An Abstract Deep Network for Image Classification

Anthony Knittel and Alan D. Blair

University of New South Wales
{aek,blair}@cse.unsw.edu.au

Abstract. In order to allow more flexible and general learning, it is an advantage for artificial systems to be able to discover re-usable features that capture structure in the environment, known as Deep Learning. Techniques have been shown based on convolutional neural networks and stacked Restricted Boltzmann Machines, which are related to some degree with neural processes. An alternative approach using abstract representations, the ARCS Learning Classifier System, has been shown to build feature hierarchies based on reinforcement, providing a different perspective, however with limited classification performance compared to Artificial Neural Network systems. An Abstract Deep Network is presented that is based on ARCS for building the feature network, and introduces gradient descent to allow improved results on an image classification task. A number of implementations are examined, comparing the use of back-propagation at various depths of the system. The ADN system is able to produce classification error of 1.18% on the MNIST dataset, comparable with the most established general learning systems on this task. The system shows strong reliability in constructing features, and the abstract representation provides a good platform for studying further effects such as as top-down influences.

1 Introduction

Deep Learning has recently become a significant area of study in machine learning, particularly related to computer vision [1]. The main object of this approach is the discovery of intermediate features that capture structure in the environment being observed. These features can be re-used and incorporated into other features, and allow learning based on a deeper network structure than was possible with previous neural network approaches. Deep networks do not always provide better performance than shallow classification techniques, but their ability to combine and re-use elements in a compositional hierarchy makes them well suited to certain kinds of tasks, such as object and digit recognition, and gives them a lot in common with various models of cognitive processing [2–4]. They may also offer new insights into the functioning of certain learning mechanisms within the cerebral cortex (although the relationship with cortical structures has been called into question, and alternative models have been suggested [5]).

One of the established approaches is the use of stacked Restricted Boltzmann Machines (RBM) [6], which are trained in an unsupervised manner before the

M. Thielscher and D. Zhang (Eds.): AI 2012, LNCS 7691, pp. 156–169, 2012.

task-specific training is applied. This allows the system to capture features of the observed environment, and is an important design philosophy as it forces the system to capture structure, rather than just finding the minimal set of features relevant for classification.

One limitation of traditional Artificial Neural Networks (ANN), particularly those with increasing depth, is they can become stuck in local minima, where training ceases to improve performance even though better solutions are available [7]. Neural networks can give very precise solutions, however this depends on the network being initialised in a suitable area of the search space. RBM techniques address this problem, using unsupervised learning to initialise the network according to significant features of the environment, followed by a method to fine-tune performance according to the task, which can be done using back-propagation [6]. This allows initialisation of a deep network that is not likely to give adequate behaviour from random initialisation of weights. However, RBM networks have shown limitations in terms of reliability [8, 9], and a number of attempts have been made to improve discovery of features.

Evolutionary Computation provides an alternative approach to machine learning, usually based on genetic algorithms and reinforcement techniques. Evolutionary systems tend to be very reliable at finding a good solution, however the use of random variation, rather than gradient descent used in ANNs, often does not provide the same precision found with neural networks or kernel methods.

Learning Classifier Systems (LCS) are an evolutionary technique that combine evolutionary processes with reinforcement learning, to maintain a population of classifiers that collectively model the observed system [10]. The Genetic Algorithm used by many LCS approaches follows an evolutionary analogy, however the process of capturing a population of rules based on reinforcement can be viewed as an analogy of cognitive learning processes, with a greater degree of generalisation than Reinforcement Learning. The Activation-Reinforcement Classifier System (ARCS) [11] is a recent LCS approach that bases the design on abstract cognitive features, such as reinforcement of memory traces through use, as seen in cognitive models such as ACT-R [12]. This process is used as a basis for maintaining the rule and feature population. An implementation of this system [13] provides a method for building a feature hierarchy of re-used elements, rather than using a population of discrete rules with redundant building blocks typical of Genetic Algorithm systems. In this feature hierarchy elements are constructed from combinations of other features, producing a deep network related to that found in Deep Learning neural networks.

ARCS has shown reliability in constructing a feature hierarchy on the MNIST visual classification task [13], however the performance level reached is far outside that found by neural networks (10% vs 1% error). Another LCS technique based on Haar-like features has shown better performance, reaching 4% error with the aid of confusion matrices [14], however this is based on pre-defined features and does not build a deep feature hierarchy, and again is well outside the performance level seen by the best neural network and kernel systems.

The principles used for building a feature hierarchy in ARCS are very different to that used in RBM and convolutional neural network systems. ARCS uses an abstract representation, borrowing principles from behavioural psychology which deals with abstract cognitive phenomena. In contrast neural networks relate to localized phenomena such as the interconnection and behaviour of neurons, identified from studies in neuroscience [15]. Providing a different angle on a common problem can be useful for giving a broader perspective. The broader aim of studying the development of a feature hierarchy based on abstract representations, is to use the model to incorporate other important effects in visual perception, such as the role of context in the activation process and related top-down influences. As an example the model by Bar [16, 17] describes a process where the general context of the scene is interpreted first, largely from low spatial frequency information, and this provides a top-down influence on the activation of lower level features that capture details. The use of a more abstract model, which employs a hierarchical part-based representation built in a self-organising manner, gives a good platform for introducing and studying these kinds of effects.

Hinton's model [18] uses unsupervised learning to build the feature network, followed by a fine tuning process to improve classification. ARCS uses a different approach and builds a feature network based on random creation from observations, modified according to reinforcement and selection. This may also benefit from a related fine tuning approach to improve performance. With certain modifications it is possible to introduce back-propagation into the feature network constructed by ARCS, allowing an alternative and reliable manner for constructing features, along with a gradient descent technique to improve performance.

2 Connecting the Reinforcement-Based Feature Network with Gradient Descent

The features used in ARCS are abstract and do not have a direct relationship with individual neurons, but rather represent features that may be captured by a group of neurons or connection weights. The feature network used in ARCS consists of a population of low level atomic features, which can be directly compared with observations, and a network of composite features, that represent common combinations of smaller features. This produces a network of features of increasing field size, each constructed from lower level elements, producing a network related to that in models of the human visual system [3].

Atomic features are represented using a sum of weighted values, created to respond to a section of an observation. Composite features are constructed from approximately 2-8 other features, which may include other composites, and vectors representing the relative positions of the features. When an observation is made, each atomic fragment is tested at each position, producing a map of match values, and each composite is tested according to the match values of each child at their respective positions, producing a match map. Activation values are simplified as binary values according to a threshold, and a composite is activated at a given position if each child is also active.

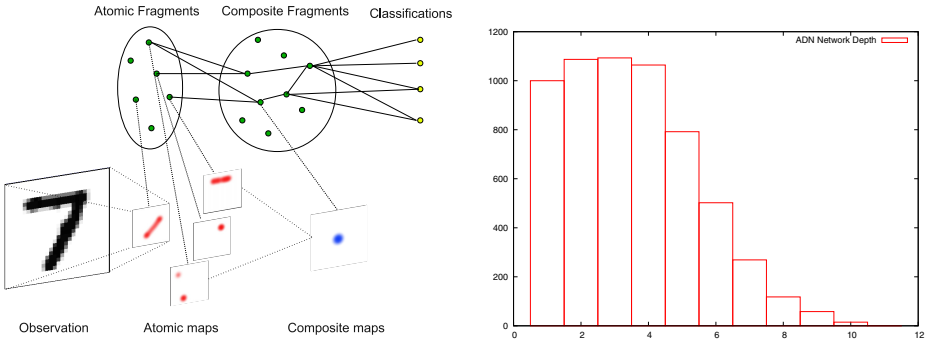

Fig. 1. (L) Connections between atomic and composite fragments, and maps representing match positions of each relative to the observation. (R) Histogram of depth of network produced with the ADN system (Top-BP).

A representation of the connections between fragments and respective match maps used by ARCS is given in Fig. 1(L).

Learning Classifier Systems act by identifying the set of classifiers, or rules, that match the current observation, and each classifier defines an action or classification, along with a value representing the accuracy or expected reward when the classifier matches. In the ARCS system composite features are connected to classifications, using a weight reflecting the probability $P(class|feature)$. Note that this description is a simplification of that used in [13][1]. For each observation a set of active composite fragments is determined, which acts in a similar manner to the set of active rules used in standard LCS systems. From this set the classification is chosen according to the set of Q values captured in the $P(class|feature)$ weights. In exploration/exploitation paradigms this is chosen probabilistically using the Boltzmann distribution $\frac{e^{q_i}}{\sum_j e^{q_j}}$, however in a classification task the maximum value can be used. According to the result the Q value is updated for each class for each active composite feature.

Each fragment also maintains an *accessibility* value, reflecting scalar reinforcement through use, which is used to maintain the feature population. At each time step a value of 1 is distributed amongst the fragments connected to the composite with the highest association with the correct class. This value is derived from models of reinforcement of memory traces such as ACT-R, using a decay function $f_t = \alpha(f_{t-1} + r)$, and provides a ranking amongst features in the population according to the frequency that the feature is significant for classification. This has been modified to use an average function so there is less variation over time for each, while providing ranking amongst the population: $f_t = f_{t-1} + \alpha(r - f_{t-1})$.

[1] The previous implementation uses a separate population of rule features to connect composites to classifications, which has the advantage of allowing further sparseness as not all connections between composites and classes are represented, but requires maintenance of another population. This has been removed to reduce complexity, and shows the same performance.

Performance of the feature-based ARCS system reaches approximately 10% error on MNIST [13]. This shows an ability to construct a feature hierarchy using reinforcement methods, and runs with stability and reliability, however classification performance is limited compared to neural network or kernel methods.

2.1 Combining the Feature Network with Neural Network Techniques

While ARCS is able to find a rough solution reliably, back-propagation has many advantages in finding a more precise solution, and can be introduced with appropriate modifications.

ARCS chooses a classification according to the feature with the highest association probability with a class. The classification decision is based on individual features rather than combinations, although combinations can be constructed as new features. One limitation of this approach is that features represent combined positive activation of child elements, and negative weights are not used. Introducing negative weights does not fit the design of the creation process well, as currently new features are constructed from the set of features currently active, emphasising relevance between the features, whereas negative weights would be randomly sampled from the set of all features currently inactive. Introducing this did not improve performance.

The classification step can be modified such that the relationship between composite features and classes acts as a Multi-layered Perceptron (MLP) [15], based on a weighted sum and activation function, modified through back-propagation, along with bias values. These weights replace the $P(class|feature)$ values in the previous design. Given a set of activation values of composite fragments, activation of each class can be determined, and modified according to back-propagation. This only acts on the connections between composites and classes at this stage, not on connections between composites. As the composite network is dynamic, and connections exist between many composites and classification nodes, the network does not have a clear layered structure, but rather contains random or fully connected links between composite fragments at different depths and the classification nodes. This is shown in Fig. 1. The back-propagation method is given in the following equations [15], using a learning rate of 0.01 and no momentum:

$$\delta_k = O_k(1 - O_k)(O_k - t_k) \tag{1}$$

$$\Delta w_{(j,k)} = -\eta \delta_k O_j \tag{2}$$

$$\Delta \theta_k = -\eta \delta_k \tag{3}$$

$$\delta_j = O_j(1 - O_j) \sum_{k \in K} \delta_k w_{(j,k)} \tag{4}$$

where O_x is the activation value, t_x the classification target, $w_{(x,y)}$ the weight, θ_x the bias and $k \in K$ the set of parents of j.

It would be more consistent with the Deep Belief Network model [18] to perform the feature discovery process first and run fine tuning such as back-propagation afterwards, however in ARCS the links between features and classifications use a method based on argmax, rather than summation and sigmoid activation function common in back-propagation systems. As such it is more suitable to change to a system based on summed activation, during the feature discovery process as well as during fine tuning. This requires a modification to the reinforcement method used to maintain the feature population. The important principle followed is allocation of a fixed resource at each time step, which is allocated to features significant to the decision made. This process has shown to be effective in balancing general and specialised rules [11], and ensuring coverage of observations. Instead of reinforcing the feature with highest Q value, the reinforcement is distributed amongst those (top-level) features that are significant in activation of the correct class, as given in Equation 6. As such the reinforcement of the feature population is integrated with the back-propagation process and both are active at the same time, rather than as two separate phases.

$$r_j = e^{\sum_{k \in K} w_{(j,k)} x_k} \tag{5}$$

$$f_j = f_j + \alpha(R \frac{r_j}{\sum_{i \in I} r_i} - f_j) \tag{6}$$

where x_k is 1 for the correct class and -1 for incorrect, R is the reinforcement to be distributed (value 1), and f_j is the accessibility reinforcement for the feature.

To distinguish this design from the Learning Classifier System approach, this system is referred to as an Abstract Deep Network. In summary the first implementation of this system, referred to as Top-BP, uses random creation of atomic and composite fragments, and atom fragments use a weighted product-sum method with sigmoid activation, and a threshold to give a binary activation value, for each position. Composites are active at each position if all child elements are active at their respective positions. The classification layer is a fully interlayer connected network between each composite and each classification node, and back-propagation acts only on the classification layer connections.

2.2 Training and Evaluation on MNIST

The MNIST dataset of handwritten digits [19] is a standard test used by many image classification techniques. The best performing systems are kernel methods and convolutional neural networks using a range of pre-processing and specific transformation techniques [19, 20], and the RBM based Deep Belief Network [18] is one of the best generalised learning systems. A convolutional RBM method [21] provides another approach that captures some of the advantages of both, using unsupervised learning in a convolutional max-pooling based architecture.

Selective training regimes are sometimes used on MNIST, such as training individual classes before introducing wider selections of the training set [18].

Table 1. Summary of results for ARCS and ADN systems, and comparison with existing systems

System	Features	Fine tuning	Error
ARCS	Convolved		10.0%
ADN	Convolved	Top BP	1.57%
ARCS	Full size		12.5%
ADN	Full size	Network BP	2.72%
ADN	Convolved	Network BP	1.39%
ADN	Convolved	Atom BP	1.47%
ADN	Convolved	Top BP, Freeze	1.23%
ADN	Convolved	Atom BP, Freeze	1.18% *
ADN	Convolved, 100	Atom BP, Freeze	1.78%
Ebadi 12	Haar features XCS		4.0%
Hinton 06	Full size RBM		1.25%
Ranzato 07	Convolved NN		0.64%
Lee 09	Convolved RBM		0.82%

To train ARCS and the ADN system a simpler process is used, for each training step a random image is chosen from the MNIST training set (60,000 images). After every 100,000 training steps an evaluation is performed using each of the 10,000 test images, with no adjustments to weights or the population.

Performance of ARCS and the Top-BP implementation of the Abstract Deep Network are shown in Table 1. The use of back-propagation gives greatly improved performance, reaching a level of 1.57% (vs 10%). Analysis of the network topology shows the number of nodes at each depth, approximately half of the 5000 composite nodes are at depth 4 or below, while the network has a maximum depth of 10. The higher depth does not seem necessary for this problem, however analysing the connectivity of the network shows most nodes (4571) have only 2 child elements, and as such the network is very sparse, in a sense representing a clustering representation. The distribution of nodes with 2 children is higher than the creation distribution, indicating a self-organising preference for nodes with limited connectivity.

3 Gradient Descent of the Feature Network

A further advantage may be found by allowing fine tuning to influence the weights of the composite feature network, as well as the classification stage. This however requires a continuous activation function for composite features rather than the existing binary approach. A softer activation function may also handle partial activations in noisy or occluded images in a more reliable way.

A continuous approximation of the AND function used by composites can be found by using a weighted summation method with a sigmoid activation function, and setting the bias such that the feature is 'active' only when each child is also activated. Composite features are created from observations by selecting a number of currently active features. New composites can be created

using the continuous activation function by setting the bias of the new composite as $\theta = -\sum_i a_i + \epsilon$, where a_i is the current activation of each feature at the chosen position, ϵ is a margin of tolerance, and it is assumed weights $w_{i,j}$ are initially set to 1.

Using this method the network produced is similar to that of MLPs [15], and the delta values from the classification layer may be passed down through the network. There are however a number of considerations, a) the network is not arranged in fixed layers, which complicates the process of passing delta values through the network, b) the external (classification) layer is connected to nodes at different depths, and c) each feature does not have a single activation value, but rather maintains a map of activation values at various positions.

3.1 Full-Size Features

The evolutionary ARCS system and the Abstract Deep Network described earlier use atomic features with a small receptive field, that are convolved on the observation. In contrast Hinton's RBM method [18] uses features that match the full-size of the image, removing translation invariance but allowing position specific information in the features. Full-size features, which respond to a single position, require less complexity to implement as each feature has a single activation value rather than a map, simplifying the activation process as well as the back-propagation procedure.

An implementation is described that is based on the ARCS system, but uses full-size features, to allow the use of gradient descent with minimum complexity. This is done by generating atomic features that have the same dimensions as the input image, however are defined only in a small region, using the same method used to define the smaller size features. Potential advantages of full-size features are position specificity and reduced processing, however disadvantages are lack of translational invariance, such that a given feature must be identified in each position it is to be used.

The gradient descent algorithm is slightly different to the standard approach used in MLPs, as the network is not defined in fixed layers, and connections between active fragments and the classification layer occur at multiple depths.

The error signal produced from the classification nodes provides a delta value for each top-level feature, as described in Equation 1, and using these values gradient descent can be applied to lower features with a variation of the standard approach. Firstly, from the set of active top-level fragments and all associated child elements, an ordering is constructed such that each child fragment occurs after its parent. Examining each fragment (k) in turn , for each child (j) the respective delta value is adjusted according to Equation 7 below, and the weight of the connection with the parent and bias are modified according to the previous Equations 2 and 3, updating the weights and biases in the feature network.

$$\Delta\delta_j = O_j(1 - O_j)\delta_k w_{(j,k)} \tag{7}$$

3.2 Behaviour with Continuous Feature Activation and Gradient Descent

Behaviour was tested for a number of methods, using full-size features with ARCS, using the continuous-valued feature activation method with full-size features, and using back-propagation, referred to as Network-BP (full-size). Results are given in Table 1. Full-size features show a slightly worse result than convolved features with ARCS of 12.5%, while full-size features using the continuous-valued activation method alone is significantly poorer at 17% error. When back-propagation is used along with continuous-value activation, passing gradient descent adjustments through the weights of the feature network, performance is improved to 2.72%, significantly better than the full-size feature approaches without gradient descent, however with higher error than the convolved system with back-propagation of the top layer.

The continuous activation function operates slightly differently to the AND function previously used by ARCS, and on its own does not support the activation process as effectively, however allows effective learning with gradient descent. Full-size features are not shown to perform as well as convolved features in this domain, however are useful for comparison as they have reduced design complexity and require less processing time.

3.3 Gradient Descent with Convolved Features

Full-size features reduce complexity and processing, however there are conceptual and practical advantages with the convolved method, as individual features can be matched in multiple positions, providing translational invariance. Convolutional methods are also able to scale to larger and more realistic image sizes, and have been shown to be a successful approach [21].

In the convolved system, each feature has a range of activation values according to match positions, however classification nodes are activated from a single value for each connected feature. As such the classification step acts as a bag-of-words model according to the top-most features, while the composition network acts as a part-based model [22]. Back-propagation from classification provides a well defined delta value for each top-level feature, however handling back-propagation through the network must handle a distribution of values for each feature for various match positions.

The activation value of each top-level composite is given by the max value over its match positions, and refers to a single location. Back-propagation can be performed according to the value at this position, and applied to the match position of each child that contributes to this value.

Passing delta values through the network without clearly defined layers, when updates are applied sparsely, would require a complex procedure to combine delta updates according to different match positions of each feature, while at the same time identifying dependencies between nodes to ensure all parents of a given node are updated before its children. For simplification update values can be applied in a distributed manner, at the cost of multiple passes through

the network. A distributed approach allows delta updates to refer to different positions of a given node in different passes, as required when receiving back-propagation signals from a number of parent features, which may each relate to different match positions.

The distributed update procedure acts using the delta value of each active top-level fragment, given in Equation 4. The activation value for the fragment is given by its match position with the highest value. For each fragment a distributed top-down pass is run on each child, setting the delta to be used for each child, and subsequently updating the weight to the child, according to Equation 8. The operation recurses to each child, using the activation value at the appropriate position for the child relative to the chosen position in the top level fragment. The weights between composite fragments and between atomic fragments and composites are updated in this manner using a depth-first process.

$$\delta_j = O_j(1 - O_j)w_{(j,k)}\delta_k \tag{8}$$

note that the δ_j value update is not added, but is set once and proceeds in a depth-first pass, with the weight and bias values updated in each pass. In contrast the full-size feature method is breadth-first.

Implementing convolved features with back-propagation (Network-BP) gives improvement over the system based on full-size features, and is comparable to the Top-BP system based on gradient descent of only the top layer, reaching an error rate of 1.39%. Advantages over the Top-BP method are minor in this domain, however it also allows greater flexibility of learning the structure of the feature network.

3.4 Adjustment of Atomic Fragments

Atomic features are created from an observation and remain fixed. Passing the BP updates to the atomic fragments allows them to be modified to better suit the observations matched by the feature, and greater sensitivity. The BP process described previously passes a delta value to each atomic fragment, which can be used to alter the feature. The top-down operation also passes position information about where the feature matches the observation. Adjustment to the feature is performed according to: $\Delta F_{(x,y)} = -\eta\delta_j I_{(x',y')}$, where $F_{(x,y)}$ is the weight value of the feature at a given position, and $I_{(x',y')}$ is the respective observed value. Behaviour (Atom-BP shown in Table 1) shows similar performance on MNIST as with the Network-BP and Top-BP methods of 1.47%, and offers further flexibility with adjusting the feature network to fit observations.

3.5 Fine Tuning

Each of the described approaches act in an online manner, continuously updating the feature population and the weights between features with each new training example. More detailed fine tuning can be performed by freezing the feature population, and allowing the back-propagation to continue independently. This

was carried out by running the online system for 10^6 examples, before halting population updates and continuing training of the network weights.

Operation using this approach shows less variation in the result, and gives an improved classification level of 2.26% (vs 2.72%) for full-size features, of 1.23% for convolved features (Top-BP), and 1.18% (vs 1.49%) for convolved features (Atom-BP), taken as the average over four runs. This result shows classification performance equivalent to that shown by the Deep Belief Network used in [18].

3.6 Reduced Feature Set

In the previously described runs, 1000 atomic fragments and 5000 composites have been used. In order to test the influence of a reduced feature set, a population of 100 atoms is used, as shown in Table 1. The full-size system showed a significantly increased error level of 13.53% (from 2.26%), while with convolved features it is able to maintain a level of 1.78% (compared to 1.18% with 1000 atoms). For the convolved system, using a reduced population gives a good compromise, maintaining good performance while reducing processing time.

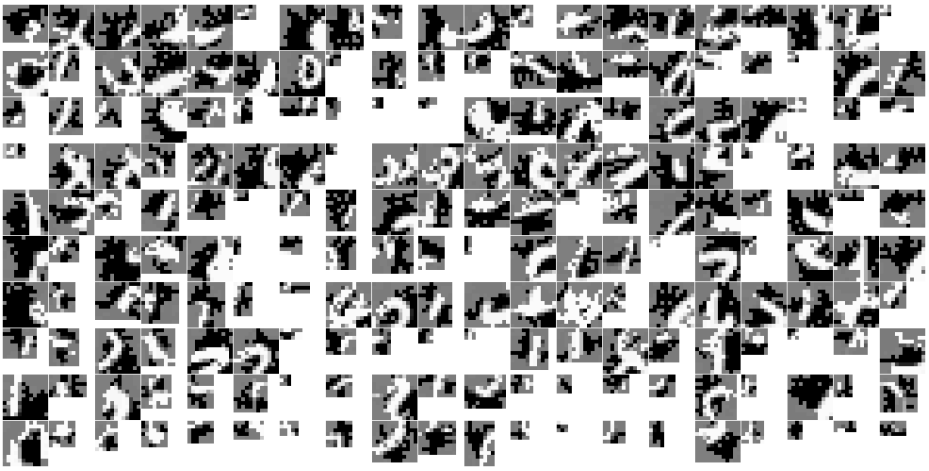

Fig. 2. Example of atomic features produced in the Atom-BP Abstract Deep Network

4 Discussion and Conclusions

We have described the use of an Abstract Deep Network, that builds a feature network using reinforcement related to Evolutionary Computing, along with a fine tuning step similar to that used by MLPs. This has been applied to the MNIST dataset without pre-processing, and only using spatial assumptions, using vectors to describe relationships between features. In general the system has shown strong reliability, producing similar behaviour with each run and robustness to parameter value changes.

The feature network is self-organising rather than pre-defined. A hierarchical feature network with a depth of 11 was able to be maintained and used for classification, the number of nodes at each depth is shown in Figure 1(R). The large depth is due to the sparseness of the network, with approximately 2-6 child elements for each feature, dominated by 2-child connections, far fewer weights than an equivalent fully connected network. Gradient descent updates were passed through this network to perform fine tuning.

Classification performance is comparable with existing deep learning techniques [18], giving an equivalent classification error rate of 1.18% compared to 1.25%. A careful training routine is not needed, training is performed simply by selecting random examples from the training set, and is able to act in a continuous online manner, without separate feature training and fine-tuning regimes, although a slight improvement to results was produced after freezing the feature population. Specialised approaches have shown lower error rates on MNIST, such as using specific convolutional neural nets [19, 20], although the ADN system is arguably a more general machine learning approach. Convolutional DBNs [21] capture some advantages of both, allowing scalability, accuracy and unsupervised learning, however our approach provides a different design angle, with likely advantages in terms of reliability, and the abstract nature allows more flexibility to capture more complex processes seen in visual perception.

One of the important principles of Deep Learning methods is the use of unsupervised learning [1, 2], as this allows the structure of the observed environment to be captured without training examples. The ADN does not directly follow this approach, as the feature population is modified according to reinforcement. The unsupervised aspect of this system occurs in feature generation, as new atomic features are constructed statistically related to the presence of a feature in observations, and composites are created from features activated from observation. As features are reinforced through use the population becomes biased towards useful features however, diverging from unsupervised learning.

The Abstract Deep Network system produces a sparse neural network based on features found from observations. The abstract representation enables functions such as convolution to be introduced, which allows translation invariance, and activation of a feature in multiple positions. The design also allows further processes to be introduced such as top-down influences resulting from context, which may be more easily incorporated than in systems based on RBMs or convolutional neural nets. In human visual processing, a number of influences can be seen in behavioural and neuroscience studies that play an important role in visual perception, and capturing these effects is important for artificial object recognition and understanding visual cognition. The use of an Abstract Deep Network allows these effects to be explored in a self-organising manner, and provides more freedom to study the big picture of processes involved in visual perception, and their benefits for artificial systems.

References

1. Bengio, Y., Courville, A.C., Vincent, P.: Unsupervised feature learning and deep learning: A review and new perspectives. CoRR abs/1206.5538 (2012)
2. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. 2, 1–127 (2009)
3. Rousselet, G.A., Thorpe, S.J., Fabre-Thorpe, M.: How parallel is visual processing in the ventral pathway? Trends in Cognitive Sciences 8(8), 363–370 (2004)
4. Serre, T., Kreiman, G., Kouh, M., Cadieu, C., Knoblich, U., Poggio, T.: A quantitative theory of immediate visual recognition. In: Paul Cisek, T.D., Kalaska, J.F. (eds.) Computational Neuroscience: Theoretical Insights into Brain Function. Progress in Brain Research, vol. 165, pp. 33–56. Elsevier (2007)
5. Weng, J.: A 5-chunk developmental brain-mind network model for multiple events in complex backgrounds. In: IJCNN, pp. 1–8 (July 2010)
6. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
7. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11, 625–660 (2010)
8. Fischer, A., Igel, C.: Empirical Analysis of the Divergence of Gibbs Sampling Based Learning Algorithms for Restricted Boltzmann Machines. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part III. LNCS, vol. 6354, pp. 208–217. Springer, Heidelberg (2010)
9. Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Delalleau, O.: Tempered Markov Chain Monte Carlo for training of restricted Boltzmann machines. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Chia Laguna Resort, Sardinia, Italy, May 13-15, pp. 145–152 (2010)
10. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: a complete introduction, review, and roadmap. J. Artif. Evol. App. 2009, 1:1–1:25 (2009)
11. Knittel, A.: An activation reinforcement based classifier system for balancing generalisation and specialisation (ARCS). In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 1871–1878. ACM, New York (2010)
12. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychological Review 111(4), 1036–1060 (2004)
13. Knittel, A.: Learning feature hierarchies under reinforcement. In: IEEE Congress on Evolutionary Computation (CEC). IEEE (2012)
14. Ebadi, T., Zhang, M., Browne, W.: XCS-based versus UCS-based feature pattern classification system. In: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference, GECCO 2012, pp. 839–846. ACM, New York (2012)
15. Haykin, S.: Neural networks and learning machines. Prentice Hall (2009)
16. Bar, M.: A cortical mechanism for triggering top-down facilitation in visual object recognition. J. Cognitive Neuroscience 15(4), 600–609 (2003)
17. Bar, M.: Visual objects in context. Nature Reviews Neuroscience 5(8), 617–629 (2004)
18. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. 18(7), 1527–1554 (2006)

19. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
20. Ranzato, M., Huang, F.J., Boureau, Y.L., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: CVPR 2007, pp. 1–8 (June 2007)
21. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: ICML, pp. 609–616. ACM, New York (2009)
22. Grauman, K., Leibe, B.: Visual Object Recognition. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers (2011)