared to using only single type of information.

To model the audio data we used deep convolutional RBM in which the lower layers encode simple features while higher level layers encode more complex features. Similarly, we use stacked ISA network to extract features from visual data. Finally we jointly train audio and visual features using deep stacked sparse autoencoders with discriminative fine tuning.

Previous Work

Raina et al(2007) were able to show that using unrelated training examples increase the performance of recognition accuracy in a framework which they call self taught learning. It was shown by Honglak Lee et al(2009) that a good image recognition accuracy on Caltech 101 dataset can be achieved using only features learnt from natural images using CDBN network and they show that they are able to learn a hierarchical representation of objects. Using almost the same architecture of CDBN as above Lee et al(2009) were able to achieve a high speech and speaker recognition accuracy in case of TIMIT database.

Quoc Le et al(2011) were able to obtain state of the art results in action recognition in videos using sparsity on ISA network. They show that using both 1st layer and 2nd layer features boosts the classification accuracy. But in doing so they used only the video content features.

There has been work done by Srivastava et al(2012) which proposes DBM for learning multimodal representations of data. They pretrained deep network for each separate layer and then fused them for joint training. Ngiam(2011) et al also proposed deep learning architectures for multimodal (audio and video) classification tasks.

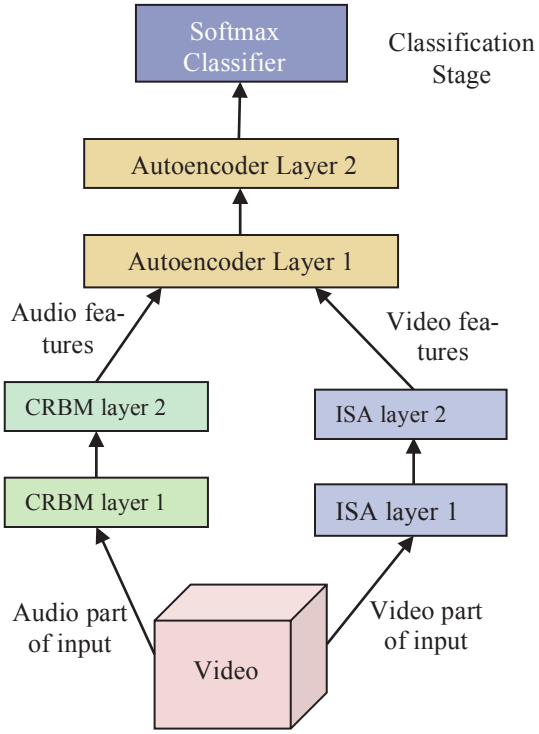

Fig 1. Feature extraction pipeline of our approach

System Architecture

Our deep learning architecture has three major blocks as shown in Fig 1. First data is separated into its audio and visual components respectively. The first block extracts features from audio part of the data, second block extracts features from video part of the data while the third block gets trained over the features extracted from both audio and visual part. Then we use softmax classifier with discriminative finetuning to report the results.

Feature extraction from audio data

Restricted Boltzmann Machine (RBM)

Restricted Boltzmann Machine commonly called RBM is bipartite, undirected graphical model. They can also be viewed as Markov Random Field (MRF) with connections only between visible and hidden layers. In this, each visible unit is connected to all the hidden units while there are no intra connections between visible and hidden units. The figure below shows the hidden units and visible units and the weights connected between these units.

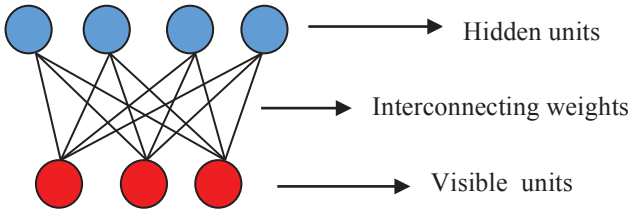

Fig 2. Restricted Boltzmann Machine (RBM) Model

Mathematically, the energy function of a binary-binary RBM is defined as(Honglak Lee et al 2009)

$$E(v,h) = -\left(\sum_{i=1}^{V}\sum_{j=1}^{H} v_i w_{ij} h_j + \sum_{i=1}^{V} v_i c_i + \sum_{j=1}^{H} h_j b_j \right) \quad (1)$$

where

- ' v ' is the state of visible units
- ' h ' is the state of hidden units
- ' w_{ij} ' is the weight connecting i^{th} visible unit and j^{th} hidden unit.
- ' c ' is the bias of visible unit
- ' b ' is the bias of hidden unit

The visible and hidden units can be modeled as either binary or real valued Gaussian random variable. When visible or hidden units are binary random variables they can take either of two states '0' or '1'. The probability assigned to a joint configuration of visible and hidden units is (Sohn et al 2011)

$$P(v,h) = \frac{e^{-E(v,h)}}{Z}, \text{ where } Z = \sum_{i=1}^{V}\sum_{j=1}^{H} e^{-E(v_i,h_j)} \quad (2)$$

Z is called 'partition function' .

Conditional probability of $P(h/v)$ when hidden units are binary is found out to be

$$P(h_j = 1/v) = sigmoid\left(\sum_{i=1}^{V} w_{ij} v_i + b_j \right) \quad (3)$$

From the expression we see that the hidden random variables when conditioned on visible random variables are independent of each other. Similarly, conditional probability of $P(v/h)$ when visible units are binary can be derived as

$$P(v_i = 1/h) = sigmoid\left(\sum_{j=1}^{H} w_{ij} h_j + c_i \right), \text{ where} \quad (4)$$

$$sigmoid(x) = \left(1/(1+e^{-x})\right)$$

For real-valued visible units and binary hidden units the energy function and conditional probabilities are defined as

$$E(v,h) = -\left(\frac{1}{\sigma} \sum_{i=1}^{V} \sum_{j=1}^{H} v_i w_{ij} h_j + \left(\frac{-1}{2\sigma^2} \right) \sum_{i=1}^{V} (v_i - c_i)^2 + \sum_{j=1}^{H} h_j b_j \right) \quad (5)$$

where 'σ' = variance of visible units

$$P(h_j = 1 / v) = sigmoid\left(\frac{1}{\sigma} \sum_{i=1}^{V} w_{ij} v_i + b_j \right) \quad (6)$$

$$P(v_i / h) = N\left(\sigma \sum_{j=1}^{H} w_{ij} h_j + c_i, \sigma^2 \right), \quad (7)$$

where N is Normal Distribution.

We use the above defined equations of conditional probability and energy function in training of the RBM model to learn the weights and biases.

Training of RBMs

In order to train the model the goal is to maximize log-likelihood of training data. The training involves updating of weights, visible and hidden biases. The optimization function becomes (Krizhevisky et al 2009)

$$\text{minimize } -\sum_{k=1}^{m} \log(P(v^{(k)})) \quad (8)$$

where 'm' is the total number of input examples.

Instead of applying gradient descent to this problem which is very computationally expensive, we use contrastive divergence learning algorithm which is an approximation of the gradient of the log- likelihood (Hinton 2010). Hence the update equations for weights and biases are as follows:

$$\frac{\partial}{\partial w_{ij}} \left(-\sum_{k=1}^{m} \log(P(v^{(k)})) \right) \approx (< v_i h_j >_{data} - < v_i h_j >_{recon}) \quad (9)$$

where $< . >$ represents expectation with respect to data and reconstructed data respectively. $< v_i h_j >_{data}$ is the product of visible units (input data) and hidden units. $< v_i h_j >_{recon}$ is calculated via contrastive divergence approximation (Hinton 2010). This starts by setting the input as states of visible units then compute the hidden units (after doing Gibb's sampling on equation (6)). Then a reconstruction is produced using the hidden units using equation (7). We get visible units from the reconstruction from which we again compute the hidden units (doing Gibb's sampling on equation (6)). The product of reconstructed visible units and the recomputed hidden units is $< v_i h_j >_{recon}$. The update equation for weights and biases are as follows:

$$\Delta w_{ij} = \varepsilon_w (< v_i h_j >_{data} - < v_i h_j >_{recon}) \quad (10)$$

$$\Delta b_j = \varepsilon_b (< h_j >_{data} - < h_j >_{recon}) \quad (11)$$

$$\Delta c_i = \varepsilon_c (< v_i >_{data} - < v_i >_{recon}) \quad (12)$$

where 'ε' is a learning rate constant.

It has been found that in order to learn rich representations of input the activations have to be overcomplete and sparse. The hidden layer activations in the RBM model are overcomplete in a sense that their dimensions are more than the input dimensions. Hence we introduce sparsity in the RBM model in the next section.

Sparse RBMs

In sparse RBM's the activation of each hidden unit is constrained close to a constant. By doing this it is ensured that the activation of each hidden unit is kept close to a constant. The motivation behind inducing sparsity is two-fold. First, in our brain when particular information is perceived only a few neurons are fired among the large number of neurons it contains (Hawkins 2004). Secondly it is worth to interpret information from hidden units which are active only for a small number of examples compared to the units which are active for a large number of examples.

So, after introducing a regularization term for sparsity our modified objective function becomes (Lee et al 2008):

$$\text{minimize } J = -\sum_{k=1}^{m} \log(P(v^{(k)})) + \lambda \sum_{j=i}^{H} \left| \rho - \pi_j \right|^2 \quad (13)$$

where

λ is regularization constant, ρ is target sparsity.

$$\pi_j = \frac{1}{m} \sum_{k=1}^{m} P(h_j = 1 / v) \quad (14)$$

Following Lee et al (2009), (2008), only hidden biases are updated using the gradient of the regularization term. The modified equation for updating the hidden bias becomes:

$$\Delta b_j = \varepsilon_b (< h_j >_{data} - < h_j >_{recon}) + h\_bias(\rho - \pi_j) \quad (15)$$

As the input dimension increases the size of weight matrix along with the hidden units increase. This makes RBM algorithm computationally intractable on small machines. Also it has been observed that using larger weights is not a good practice to learn features (Lee 2009). So we use locally connected weights in order to learn overcomplete features. For this we use convolutional RBMs.

Convolutional RBMs(CRBMs)

If the input data is very high dimensional then connecting every hidden unit to every visible unit becomes computationally intractable for a normal processor with not very high memory. To scale up RBMs to high dimensional inputs we use distributed convolutional RBMs. In CRBMs we have K groups of hidden units. For each group of hidden units we have a weight matrix which gets convolved with the visible layer to give the feature representation of that particular hidden group. In detail, following Lee et al 2009, 2011, if the dimensions of input are as ($N_v * N_v$) and number of input channels are L, the size of a weight filter ($N_w * N_w$) for each channel and K groups. So the weight matrix is

$W \in R^{N_w * N_w * L * K}$

In the same way as RBMs the energy function and conditional probability equations for CRBMs can be defined as:

$$E(v,h) = \frac{1}{2\sigma^2}\sum_{l=1}^{L}\sum_{i,j}\left(v_{i,j}^{l} - c_l\right)^2 - \sum_{k=1}^{K}\sum_{i,j} h_{i,j}^{k}\left(\sum_{l=1}^{L}\frac{1}{\sigma}\left(W_f^{k,l} * v^l\right)_{i,j} + b_k\right) \quad (16)$$

$$P(v^l \mid h) = N\left(\sigma\sum_{k=1}^{K} W^{k,l} * h^k + c_l, \sigma^2\right) \quad (17)$$

$$P(h_{i,j}^{k} = 1 \mid v) = sigmoid\left(\sum_{l=1}^{L}\frac{1}{\sigma}\left(W_f^{k,l} * v^l\right)_{i,j} + b_k\right) \quad (18)$$

where W_f is the left-right and top-down flipped W

The training of CRBM is done same as RBMs using contrastive divergence learning algorithm. Also sparsity is introduced in the model so that the activation of hidden units is sparse. Another important point is that a weight regularization term is introduced in the objective function in order to decrease over-fitting. We also apply probabilistic max-pooling on the hidden activations to reduce the number of output feature dimension, retaining all the useful information.

Probabilistic Max-Pooling

Following Lee et al. 2009, the hidden units in a basis are partitioned into small groups of size say $N_C * N_C$ and their activations are constrained so that only one hidden unit is active in a group. Above the hidden layer, there is a pooling layer in which a pooling unit corresponds to a particular group in the hidden unit. The pooling unit is 'on' only when one of the hidden units is 'on' else it remains 'off'. The advantages of pooling are that it encodes small translation invariance and it also reduces the dimensionality of the features. So, the conditional probability of each hidden unit is modified as

$$P(h_{i,j}^{k} = 1 \mid v) = \frac{\exp\left(\sum_{l=1}^{L}\frac{1}{\sigma}\left(W_f^{k,l} * v^l\right)_{i,j} + b_k\right)}{1 + \sum_{i',j' \varepsilon group}\exp\left(\sum_{l=1}^{L}\frac{1}{\sigma}\left(W_f^{k,l} * v^l\right)_{i',j'} + b_k\right)} \quad (19)$$

$$P(p_{group}^{k} = 0 \mid v) = \frac{1}{1 + \sum_{i',j' \varepsilon group}\exp\left(\sum_{l=1}^{L}\frac{1}{\sigma}\left(W_f^{k,l} * v^l\right)_{i',j'} + b_k\right)} \quad (20)$$

Where ' p^k ' corresponds to pooling unit of a group. From the above equations, it is clear that a group behaves as a 'softmax' random variable.

Extending CRBMs to Convolutinal Deep Belief Networks

In order to form a hierarchical representation of inputs, the CRBM layers can be stacked one above another to make a deep network. This deep network can be trained greedily one layer at a time (Hinton 2006). The pooling layer activations are given as inputs to the layer above it. The visible layer biases are kept constant during training and only the hidden layer biases are updated. For subsequent layers the hidden layer biases of the lower layer become the visible layer biases of the above visible layer.

3. Feature extraction from video data

Independent Subspace Analysis (ISA)

ISA can be used to extract low-level visual features like those present in the simple cells of V1 area of visual cortex (Quoc Le 2011, Hyvarinen 2011). ISA model consist of a visible layer, a hidden layer and a pooling layer. Let each spatio-temporal patch be of dimension (visible layer) $x^{(i)} \in R^d$ and the dimension of hidden layer is also same. The dimension of our weight matrix is $W \in R^{d*d}$, such that each row of the weight matrix corresponds to the weights by which one hidden unit is connected to visible layer. Activation of each hidden unit is given by $h_j = \left(W^j \odot x^{(i)}\right)^2$ where $\odot$ represents dot-product between two vectors.

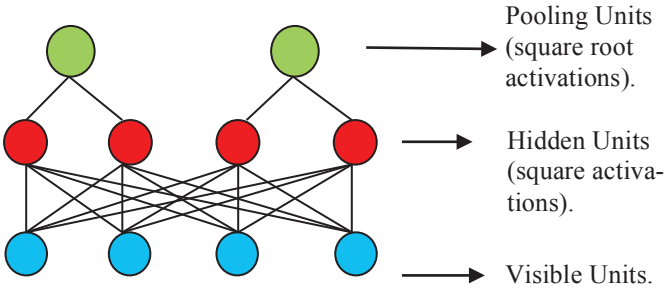

Fig 3. ISA Architecture

Each pooling unit pools over ' N_C ' adjacent hidden units and the activation of each pooling unit is given by

$$p_k = \sqrt{\sum_{j=1}^{N_C} h_j^k}$$

In order to train the ISA network, the activations of the pooling layer units over all the training examples is minimized to make them sparse. So, the objective function becomes

$$J = \text{minimize} \sum_{i=1}^{m}\sum_{k=1}^{K} p_k^{(i)} \text{ subjected to } WW^T = I$$

where 'K' is total number of pooling units, 'm' is total number of patches. Orthogonality condition is imposed on the weights to make the learned weights diverse(Quoc Le 2011). Above equation is solved by doing batch projected gradient descent on J. The weight updating equation is as below

$$\Delta w_p^j = -\alpha \frac{\partial J}{\partial w_p^j}.$$

The weight updating step is followed by the symmetric orthogonalization step which is by (Hyvarinen 2009)

$$W := \left(WW^T\right)^{-1/2} W$$

The inverse square root $\left(WW^T\right)^{-1/2}$ is obtained by the eigenvalue decomposition of

$WW^T = Vdiag(d_1, d_2, .., d_m)V^T$ as

$$\left(WW^T\right)^{-1/2} = Vdiag(d_1^{-1/2}, d_2^{-1/2}, .., d_m^{-1/2})V^T$$

In order to learn high-level features from videos we stack one layer of ISA above another layer and these layers are trained greedily one by one (*Quoc Le 2011). First layer of ISA is trained as mentioned above and to train the second layer, larger patches are extracted from videos. These patches are given as input to first layer in a convolutional manner and then activations of the first layer pooling units are computed. After applying PCA on these activations, its dimensions are reduced then these reduced dimensional feature vectors are given as input to the second layer of ISA. The weights of the second layer are trained in the same way as weights of the first layer.

Visualization of 1st layer learned weights

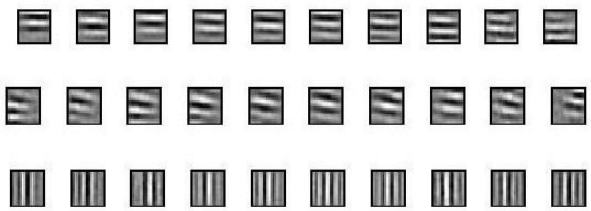

Fig 4. Each row represents the optimal stimuli for one hidden unit for 10 consecutive frames

Training the multimodal network

Deep Stacked Autoencoder

We extract audio features from CDBN and apply PCA on them. Similarly we sample the videos over a dense grid and input them to stacked ISA network to extract features and then do 'k-means clustering' (Arthur 2007) on every training example followed by PCA. Then we train two layers of sparse autoencoders according to Stanford Deep Learning tutorial over both audio and visual features and stack them. Finally we train softmax classifier over the features extracted from autoencoder and do supervised finetuning by backpropagation algorithm.

Experiments

In this section we show the classification accuracy of various experiments done by us over audio, visual and audio-visual features. These experiments were done on database created by us, which we will mention along with the details our model.

Database

Data Collection

Even after our best efforts we could not find a sports video database containing both audio and visual modes. So to conduct our experiments we manually created a database of sports videos. We downloaded the videos from youtube.com and manually clipped them to separate them into individual actions. We created four classes of videos cricket, basketball, baseball and football. The video length of each file is five seconds and each file contains a particular action of that particular sport.

We downloaded videos of four different sports cricket, football, basketball and baseball. We then manually clipped these videos into the videos of length five seconds which contained a particular action. We then resized all the video frames to a common size of 320x240. We separated the audio and visual data of the files which were then trained on their respective models. We have 146 clips for baseball, 188 for basketball, 106 for cricket and 115 for football.

Details of Our Model

For the training the audio data, the model for our first layer of CRBM has input of size 500x80 spectrogram (500 frames and 80 channels). The filter size for convolution is 6x80 and the model has 300 such filters. Pooling is done across every 3 hidden units. The inputs to second layer are the pooled activations of each audio file which are of size 165x300. The filter size for convolution is 6x300 and the model has 300 such filters. Pooling is done across every 3 hidden units. The pooled activations are of size 53x300. Hence for each input file the two-layer CDBN learns 53x300 features. For the training the visual data, the model for our first layer of ISA has input of size 16x16x10 spatio-temporal patches on which we apply PCA to reduce them to 300 dimensions. The model learns 300 feature. The inputs to second layer are the activations of each spatio-temporal patch of size 20x20x14. The model learns 200 features. Pooling is done across every 2 hidden units. The pooled activations are of size 100. Hence for each spatio-

temporal patch of size 20x20x14 the two-layer ISA learns 100 features. For our five second video file we get 1920 such spatio-temporal features and hence we get 1920x100 features.

For jointly training the audio and visual features we use deep stacked autoencoders. For each input example we have 53x300 audio features on which we apply PCA to reduce them to 500 features. Similarly for each input example we have 1920x100 visual features on which we do k-means clustering to reduce them to 300x100 features after which we apply PCA to further reduce them to 500 features. We concatenate these features to form a 1000 dimensional feature for each input example which becomes the input for sparse autoencoders.

Results

Table 1*.The results in percentage for audio and video only features for different number of training examples. (without finetuning).*

Second Layer features used	10 training examples	20 training examples	40 training examples
Audio only	83.68	87.57	92.65
Video only	82.13	83.15	88.86

Table 2*. The results in percentage for different number of features in stacked deep autoencoders.*

No. of Third Layer features	10 training examples	20 training examples	40 training examples
500	83.68	**96.97**	**97.22**
1000	**86.34**	94.60	96.71

Conclusions

As we have noted, when our network was deep and more number of examples were used we got better results, but we think that we need a larger network and more computational power to match the human visual and auditory system of brain. We see that stacked autoencoders gives relatively small improvement in classification accuracy for large training examples. This shows that deep networks are important but only stacking of different deep architectures may not be the best way to form a deep neural network.

Acknowledgements

We thank Ian Goodfellow (LISA Lab), Pawan Kumar (IIT-G) for their helpful comments and discussions in carrying out the experiments.

References

Arthur, D., Vassilvitskii, S., k-means++: The advantages of careful seeding, *SODA* 2007

Goodfellow, I., Le, Q., Saxe, A., Lee, H., Ng, A.Y., – Measuring invariances in deep networks, NIPS 2010

Hinton, G., A tutorial for training Restricted Boltzmann Machines, 2010

Hinton, G.E., Osindero, S., The, Y.W., – A fast learning algorithm for deep belief nets, *Neural Computation* 2006.

Hinton, G.E., Salakhutdinov, R.R, Reducing the dimensionality of data with neural networks, *Science* 2006.

Hyvarinen, A., Hurri, J., Hoyer, P., Natural image statistics, Springer 2009.

Jeff Hawkins , On Intelligence. 2004

Krizhevisky, A., Learning Multiple Layers of Features from Tiny images, *MSc Thesis*, 2009

Le, Q.V., Zou, W.Y., Yeung S., Ng A.Y., Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis*, CVPR*, 2011

Lee, H., Ekanadham, C., Ng, A.Y., Sparse deep belief net model for visual area V2, *NIPS* 2008.

Lee, H., Grosse, R., Ranganathan, R., and Ng, A.Y., Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, *ICML* 2009.

Lee, H., Largman, Y.,Pham, P., Ng, A.Y., – Unsupervised feature learning for audio classification using convolutional Deep Belief Networks, *NIPS* 2009.

Muja, M., Lowe, D.G., Fast approximate nearest neighbors with automatic algorithm configuration, *VISAPP* 2009

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., Ng, A.Y., Multimodal Deep Learning*, ICML* 2011.

Raina, R., Battle, A., Lee, H., Packer, V., Ng, A.Y., Self-taught learning: Transfer learning from unlabeled data, *ICML* 2007.

Sohn, K., Jung, D,Y., Lee, H., Hero, A.O., Efficient learning of sparse, distributed, convolutional feature representations for object recognition, *ICCV* 2011.

Srivastava, N., Salakhutdinov, R., Multimodal Learning with Deep Boltzmann Machines, *NIPS* 2012

ufldl.stanford.edu – Deep Learning Tutorial