Knowledge Extraction from Deep Belief Networks for Images

Son N. Tran
City University London
Northampton Square, EC1V 0HB, UK
Son.Tran.1@city.ac.uk

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

Abstract

In this paper, we introduce a rule extraction method for discovering knowledge in a set of images using Deep Belief Networks. The method is shown to be useful at pruning the networks trained on image datasets, explaining the relationship between a label and the data distribution, and training the networks on related image datasets. Deep architectures have been studied intensively recently thanks to their applicability on 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. To the best of our knowledge, this is the first knowledge extraction system for deep networks. Keywords: Deep Belief Networks, Neural-Symbolic Integration, Knowledge Extraction, Vision and Perception.

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 representing 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 problem domains and a wider applicability of such networks through a better understanding and explanation of the relations between network layers. Hence, in this paper, we focus attention on the problem of knowledge extraction [d'Avila Garcez *et al.*, 2001] from deep networks. We start by proposing a method for knowledge extraction from Restricted Boltzmann Machines (RBMs) [Smolensky, 1986], which can be seen as the basic building block of deep networks. We then unroll this method into an approach for extracting logical representations from Deep Belief Networks (DBN). We present a new extraction algorithm and experimental results on pruning the networks trained on image datasets, explaining the relationship between a label and the data distribution, and transferring the extracted knowledge to train the networks on related image datasets. 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 symmetrical networks, and applied in modified form recently to recurrent temporal restricted Boltzmann machines [de Penning *et al.*, 2011]. We expect knowledge extraction to serve as a tool to help analyse how representations are built in the different layers of the network, and in the study of the relationship between the depth of DBNs and performance. In a nutshell, this paper shows that, given a trained DBN, logical rules of the forms $h_j \leftrightarrow v_1 \wedge \neg v_2 \wedge v_3$ can be extracted from each pair of layers, each rule having a *confidence* value (defined in what follows). Here, the proposition h_j represents the activation state of a unit j in a layer, and the proposition $v_i (i = 1, 2, 3)$ represents the activation state of a unit i in a layer immediately below it (hence, v_i denotes that i is activated, and $\neg v_i$ denotes that i is deactivated). 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 shedding light into the structure of the network and the role of each layer/set of neurons. 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 values. 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 an extraction method for RBMs. In Section 4, we introduce the extraction approach and algorithm for DBNs. In Section 5, we present and discuss the experimental results, and in Section 6 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]. 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 an RBM with visible layer v and hidden layer h, when its global state reaches equilibrium, is exactly of the form above, more precisely:
$P(v,h) = \frac{1}{Z}\exp(-\sum_{i,j} v_i w_{ij} h_j - \sum_i a_i v_i - \sum_j b_j h_j)$.

The equilibrium state of an RBM can be achieved by Gibbs 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 corresponding 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 (an RBM) is trained in the normal way using Contrastive Divergence [Carreira-Perpinan and Hinton, 2005]. This greedy training has been shown capable of approximating well enough the idea of complementary priors; however, to improve performance of DBNs as a classifier an additional back-propagation 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. Higher layers in the hierarchy are expected to represent more concrete features such as edges, curves and shapes (in the case of image datasets as used in this paper). 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 distribution $P(\mathbf{v}|h^l_j = 1)$. 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], symmetric connectionist systems characterised 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. Here, extraction is explicit in that it produces symbolic knowledge rather than visualizations. In [Pinkas, 1995], a connectionist system and a rule set are said to be 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 $\mathbf{h}, \mathbf{v}$. Since RBMs are also symmetric connectionist systems, in what follows we present an extension of the penalty logic approach to rule extraction that is suitable for DBNs.

3 Knowledge Extraction from RBMs

In this section, we apply and modify the idea of penalty logic to extract knowledge from an RBM model. However, for reasons that will become clear in what follows, instead of rewriting the model as a set of formulas associated with a ranking function, a set of rules will be extracted and used for inference to evaluate how closely the rules represent the behaviour of the network model.

The propositional knowledge captured from an RBM $N = \{V, H, W\}$ can be represented as:

$$R = \{\langle w_{ij} : v_i \wedge h_j\rangle | w_{ij} > 0\} \cup \{\langle -w_{pq} : \neg(v_p \wedge h_q)\rangle | w_{pq} < 0\}$$

The extracted set of rules will capture the behaviour of an RBM, as has been discussed in the previous section. However, they are not as capable of expressing the relationships between the features represented by units in the RBM's hidden layer and raw data encoded in its visible layer (e.g. an image). Moreover, they are not capable of representing relationships among input variables. Finally, the penalty logic approach would extract $N_v \times N_j$ rules from an RBM, which is a concern in the case of a model with thousands of units such as the RBMs or DBNs used for image processing. We propose an alternative based on the concept of partial and complete model, as below.

Proposition 3.1. *For each unit h_j in the hidden layer of an RBM, it is possible to extract a single rule in the weighted-if-and-only-if (weighted-IFF) form:*

$$c_j : h_j \leftrightarrow \bigwedge_i v_i \wedge \bigwedge_t \neg v_t$$

where v_i and v_t correspond to the activated unit i and deactivated unit j in the visible layer. The rule's confidence-value is given by $c_j = \sum_{v_i} w_{ij} - \sum_{v_t} w_{tj}$

Definition 3.1. *(partial-model): A partial-model of a unit h_j in the hidden layer of an RBM is a weighted-IFF rule of the form:*

$$c_j : h_j \leftrightarrow \bigwedge_{i,w_{ij}>0} v_i \wedge \bigwedge_{t,w_{tj}<0} \neg v_t$$

Proposition 3.2. *The partial-model of a unit in the hidden layer of an RBM is the rule with the highest confidence-value possibly extracted from that unit.*

However, it is difficult to use the *partial-model* for inference because it does not capture information about the weight values between the set of input units and hidden unit h_j in the rule. This problem can be solved by having a complete model, as follows.

Definition 3.2. *(complete-model): A complete-model for a unit h_j in an RBM is a* partial-model *associated with a list of the absolute values of the weights between the units corresponding to conjuncts in the rule and h_j. It can be denoted as:*

$$c_j : h_j \overset{hv_j}{\leftrightarrow} \bigwedge_{i,w_{ij}>0} v_i \wedge \bigwedge_{t,w_{tj}<0} \neg v_t$$

with $hv_j = \{|w_{1j}|, |w_{2j}|, ..., |w_{N_vj}|\}$

In what follows, we investigate the pros and cons of partial-models and complete-models at representing RBMs. We start with an example.

Example (learning XOR function): We examine the theory by training an RBM to model a XOR function from its truth-table, where $true$ and $false$ are represented by 1 and 0, respectively. Our goal is to compare the rules extracted from trained RBMs having inputs x, y, z with different numbers of hidden units, against the rules of the XOR function as shown in Table 1.

X	Y	Z	Rules
0	0	0	$\top \leftrightarrow \neg x \wedge \neg y \wedge \neg z$
0	1	1	$\top \leftrightarrow \neg x \wedge y \wedge z$
1	0	1	$\top \leftrightarrow x \wedge \neg y \wedge z$
1	1	0	$\top \leftrightarrow x \wedge y \wedge \neg z$

Table 1: XOR problem: Truth-table and rules defining the conditions for activating hidden unit $\top$. All other truth-value assignments not specified by the rules should not activate any hidden unit.

Training z $\leftrightarrow$ x$\oplus$y in an RBM should also imply x $\leftrightarrow$ y$\oplus$z and y $\leftrightarrow$ x $\oplus$ z, given the symmetry of the network. Three RBMs having, respectively, 1, 3 and 10 hidden units were trained using Contrastive Divergence and the same learning parameters. After training, partial-model rule sets were extracted, as shown in Table 2. Notice how the rules in Table 2 that do not match a rule in Table 1 have a much smaller confidence value. In fact, if we were to remove the rules with confidence value smaller than the mean confidence for each rule set, only the rules that match the description of the XOR function from Table 1 would have been kept in Table 2. It is interesting to see that partial models have been able to capture the correct relationships between the input variables in this example, despite the inherent loss of information of partial models. Complete-models, with their confidence vectors, should be able to represent the behaviour of the network almost exactly, but partial models do not contain weight values, as discussed earlier. In what follows, we continue this analysis by applying the definitions of partial and complete models to DBNs.

#hiddens	Extracted rules
1	$17.9484 : h_1 \leftrightarrow x \wedge y \wedge \neg z$✓
	$1.499 : \top \leftrightarrow \neg x \wedge \neg y \wedge z$
3	$24.7653 : h_1 \leftrightarrow \neg x \wedge y \wedge z$✓
	$23.389 : h_2 \leftrightarrow x \wedge y \wedge \neg z$✓
	$27.1937 : h_3 \leftrightarrow \neg x \wedge \neg y \wedge \neg z$✓
	$13.4209 : \top \leftrightarrow x \wedge \neg y \wedge z$✓
10	$4.0184 : h_1 \leftrightarrow x \wedge \neg y \wedge z$✓
	$8.9092 : h_2 \leftrightarrow x \wedge \neg y \wedge z$✓
	$18.4939 : h_3 \leftrightarrow \neg x \wedge y \wedge z$✓
	$0.5027 : h_4 \leftrightarrow \neg x \wedge y \wedge \neg z$
	$7.4417 : h_5 \leftrightarrow x \wedge \neg y \wedge z$ ✓
	$5.0297 : h_6 \leftrightarrow \neg x \wedge y \wedge z$ ✓
	$7.6313 : h_7 \leftrightarrow x \wedge \neg y \wedge z$ ✓
	$22.0653 : h_8 \leftrightarrow x \wedge y \wedge \neg z$✓
	$19.6188 : h_9 \leftrightarrow \neg x \wedge \neg y \wedge \neg z$✓
	$14.6054 : h_{10} \leftrightarrow \neg x \wedge \neg y \wedge \neg z$✓
	$3.4366 : \top \leftrightarrow x \wedge y \wedge z$

Table 2: Rules extracted from RBMs trained on the XOR function (✓ means the rule matches a rule in Table 1).

4 Knowledge Extraction from DBNs

Since the structure of a DBN can be seen as a stack of RBMs, the simplest method for extracting knowledge from a DBN is to combine the rules extracted from each RBM. Hence, the rules extracted from a DBN can be represented (using *partial-models*) as:

$$R = \{c_j^{(1)} : h_j^{(1)} \leftrightarrow \bigwedge_{i,w_{ij}^{(0)}>0} v_i \wedge \bigwedge_{t,w_{tj<0}^{(0)}} \neg v_t\}$$
$$\bigcup_{l=1}^{L-1} \{c_j^{(l+1)} : h_j^{(l+1)} \leftrightarrow \bigwedge_{i,w_{ij}^{(l)}>0} h_i^{(l)} \wedge \bigwedge_{t,w_{tj<0}^{(l)}} \neg h_t^{(l)}\}$$

In the case of *complete-models* each rule is also associated with a confidence-vector, as discussed in the previous section.

Even though DBNs are stochastic systems, our experiments have shown that deterministic inference on the extracted rules can capture the behaviour of the networks. 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:

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

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 unit connected to it is already activated. However, it is not easy to determine the activation state of a unit when it appears in different rules with positive and negative weights, respectively. 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 $Sum_j = \sum_{i,v_i=1} w_{ij}$ be the sum of the weights of the connections between h_j and all activated units v_i. If $Sum_j > 0$, we say that the *positive* rules 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 larger Sum_j is, the higher the probability is of h_j being activated. From this, we can also see that if $Sum_j < 0$ then one should have less confidence in the activation of h_j, that is, the probability of h_j being activated should be low.

In general, the inference rule for our extracted rules, where $c, \alpha_1, \alpha_2, \ldots, \alpha_n, \alpha_h$ are confidence-values, should be:

$$
\begin{array}{l}
c : h \overset{w_1,w_2,\ldots,w_n}{\leftrightarrow} belief_1 \wedge \neg belief_2 \wedge \ldots \wedge belief_n \\
\alpha_1 : belief_1 \\
\alpha_2 : belief_2 \\
\vdots \\
\alpha_n : belief_n \\
\hline
\alpha_h : h \text{ with } \alpha_h = w_1\alpha_1 - w_2\alpha_2 + \ldots + w_n\alpha_n
\end{array}
$$

In the case of partial-models, $\alpha_h = c/n$.

When rules are extracted from a deep architecture, first one should apply the above inference rule to derive hypotheses for the first RBM in the stack, and then repeat the process, assuming that those derived hypothesis are beliefs, in order to infer new hypotheses for the second RBM in the stack, and so on. For this, the confidence value of a belief should be normalized to $[0, 1]$ using, e.g., a sigmoid function. Alternatively, one might set the confidence value of a belief to 1, if the confidence value of its associated hypothesis is positive, or 0 otherwise.

5 Experiments and Results

In this section, we validate empirically the connection between logical rules and stochastic sampling established above by extracting rules and visualizing features of Deep Belief Networks. We also present new results on transfer learning using rule extraction. We have trained a system with 784, 500, 500 and 2000 units over four layers on 5000 examples from the MNIST handwritten digit dataset [Hinton *et al.*, 2006]. Some of the rules extracted from the deep network are shown below [Son Tran and Garcez, 2012].

$$0.99250 : h^2_0 \leftrightarrow \neg h^1_0 \wedge h^1_1 \wedge \neg h^1_2 \wedge ...\neg h^1_{783}$$
$$0.99250 : h^1_0 \leftrightarrow \neg v^0_0 \wedge v^0_1 \wedge \neg v^0_2 \wedge ...\neg v^0_{783}$$
$$1.00000 : h^1_1 \leftrightarrow \neg v^0_0 \wedge \neg v^0_1 \wedge \neg v^0_2 \wedge ...\neg v^0_{783}$$

where the confidence values have been normalized by a sigmoid function, and h^l_j represents a unit j of hidden layer l.

Let us consider the use of logical inference, as discussed above, to try and visualize the features captured by the units in the second hidden layer. In this experiment, the confidence value of a belief b associated with a hypothesis $c : h$ is set to 1 if $c > 0$, and 0 otherwise. Figure 1 shows the progression of 10 images (from left to right) generated using network sampling (shown in the top row) and the above, sign-based, logical inference (shown in the bottom-row). The results indicate information loss, as expected, but also suggest that inference with sign-based activation should be faster than network sampling; the network might have several local minima for each concept, so that each image was generated after intervals of 10 iterations, totalling 100 iterations in the network). We then

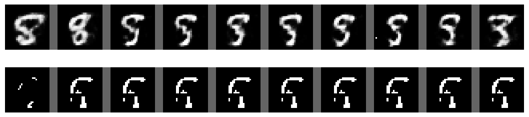

Figure 1: Images generated from network samplings (top row) and logical inference using sign-based activation/confidence value calculations (bottom row).

added labels to the network for classification. The labels are added to the visible layer of the top RBM in the hierarchy as soft-max units [Hinton *et al.*, 2006]. After training on $60,000$ samples of the MNIST dataset, the network achieved 97.63% accuracy on a test set consisting of $10,000$ samples. If we apply logical inference using a sigmoid function to normalize the confidence values, the rules achieve 93.97% accuracy on the same test set. If we then clamp the labels and reconstruct the input images, it can be verified that logical inference can generate fairly similar images, as depicted in Figure 2, where the top row shows the images generated by the network using Gibbs sampling, and the bottom row shows the images produced by inference.

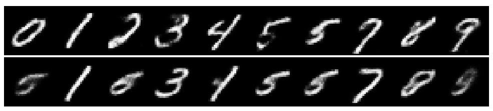

Figure 2: Images reconstructed by clamping labels of DBN and using Gibbs sampling (top row), and by setting hypotheses to *true* in the set of rules and using logical inference (bottom row).

5.1 System Pruning

The goal here is to use the confidence value of each rule to evaluate the contribution of its corresponding hidden unit in an RBM (or in the hidden layers of a DBN), removing from the system the "weak" hidden units, i.e. those with a low confidence value. We shall test the drop in network performance as a measure of usefulness of a rule's confidence value. Rules were extracted from an RBM trained on the MNIST dataset. Rules with confidence value smaller than a specified threshold were removed, and the remaining rules were re-encoded into a smaller RBM. The performance of the new RBM can be evaluated by comparing the reconstruction of images with that done by the original RBM, as illustrated in Figure 3.

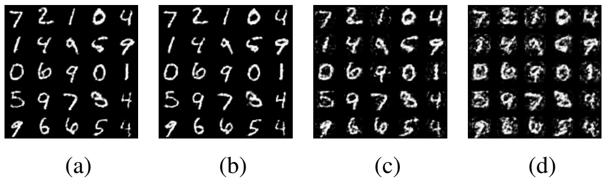

Figure 3: The reconstructed images from original RBM trained on 60,000 samples (3a), RBM encoding 382 rules with highest confidence values (3b), RBM encoding 212 rules with highest confidence values (3c), and RBM encoding 145 rules with highest confidence values (3d).

Figure 3 indicates that it is possible to remove rules/hidden units and maintain performance, with the behaviour of the system remaining pretty much unchanged until some "important" rules start to be removed. In order to measure this more precisely, we have provided the features obtained from the RBMs as an input to an SVM classifier. Figure 4 shows the relationship between accuracy and the sizes of the RBMs. Interestingly, at first, accuracy has increased slightly, and has remained stable until some 50% of the hidden units had been removed. It then started to deteriorate rapidly, but was still at around 95% when 400 of the 500 hidden nodes were removed.

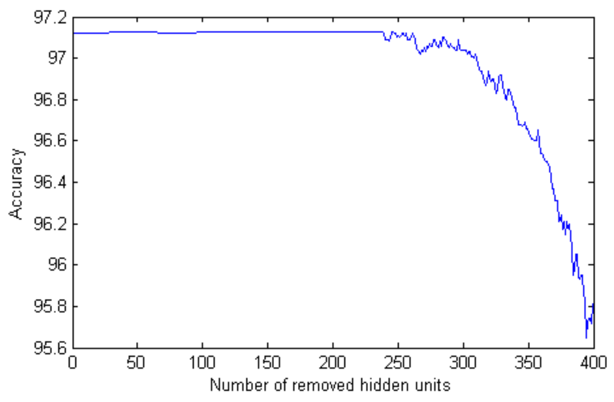

Figure 4: Classification performance of SVM on MNIST dataset with input features from RBM whose hidden layer is reduced by removing hidden units with lower confidence values.

5.2 Transferring Knowledge

Rule extraction in the context of images may find application in transfer learning [Torrey *et al.*, 2010]. To evaluate this, we have extracted rules from a network trained on an image dataset and encoded a subset of these rules on another network to be trained on a related image dataset. In particular, we have measured classification performance when transferring from the handwritten digit dataset to natural images of digits, and to writing styles, as shown in Figure 5. We have trained an RBM on 5,000 images from the MNIST dataset, extracted and pruned the rules before transferring them to RBMs that were trained on natural images of digits (ICDAR dataset[1]) and on character writing styles (TiCC dataset[Maaten, 2009]). Figure 5 shows some examples from the source and target domains.

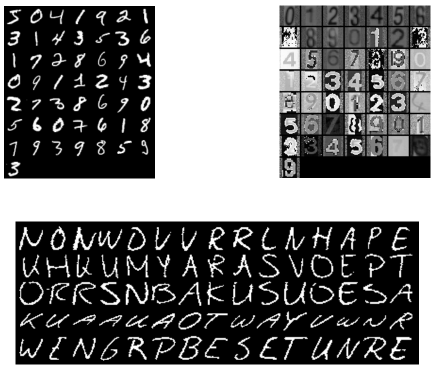

Figure 5: Visualization of MNIST images (top left), ICDAR images (top right), and TiCC character images (below) for five writers (one per line).

We started by extracting and transferring all complete-models. This is basically the same as using the RBM trained on MNIST as the starting point for training in the new domains (ICDAR and TiCC). The RBM was then augmented with newly-added hidden neurons for training in the new domains. The transferred knowledge is expected to help learning in the new domains; this knowledge was kept unchanged, with only the weights of the newly-added neurons being allowed to change. The results were put through an SVM to provide a classification measure, for the sake of comparison. As expected, in the case of transferring complete-models from MNIST to ICDAR, we have observed a small improvement in classification performance: the network trained with transfer achieved 51.55% accuracy on the ICDAR dataset, while a knowledge-free RBM achieved 50.00%. We also ran just an SVM on the raw data, which achieved 38.14% accuracy. A slightly better improvement was observed when transferring from MNIST (handwritten digits) to TiCC (handwritten characters) in order to recognize writing styles. Table 3 shows all the results.

Target	SVM	RBM	RBM with Transfer
ICDAR	38.14%	50.00%	**51.55%**
TiCC	72.94%	78.82%	**81.18%**

Table 3: Comparison with RBM with transfer learning

It is generally accepted that the performance of a model in a target domain will depend on the quality of the knowledge it

[1]http://algoval.essex.ac.uk:8080/icdar2005/index.jsp?page=ocr.html

receives and the structure of the model. We then evaluate performance also using different sizes of transferred knowledge. Figure 6 shows that if this size is too small, the model will be dominated by the data from the target domain. If, on the other hand, this size is too large, the model might not learn from the target domain, with a consequent drop in performance as the model responds mainly to the knowledge transferred. Further analysis of our rule extraction method might help guide this transfer process, which nevertheless could turn out to be a domain-dependent task.

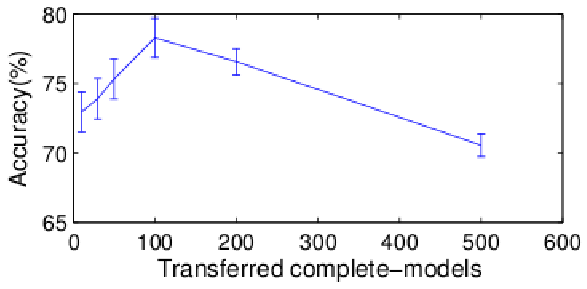

Figure 6: Performance with different sizes of transferred knowledge on the TiCC dataset: for each number of rules transferred, we have added zero, 30, 50, 100, 200, 500 and 1000 nodes to the network. Each network was tested 50 times with the graph showing the mean accuracies and standard deviations. Our results also indicate that adding 1000 nodes produce worse accuracy than adding no nodes at all, with the best performance at 200 extra nodes.

6 Conclusions And Future Work

We have proposed a knowledge extraction method and have applied it to deep belief networks trained on images. The results indicate that a deep belief network can be represented by a set of rules with logical inference serving as an alternative to stochastic sampling. In addition, by transferring the extracted rules onto a target domain, classification performance can be improved on that domain. As future work, we are interested in studying the relationships between rules' confidence values and networks in a more systematic way, and in using rule extraction based on partial models to guide transfer learning.

References

[Aha *et al.*, 2010] David W. Aha, M. Boddy, V. Bulitko, and A. S. d'Avila Garcez et al. Reports of the AAAI 2010 conference workshops. *AI Magazine*, 31(4):95–108, 2010.

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

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

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

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

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

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

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

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

[Lee *et al.*, 2009a] 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*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

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

[Maaten, 2009] Laurens Maaten. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009.

[Pinkas, 1995] Gadi Pinkas. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, 77(2):203 – 247, 1995.

[Smolensky, 1986] Paul Smolensky. 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*, pages 194–281. MIT Press, Cambridge, 1986.

[Son Tran and Garcez, 2012] Son Tran and Artur Garcez. Logic extraction from deep belief networks. In *ICML 2012 Representation Learning Workshop*, Edinburgh, July 2012.

[Torrey *et al.*, 2010] Lisa Torrey, Jude W. Shavlik, Trevor Walker, and Richard Maclin. Transfer learning via advice taking. In *Advances in Machine Learning I*, pages 147–170. 2010.