Logic Extraction From Deep Belief Network

Son N. Tran abdz481@city.ac.uk
Department of Computing, City University London, EC1V 0HB, UK

Artur d'Avila Garcez aag@soi.city.ac.uk
Department of Computing, City University London, EC1V 0HB, UK

Abstract

Deep architectures have been studied intensively recently thanks to their applicability to different areas of Machine Learning Considerable success has been achieved in a number of application areas, notably in image, video and multimodal classification and retrieval tasks. We argue that knowledge extraction from deep networks can provide further improvements in specific areas and a wider applicability of such networks through a better understanding and explanation of the relations between network layers. In this paper we focus attention, therefore, on the problem of knowledge extraction from deep architectures. We propose an approach to extract logical representations from Deep Belief Networks, presenting a new extraction algorithm and initial experimental results. The extraction algorithm was implemented in C++ and applied to the handwritten digit recognition problem, and is available upon request. To the best of our knowledge, this is the first knowledge extraction system for deep networks. We also propose a method, based on knowledge extraction, to help analyse how representations are built in the different layers of the network.

1. Introduction

Recent publications indicate the emergence of two interesting sub-areas of Machine Learning: deep networks (Lee et al., 2009b), (Lee et al., 2009a) and neural-symbolic systems (Borges et al., 2011), (Aha et al., 2010). While the former have been shown capable of learning and representating features effectively in different application areas, the latter has successfully proved a strong relation between logic and connectionist systems. Deep architectures have been studied intensively recently thanks to their applicability to different areas of Machine Learning (Bengio, 2009). Considerable success has been achieved in a number of application areas, notably in image, video and multimodal classification and retrieval tasks (Ji et al., 2010). Following a neural-symbolic approach, we argue that knowledge extraction from deep networks can provide further improvements in specific problems and a wider applicability of such networks through a better understanding and explanation of the relations between network layers. In this paper we focus attention, therefore, on the problem of knowledge extraction (d'Avila Garcez et al., 2001) from deep architectures. We propose an approach to extract logical representations from Deep Belief Networks (DBN), presenting a new extraction algorithm and initial experimental results. The extraction algorithm was implemented in C++ and applied to the handwritten digit recognition problem, and is available upon request. To the best of our knowledge, this is the first knowledge extraction system for deep networks. The approach is inspired by and extends the idea of penalty logic (Pinkas, 1995), which was applicable originally to shallow symetrical networks, and applied in modified form recently to recurrent temporal restricted Boltzmann machines (de Penning et al., 2011). We also propose a method, based on knowledge extraction, to help analyse how representations are built in the different layers of the network. We expect knowledge extraction to serve as a tool in the study of the depth of DBNs and the relationship between network depth and performance.

In a nutshell, this paper shows that, given a DBN, each pair of layers can be sampled in generative mode producing logical rules of the form $A \rightarrow B$, each rule having a *confidence* value (defined in what follows). Rules can be chained down all the way to the visible layer so that the extracted rule set can be compared with the DBN in terms, for example, of the images that they generate (experiments section below). The rules can be inspected or queried taking into consideration any subset of the set of layers, hopefully sheding light into the structure of the network and the role of each layer/set of nodes. This will be visualized in the case of a handwritten digit case study. Also, the chaining between layers can be analysed more explicitly with the use of the rules' confidence levels.

The remainder of the paper is organised as follows. In Section 2, we introduce the relevant background on DBNs and penalty logic. In Section 3, we introduce the extraction algorithm. In Section 4, we discuss the experimental results, and in Section 5 we conclude and discuss directions for future work.

2. Background

In this section, we define the notation used in the paper and briefly recall the constructs of DBNs and penalty logic used later by the extraction algorithm.

A DBN is constructed by stacking several restricted Boltzmann machines (RBMs) (Smolensky, 1986) for the purpose of cancelling out the "explaining away" phenomenon. Theoretically, this can be done by adding complementary prior probabilities so that the joint distribution will lead to a factorial posterior (Hinton et al., 2006). In a two-layered graphical model, if the joint distribution is of the form $P(x,y) = \frac{1}{C}\exp(\sum_{i,j}\Psi_{i,j}(x_i,y_j) + \sum_i \gamma_i(x_i) + \sum_j \alpha_j(y_j))$ then there will exist complementary priors that make:

$$P(x|y) = \prod_i P(x_i|y)$$
$$P(y|x) = \prod_j P(y_j|x)$$

The joint distribution of a RBM with visible layer v and hidden layer h, is exactly of the form above, more precisely:

$$P(v,h) = \frac{1}{Z}\exp(\sum_{i,j} v_i w_{ij} h_j + \sum_i a_i v_i + \sum_j b_j h_j).$$

From this distribution, one may want to sample a value of v by setting RBM to equilibrium state which can be achieved by running up-down Gibb samplings for long iterations (Hinton et al., 2006). This process simulates the idea of creating an infinite stack of directed RBMs with fixed weights and then, from the very top layer, performing a down-pass sampling. The units in the bottom pair of layers would represent the equilibrium state of the coressponding RBM. In practice, a DBN has an undirected RBM at the top to simulate the upper-part infinite stack and several directed RBMs underneath.
Training a DBN starts from the lowest component model upwards, each component model (a RBM) is trained in the normal way using contrastive divergence (Carreira-Perpinan & Hinton, 2005). This greedy training has been shown capable of aproximating well enough the idea of complementary priors; however, to improve performance of DBNs as a classifier an additional back-probagation training phase is normally applied (Hinton et al., 2006).
After training, each unit in the hidden layer is seen as a holder of features learned from the data. The higher a layer a hidden unit belongs to, the more concrete features can be (e.g. edges, curves, shapes in the handwritten digits case study). In (Hinton et al., 2006), the exact reconstruction of the data from labels is tested by clamping a softmax unit and performing Gibbs sampling at the top layer followed by a down-pass through the network to the visible units. This "clamping" method samples visible states using the disitribution $P(\mathbf{v}|h^l_j = 1)$. This is also used by (Erhan et al., 2009) to visualize features in the network's second hidden layer, which is a form of "extraction". In (Erhan et al., 2009), a visualization method is also introduced following the idea of finding visible states which maximize the activation of a hidden unit, $\overset{*}{\mathbf{v}} = \arg\max_{\mathbf{v}} P(h^l_j = 1|\mathbf{v})$.

In (Pinkas, 1995), symetric connectionist systems charactered by an energy function have been shown equivalent to sets of pairs of logic formulae and real numbers $\{< p_i, f_i >\}$. The real number p_i is called a "penalty" and the set of pairs of formulas is known as a *penalty logic well-formed formula* or PLOFF. We call each pair of formulas a *penalty logic literal* or PLL. A connectionist system and a rule set are equivalent if and only if there exists a ranking function V_{rank} for the latter such that: $V_{rank}(\vec{x}) = E(\vec{x}) + c$, where c is a constant and E is the energy function of the system. In the case of an RBM, this can be defined as:
$E(h,v) = -\sum_{i,j} v_i w_{ij} h_j - \sum_i a_i v_i - \sum_j b_j h_j$
with the corresponding extracted rules being:
$\{\langle w_{ij}, v_i \wedge h_j\rangle | w_{ij} > 0\} \cup \{\langle -w_{pq}, \neg(v_p \wedge h_q)\rangle | w_{pq} < 0\}$
and with ranking function the sum of the penalties of the pairs whose formulae are violated given truth-values for $\mathbf{h}, \mathbf{v}$.

3. Logic Extraction From DBN

In this paper, we extract rules from a set of RBMs and combine them into a rule set for DBNs. If we extend the above rule equivalence result from penalty logic to DBNs, we obtain the rule set below for a deep network with L hidden layers:

$\{\langle w_{ij}^0, v_i \wedge h_j^1\rangle | w_{ij}^0 > 0\} \cup$
$\{\langle -w_{pq}^0, \neg(v_p \wedge h_q^1)\rangle | w_{pq}^0 < 0\} \cup$
$\bigcup_{l=1}^{L-1} (\{\langle w_{ij}^l, h_i^l \wedge h_j^{l+1}\rangle | w_{ij}^l > 0\} \cup$
$\{\langle -w_{pq}^l, \neg(h_p^l \wedge h_q^{l+1})\rangle | w_{pq}^l < 0\})$

Taking a pair $\langle w_{ij}, v_i \wedge h_j\rangle$ with $w_{ij} > 0$, it is clear that if v_i is activated ($v_i = 1$), h_j should also be activated with "confidence" value w_{ij} (since we are interested in minimizing the energy function). Similarly, for a pair $\langle -w_{ij}, \neg(v_i \wedge h_j)\rangle$ ($w_{ij} < 0$ in this case), if v_i is activated then the energy will decrease only if h_j is not activated ($h_j = 0$) with confident value $-w_{ij}$. Therefore:

$$h_j = \begin{cases} 1 & \text{if } v_i = 1 \text{ and } w_{ij} > 0 \\ 0 & \text{if } v_i = 1 \text{ and } w_{ij} < 0 \end{cases} \qquad \text{(Eq.1)}$$

with confidence value $|w_{ij}|$.

A confidence value is the dual of the penalty in (Pinkas, 1995) and it represents how reliable (or likely to activate) a unit is when another connected unit is already activated. However, it is not easy to determine the activation state of a unit within the complete rule set because it may apprear in different PLLs whose weights are either positive or negative. For example, let $\{\langle w_{ij}, v_i \wedge h_j\rangle | w_{ij} > 0\} \cup \{\langle -w_{ij}, \neg(v_i \wedge h_j)\rangle | w_{ij} < 0\}$ be a subset of the rules containing h_j, and $Sum_j = \sum_{i,v_i=1} w_{ij}$ the sum of weights of the connections between h_j and all activated units v_i. If $Sum_j > 0$, we say that the positive rules, in total, are more likely than the negative rules; hence h_j is more likely to activate. We can check this by looking at the activation of h_j : $P(h_j = 1) = \frac{1}{1+\exp(-\sum_i w_{ij} v_i)} = \frac{1}{1+\exp(-Sum_j)}$ in which the more positive Sum_j is, the higher the propability of h_j being activated. From this function, we also see dually that if $Sum_j < 0$ then there is less confidence in the activation of h_j, that is, it is more likely not to active. Therefore, we define:

$$h_j = \begin{cases} 1 & \text{if } \sum_{i,v_i=1} w_{ij} > 0 \\ 0 & \text{if } \sum_{i,v_i=1} w_{ij} < 0 \end{cases} \qquad \text{(Eq.2)}$$

with confidence value $|\sum_{i,v_i=1} w_{ij}|$.

In general, using Eq.1 and Eq.2 one should be able to capture the relations between the units in different layers of a DBN. Our extraction algorithm applies Eq.1 and Eq.2 to determine which units should be activated when higher adjacent unit(s) are activated and then it generates rules from this. For any pair of layers other than the visible layer, it sets a hidden unit to *true* (or activated) and applies downward inference to get truth-values for all the visible units which share connections with h_j. If for convenience, we let v_i be denoted by h_i^0, the conjunction of the extracted rules can be written in simplified form as:

$$h_j^{l+1} \rightarrow (\bigwedge^{i,w_{ij}^l>0} h_i^l) \wedge (\bigwedge^{i,w_{ij}^l<0} \neg h_i^l).$$

One might assume, as done in (de Penning et al., 2011) for RT-RBMs, since the network is symmetric, that the above rule should use $\leftrightarrow$ instead of $\rightarrow$. However, in the inverse direction of the arrow, a different rule might be extracted if we were to apply a process similar to the above. More precisely, we would have obtained:

$$h_j^{l+1} \overset{Sum_j}{\leftarrow} (\bigwedge^{i,w_{ij}^l>0} h_i^l) \wedge (\bigwedge^{i,w_{ij}^l<0} \neg h_i^l),$$

where Sum_j is used on top of the arrow to highlight the need to calculate Sum_j, as defined above, before one can determine the value of h_j.

Returning to the downward process of rule extraction, notice that an inference step from layer l to layer $l-1$ now uses a conjunction of h_j^l to infer the truth-value of h_i^{l-1}, as follows:

$$h_i^{l-1} \overset{Sum_i}{\leftarrow} (\bigwedge^{j,w_{ij}^{l-1}>0} h_j^l) \wedge (\bigwedge^{j,w_{ij}^{l-1}<0} \neg h_j^l).$$

Notation: In the equations above we always use index i to denote the units in the lower layer and index j to denote the units in the higher layer. The figure below illustrates the process. In the downward direction, the inference step from layer $l+1$ to layer l is the same process from h_j to v_i, as mentioned earlier.

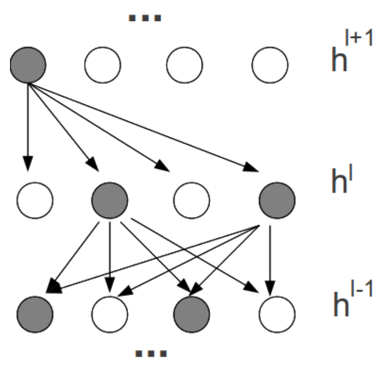


Figure 1. Down-pass inference mechanism from unit(s) in higher hidden layers. Activated units are filled in gray and connections from non-activated units to lower units are not shown.

The extracted rules obtained should allow us to explain in a formal symbolic language how a hidden unit in a DBN is responsible for reconstructing an observation. The algorithm below contains the details of this process.

Algorithm 1 Multi-layer down-pass inference

```
**Input:**
$DBN = \{\langle h^0, h^1, w^1\rangle, ..., \langle h^l, h^{l+1}, w^l\rangle ...\}$ $l = 1, .., L$
$l^*$ ( defines the higher hidden layer to start with )
$j^*$ ( sets hidden unit $l^*$ as activated to start with)
**Output:**
$\Psi$ (set of rules)
**Start:**
Initialize $\Psi = \{\}$
Initialize $h^{l^*}_{j^*} = 1$
Extract rules $\psi$ from $w^{l^*-1}$
$\Psi = \Psi \cup \psi$
Infer $h^{l^*-1}$ from $h^{l^*}_{j^*}$ using Eq.1
for $iter = 0$ to $K$ do
  Infer $h^{l^*}$ from $h^{l^*-1}$ using Eq.2
  Infer $h^{l^*-1}$ from $h^{l^*}$ using Eq.2
end for
for $l = l^* - 2$ down to $0$ do
  Extract rules $\psi$ from $w^{l^*}$
  $\Psi = \Psi \cup \psi$
  Infer $h^l$ from $h^{l^*+1}$using Eq.2
end for
return: $\Psi, h^0$
```

4. Experiments and Results

In this section, we shall empirically verify the connection between logical rules and stochastic sampling by extraction and feature visualization of rules from Deep Belief Networks. Using a $784 - 500 - 500 - 2000$ structure, we have trained the system with 5000 examples from the MNIST handwritten digit dataset. Some of the rules extracted from the deep network are exemplified below. First, the sampling from the network will extract a set of PLLs, as follows:

$$\langle 0.54, \neg(h_0^2 \wedge h_0^1)\rangle, \langle 0.76, h_0^2 \wedge h_1^1\rangle, \langle 0.52, h_0^2 \wedge h_2^1\rangle, ...$$
$$\langle 0.52, \neg(h_0^1 \wedge v_0^0)\rangle, \langle 0.68, h_0^1 \wedge v_1^0\rangle, \langle 0.53, \neg(h_0^1 \wedge v_2^0)\rangle, ...$$
$$\langle 0.52, \neg(h_1^1 \wedge v_0^0)\rangle, \langle 0.55, \neg(h_1^1 \wedge v_1^0)\rangle, \langle 0.53, \neg(h_1^1 \wedge v_2^0)\rangle, ...$$
$$\langle 0.53, \neg(h_2^1 \wedge v_0^0)\rangle, \langle 0.58, \neg(h_2^1 \wedge v_1^0)\rangle, \langle 0.53, \neg(h_2^1 \wedge v_2^0)\rangle, ...$$

Each line above contains the PLLs extracted for each hidden unit h_0^2, h_1^2, etc, and then h_1^1, h_2^1, in each layer the network. This creates a hierarchy of rules in short form, as follows:

$$0.00345 : h_0^2 \leftrightarrow \neg h_0^1 \wedge h_1^1 \wedge h_2^1 .... \wedge \neg h_{499}^0$$
$$0.00001 : h_0^1 \leftrightarrow \neg v_0^0 \wedge v_1^0 \wedge \neg v_2^0 \wedge ... \neg v_{783}^0$$
$$0.00163 : h_1^1 \leftrightarrow \neg v_0^0 \wedge \neg v_1^0 \wedge \neg v_2^0 \wedge ... \neg v_{783}^0$$
$$0.00944 : h_2^1 \leftrightarrow \neg v_0^0 \wedge \neg v_1^0 \wedge \neg v_2^0 \wedge ... \neg v_{783}^0$$

showing the likelihood between layers as determined by Eq.1 and Eq.2, where the confidence values are calculated by setting the value of the hidden unit in the higher layer to 1 and then inferring the truth-values of the units in the lower layer before using these values to infer upwards and determining the confidence value as the real-number activation of that higher-layer hidden unit, as done in (de Penning et al., 2011).

In our first experiment, after all the rules have been extracted, we have activated 25 hidden units, one at a time, in the second hidden layer by setting their state to true, and used the downward inference algorithm described above to find the states of the units in the lower hidden layer and the visible layer. Notice that each inference step (from higher hidden units to lower ones) just reduces the energy of the corresponding RBM while we are interested in finding, as mentioned above, an equilibrium state that sets the energy function to a global minimum. As a result, as usual, we run the network up-down inference mechanism a number of times, in our experiments 1000 times, producing images from the visible units at every 20 runs for monitoring. This sampling process in the network produced the images in Figure 2(a). Using logical inference in the extracted set of rules produced the images in Figure 2(b), as detailed below. The results show that very similar images were obtained from using sampling and the logical rules. This was to be expected given the relationship between the network and the rules described above and since the confidence values of the rules and the input of the network's logistic activation function when we use Gibbs sampling are the same.

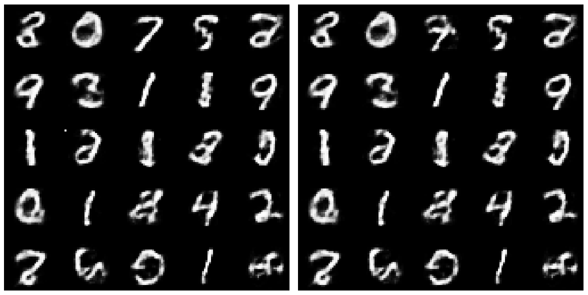

(a) (b)

Figure 2. Feature visualization by sampling the network with clamping (a), and visualization of logical inference rules (b); in this test, logistic function and stochastic activation are used.

Figure 2(a) shows the images reconstructed from 25 hidden units selected in the second hidden layer (only one unit is clamped at a time and we run Gibbs sampling 1000 times before a down-pass to the visible layer). Figure 2(b) is generated by applying logical inference on the extracted rule set with logistic function for normalizing the confidence values. This experiment shows a close relation between the extracted rules and the deep network. This confirms the strong connection between the logical rules that our algorithm produces, as discussed in Section 3, and Gibbs sampling. Logical inference within the extracted rules has been shown to approximate the Gibbs sampling process.

In the above first experiment, the confidence values were the input of the network's logistic activation function. We now consider the use of a different - less stochastic and more logical - function. We run the following experiment using only the sign of the confidence value to activate (if it is positive) or deactivate (if it is negative) a unit. We call this *sign-based activation*. This deterministic setting makes our operations on RBMs similar to applying "herding" algorithm (Welling, 2009) to Markov Random Field (without the updating step). Figure 3 shows the progression of 10 images generated (from left to right) using network samplings (top row), logical inferences with stochastic confidence values (middle row) and signed-based logical inference (bottom-row). The result shows fast convergence of inference with sign-based activation, while the other two may represent different models at several points (each point usually being a local minimum; each image was produced after intervals of 10 iterations, totalling 100 iterations in the network).

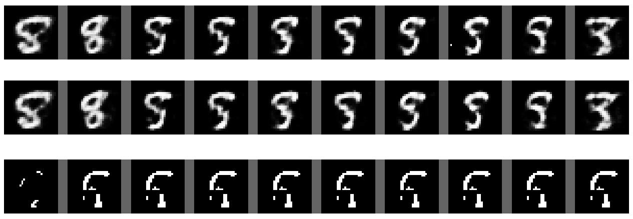
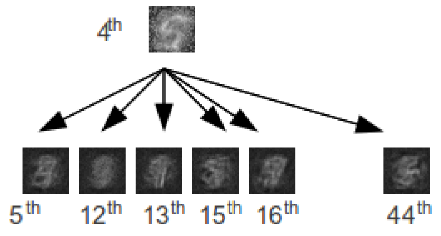
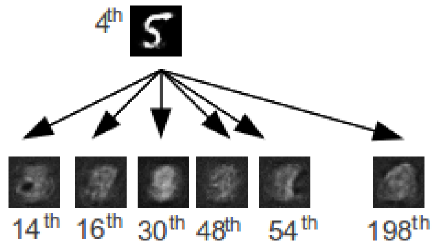


Figure 3. Images generated from network samplings (top row), logical inferences with stochastic activation/confidence values (middle row), and logical inference using signed-based activation/confidence values.

Finally, we applied the logical rules in one direction only (all previous experiments used up-down inference within the rule set to search for an equilibrium state, as discussed earlier) to test the propagation of a single activated unit in a higher-order hidden layer to other units in lower layers. We set a hidden unit to be *on* (in this experiment, h^2_4) in the second hidden layer, and inferred the truth-values of all the hidden units in the first hidden layer. We then visualized the image reconstruction when h^2_4 is activated and each of the activated units in h^1 are sellected one at a time to produce images in the visible units one layer down.



(a)



(b)

Figure 4. Features represented by units in lower hidden layer whose activation is triggered by a given unit in higher layer; the numbers 4th, 5th, etc. indicate the index of the activated unit at a layer.

The blurred picture at the top of Figure 4(a) shows that the higher unit did not activate the right feature holders at the lower layer. In order to discover which units at the first hidden layer are feature holders for the image in question (number 5 in this case), we have run up-down inference as usual to produce the result shown in Figure 4(b). Comparing Figure 4(a) with 4(b), we can see that some features are common (i.e the number 16th) and thus the right activation seems to have been made there in Figure 4(a) to contribute to the partial reconstruction of number 5. We believe that this kind of analysis can be useful in general to provide descriptions of the relative importance of features in a given application and to explain the role of the different hidden layers in a deep network.

5. Conclusion and Future Work

We have proposed a knowledge extraction method for deep belief networks. The extracted rules closely resemble the process of minimizing the energy function of RBMs, and generalize to stacks of RBMs so that logical rules can be extracted from hidden layers at any point in a deep network. The results indicate that a deep belief network can be represented by sets of rules with a logical inference algorithm that serve as an alternative to stochastic sampling. We expect that by inspecting the rules and carrying out inference at the rule-level, a better understanding of the structure and features in the deep network can be obtained.

As future work, we are interested in studying the relationships between confidence values and networks in a systematic way, at different layers to find the best way of determining the truth-value of a unit. In addition, we hope the relationships can be used to combine and simplify extracted rules into more general rules which are able to explain the network's behaviour. For example, from two rules:
$c_a : h_1 \leftrightarrow v_1 \wedge v_2 \wedge v_3$
$c_b : h_1 \leftrightarrow \neg v_1 \wedge v_2 \wedge v_3$
we want to derive a simplified rule : $c_c : h_1 \leftrightarrow v_2 \wedge v_3$
where c_c is a function of c_a and c_b.
In this combination, we may need to revise the definition of confidence-value in order to guarantee the simplified rules are equivalent to the original rules in terms of knowledge representation and inference.
We are also interested in studying how the use of alternative sampling methods might influence rule extraction, e.g. GSAT, WalkSAT (Selman et al., 1992), (Selman et al., 1994), and we would like to study further if rule extraction can be useful in practice as a general mechanism for feature extraction (Gadat & Younes, 2007), (Gorodetsky & Samoilov, 2010) and transfer learning(Rosenstein et al., 2005), (Raina et al., 2007).

References

Aha, David W., Boddy, Mark S., Bulitko, Vadim, d'Avila Garcez, Artur S., Doshi, Prashant, Edelkamp, Stefan, Geib, Christopher W., Gmytrasiewicz, Piotr J., Goldman, Robert P., Hitzler, Pascal, Isbell, Charles L., Josyula, Darsana P., Kaelbling, Leslie Pack, Kersting, Kristian, Kunda, Maithilee, Lamb, Luís C., Marthi, Bhaskara, McGreggor, Keith, Nastase, Vivi, Provan, Gregory, Raja, Anita, Ram, Ashwin, Riedl, Mark O., Russell, Stuart J., Sabharwal, Ashish, Smaus, Jan-Georg, Sukthankar, Gita, Tuyls, Karl, van der Meyden, Ron, Halevy, Alon Y., Mihalkova, Lilyana, and Natarajan, Sriraam. Reports of the aaai 2010 conference workshops. *AI Magazine*, 31(4):95–108, 2010.

Bengio, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Borges, Rafael V., d'Avila Garcez, Artur S., and Lamb, Luís C. Learning and representing temporal knowledge in recurrent networks. *IEEE Transactions on Neural Networks*, 22(12):2409–2421, 2011.

Carreira-Perpinan, M. A. and Hinton, G. E. On contrastive divergence learning. *Artificial Intelligence and Statistics*, January 2005.

d'Avila Garcez, A.S., Broda, K., and Gabbay, D.M. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence*, 125:155–207, 2001.

de Penning, Leo, d'Avila Garcez, Artur S., Lamb, Luís C., and Meyer, John-Jules Ch. A neural-symbolic cognitive agent for online learning and reasoning. In *IJCAI*, pp. 1653–1658, 2011.

Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009.

Gadat, Sébastien and Younes, Laurent. A stochastic algorithm for feature selection in pattern recognition. *J. Mach. Learn. Res.*, 8:509–547, December 2007. ISSN 1532-4435.

Gorodetsky, Vladimir and Samoilov, Vladimir. Feature extraction for machine learning: Logic-probabilistic approach. *Journal of Machine Learning Research - Proceedings Track*, 10:55–65, 2010.

Hinton, Geoffrey E., Osindero, Simon, and Teh, Yee-Whye. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comp.*, 18(7):1527–1554, July 2006.

Ji, Shuiwang, Xu, Wei, Yang, Ming, and Yu, Kai. 3d convolutional neural networks for human action recognition. In *ICML*, pp. 495–502, 2010.

Lee, Honglak, Grosse, Roger, Ranganath, Rajesh, and Ng, Andrew Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 609–616, New York, NY, USA, 2009a. ACM. ISBN 978-1-60558-516-1.

Lee, Honglak, Largman, Yan, Pham, Peter, and Ng, Andrew Y. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems 22*, pp. 1096–1104. 2009b.

Pinkas, Gadi. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, 77(2): 203 – 247, 1995. ISSN 0004-3702. doi: 10.1016/0004-3702(94)00032-V.

Raina, Rajat, Battle, Alexis, Lee, Honglak, Packer, Benjamin, and Ng, Andrew Y. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*, 2007.

Rosenstein, Michael T., Marx, Zvika, Kaelbling, Leslie Pack, and Dietterich, Thomas G. To transfer or not to transfer. In *In NIPS05 Workshop, Inductive Transfer: 10 Years Later*, 2005.

Selman, Bart, Levesque, Hector, and Mitchell, David. Gsat: A new method for solving hard satis ability problems, 1992.

Selman, Bart, Kautz, Henry A., and Cohen, Bram. Noise strategies for improving local search. In *AAAI*, pp. 337–343, 1994.

Smolensky, Paul. Information processing in dynamical systems: Foundations of harmony theory. In *In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing: Volume 1: Foundations*, pp. 194–281. MIT Press, Cambridge, 1986.

Welling, Max. Herding dynamical weights to learn. In *ICML*, pp. 141, 2009.