Contents lists available at ScienceDirect

Cognition

journal homepage: www.elsevier.com/locate/COGNIT

Flexible shaping: How learning in small steps helps

Kai A. Krueger [*], Peter Dayan

Gatsby Computational Neuroscience Unit, UCL, 17 Queen Square, London WC1N 3AR, United Kingdom

ARTICLE INFO

Article history:
Received 26 December 2007
Revised 9 November 2008
Accepted 25 November 2008

Keywords:
PFC
Gating
Shaping
Sequence learning
Computational modeling

ABSTRACT

Humans and animals can perform much more complex tasks than they can acquire using pure trial and error learning. This gap is filled by teaching. One important method of instruction is shaping, in which a teacher decomposes a complete task into sub-components, thereby providing an easier path to learning. Despite its importance, shaping has not been substantially studied in the context of computational modeling of cognitive learning. Here we study the shaping of a hierarchical working memory task using an abstract neural network model as the target learner. Shaping significantly boosts the speed of acquisition of the task compared with conventional training, to a degree that increases with the temporal complexity of the task. Further, it leads to internal representations that are more robust to task manipulations such as reversals. We use the model to investigate some of the elements of successful shaping.

1. Introduction

Humans and animals acquire an extensive and flexible repertoire of complex behaviors through learning. However, many tasks are too complex to be amenable to simple trial and error learning, and therefore external guidance or teaching is critical. Three central forms of teaching are:

- abstract or verbal communication of symbolic rules governing the underlying tasks;
- repeated demonstration of the required action sequences. This turns unsupervised problems into supervised ones, and in some cases permits imitation;
- shaping, i.e., provision of a simpler path to learning by task simplification and refinement.

Here, we provide a computational treatment of shaping in the context of a cognitive task.

The term shaping was first coined by Skinner (1938), who described it as a "method of successive approximations". In shaping, a sequence of intermediate, simple tasks is taught, in order to aid acquisition of an original, complex, task. Skinner was perhaps motivated by the possibility of taking advantage of animals' innate repertoires of responses (Breland & Breland, 1961; Peterson, 2004); but the term is also used more widely. Divide-and-conquer is inherent in shaping and appears to help with facets of complexity such as branching, hierarchies and large variations in the timescales of the effects of actions. Shaping also provides behavioral "units" that can be deployed in a range of tasks. Shaping is almost ubiquitous in animal behavioral experiments.

Two main aspects of shaping have been considered in theoretical learning frameworks. First, Elman (1993) realized a concept described by Newport (1988, 1990) in terms of "Less is More", in the context of the learning of grammars in simple recurrent networks. The idea was to use an initial phase of training with only simpler incarnations of the rules of the grammar. Although the issue is not uncontroversial (see Rohde & Plaut, 1999), Elman (1993) argued that this simplification may arise intrinsically, through a process of self-shaping. This would happen on the basis of a working memory that, consistent with evidence for differential rate of maturation of parts of the prefrontal cortex (Brown, Joseph, & Stopfer, 2005; Sowell, Thompson, Holmes, Jernigan, & Toga, 1999), initially has a very low capacity, but expands over development. The

[*] Corresponding author.
E-mail address: kai.krueger@ucl.ac.uk (K.A. Krueger).

0010-0277/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.cognition.2008.11.014

second aspect of shaping that has been studied is associated with robot learning or reinforcement learning (Dorigo & Colombetti, 1998; Savage, 1998, 2001; Saksida, Raymond, & Touretzky, 1997; Singh, 1992), typically in the context of navigation.

By contrast with these suggestions, we consider shaping for the adult learning of the sort of complex cognitive tasks that are popular for the elucidation of the prefrontal neural architecture of cognition (Shallice & Burgess, 1991; Gilbert, Frith, & Burgess, 2005; Koechlin, Ody, & Kouneiher, 2003; Badre, Poldrack, Pare-Blagoev, Juliana amd Insler, & Wagner, 2005). The main emphasis in the computational modeling of these tasks has so far been in developing architectural mechanistic elaborations (Frank, Loughry, & O'Reilly, 2001; O'Reilly & Frank, 2005), to overcome the complexity of learning. However, shaping is extensively used in training human and animal subjects in order to simplify complex learning; here, we seek to model it and understand aspects of its power.

O'Reilly and Frank (2005), Hazy, Frank, and O'Reilly (2007) suggested one of the most powerful and effective architectures in their prefrontal, basal ganglia, working memory (PBWM) model. This employs a gated working memory (adapted from the long short-term memory (LSTM), architecture of Hochreiter & Schmidhuber, 1997; Gers, Schmidhuber, & Cummins, 2000) in an elaborate overall structure. O'Reilly and Frank (2005), Hazy et al. (2007) illustrated their model using an abstract, hierarchical, version of the continuous performance working memory task (CPT) called the `12-AX` task, which they invented for the purpose. The complexity of this task arises from its hierarchical organization, which involves what amounts to subroutines.

Here, we build an unelaborated LSTM model (which O'Reilly & Frank, 2005; Hazy et al., 2007, used as a comparison point for the learning performance of PBWM) and study the additional role that shaping might play in generating complex behavior in tasks such as the `12-AX`. We consider a straightforward shaping path for this task, highlight the importance of the allocation of resources in shaping, and assess the improvements in training times that come from the external guidance, as a function of parametric task complexity. Finally, we look at the effects of shaping on the flexibility of the network in dealing with variations of the stimulus statistics (while keeping the rules constant), and with a shift in the task rules themselves.

2. General methods

In this section, we describe the `12-AX` task, the unelaborated LSTM network used to solve it, the particular shaping path that decomposes the task into its elements, and the learning methodology. One of the most important questions in shaping is how to increase the capacity or power of the network as new elements of a task are presented. In order to focus cleanly on the effects of shaping, the main results in Section 3 depend on *manually allocating* new LSTM components at each additional step of shaping. However, in Section 4, we show results from a simple (uncertainty-based, Yu & Dayan, 2005; or error-based, Zacks, Speer, Swallow, Braver, & Reynolds, 2007; Reynolds, Zacks, & Braver, 2007) scheme for *automatically allocating* these components. This proves that shaping can still be effective without extra external intervention.

2.1. The `12-AX` task

The `12-AX` task (Fig. 1) is a complex problem involving inner and outer loops of memory and control (signalled by numerical, and particular alphabetic, inputs, respectively). In the task, subjects see a sequence drawn from an alphabet of the eight symbols `1`, `2`, `A`, `B`, `C`, `X`, `Y`, `Z`; every symbol has to be followed by a response. The 'target' key ('R') must be pressed for symbols defined as targets by the rules of the task, and the distractor key ('L') for all other symbols. There are two different inner loops, both of which are CPT 1-back tasks: subjects must declare as a target *either* `X` when preceded by `A` (i.e., to the segment `AX`) *or* `Y` when preceded by `B` (`BY`). These pairs appear without warning in a stream of uniformly-selected random distractor pairs. Every symbol not defining the end of a target pair should be declared as a distractor. The outer loop is signalled by the numbers, with a `1` meaning that the `AX` task should be performed until the next context marker; and a `2` meaning that the `BY` task should be performed instead. The numbers `1` and `2` themselves should also be declared to be distractors ('L').

Different statistics and contents of the inner loops define different variants of the task. For ease of comparison, we work with the version defined by O'Reilly and Frank (2005) unless otherwise specified. In this, each random pair consists of one of {`A`, `B`, `C`} followed by one of {`X`, `Y`, `Z`}, and there are $n = 1 \ldots 4$ pairs in each outer loop. The outer loops are equiprobably `1` and `2`. At least 50% of the inner loops consist of potential target sequences `AX` or `BY`. The other 50% are drawn uniformly from all 9 possible inner loop sequences. An epoch is (arbitrarily) defined as 25 outer loops, and network activity is reset after each epoch.

O'Reilly and his colleagues (Frank et al., 2001; O'Reilly & Frank, 2005) defined successful acquisition of the task by the absence of errors in two consecutive epochs. In our simulations, we find this not to be sufficient, as a substan-

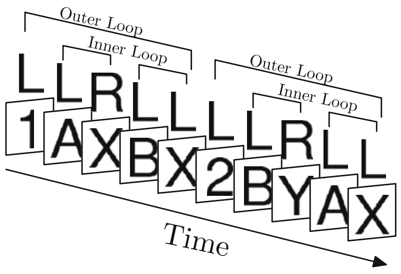

Fig. 1. The `12-AX` task. Each symbol within a rectangle represents the stimulus presented to the network at a single timestep. Required responses are indicated by the letters above ('L', distractor, and 'R', target). The outer loop, beginning with either a `1` or `2`, determines the correct target sequence (`AX` or `BY`) for the following *n* duplets (inner loops). Adapted from O'Reilly and Frank (2005).

tial number of errors can be made even after reaching this criterion. Instead, we use a softer, but more prolonged criterion, requiring networks to make no more than 5 errors in 30 consecutive epochs, reducing the error rate to 0.5 errors in a thousand responses. These 30 epochs are excluded from the reported training times. Experiments are repeated 100 times with different random weight initialisation and stimulus sequences on each repetition.

2.2. The network

Our network is an unelaborated form of the long short-term memory (LSTM) model with forget gates (Gers et al., 2000). This model derives from standard recurrent neural networks, but with the addition of a gating mechanism to facilitate long term activation based memory in its recurrent layer, while still allowing for rapid updating and read-out. LSTM provides the original functional foundation of O'Reilly and Frank's (2005) PBWM.

Fig. 2 shows the structure of the model and in this section we describe an overview of the important elements of the model. The detailed equations are included in Appendix A. The LSTM network can be decomposed into three layers: input, recurrent or memory, and output layer (from bottom to top). The output layer is a standard connectionist neural network layer, i.e., a linearly weighted sum of inputs with a sigmoidal activation function. The activations of the input layer are set to represent the 'percepts' of the network and are represented as a unary encoding of the stimulus alphabet. These are fed forward to both the recurrent memory layer and directly to the output layer.

The recurrent layer consists of several so called memory blocks. Each memory block (the inset in Fig. 2) is associated with a set of three gating units, input, forget and output gates. In addition each memory block has several memory cells, which store the actual memory (depicted by the middle circle of the cell in Fig. 2) and whose activation can range from positive to negative real values (for reasons of numeric overflow in the output nonlinearity bounded between -20 and 20). At each timestep the activation of a memory cell is decayed multiplicatively by the activation value of the forget gate of the corresponding memory block. Thus the forget gate controls smoothly between forgetting the memory completely (value of 0) and retaining perfect memory (value of 1). Similarly, the input to the memory cell, a linear combination of the input layer and the outputs of all of the memory cells passed through a sigmoidal nonlinearity ranging from -1 to 1, is multiplied by the activation value of the input gate before being added to the activation of the memory cell, thus allowing inputs to be differentially ignored. Finally, the output of a memory cell is passed through another sigmoidal nonlinearity and multiplied by the activation of the output gate unit.

All sets of weights, i.e., for the output, cell input and gating units (depicted by triangles in the figure), are fully plastic. Altogether, this allows the network to learn to select

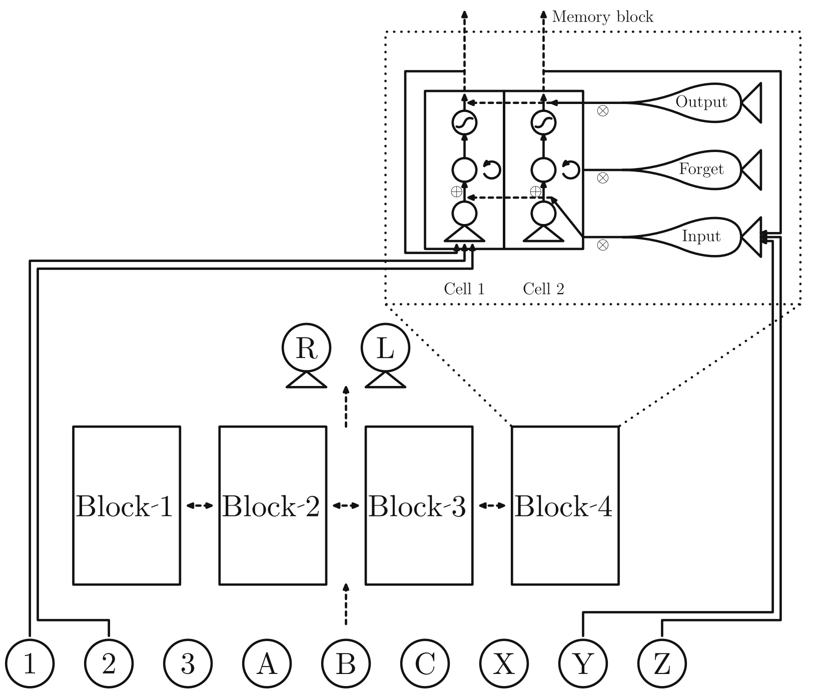

Fig. 2. The network model. The long short-term memory (LSTM) network with 'forget' gates. Stimuli are encoded punctately in the input layer of the network using binary units that each represent one element of the alphabet. The number of input units used varies between 9 (for the main `12-AX` case) and 18 (for both shaping and reversal tasks). All networks have two output units representing an 'L' or 'R' decision of the network. The hidden layer is comprised in most simulations of 4 recurrently connected memory modules or blocks, although in some experiments it can be more or vary, especially during automatic allocation. Feed–forward weights connect the layers. A close-up of a memory block is shown, visualising two cells that are gated by common multiplicative input, output and forget gates. Each triangle represents a set of linear weights both for the connection from the input layer to the memory cells and gating units, and from the gated output of the memory cells to the output layer.

between rapid updating and robust memory retention wherever required. However, the LSTM contains no direct competitive component on the output layer, and thus any anti-correlation between its units has to be explicitly learned through the weight vectors of the output units. As the output of the task is an explicit decision of either "Left" or "Right", the graded activations of the outputs are binarized at a threshold of 0.5. In cases in which both output units are the same (either 0 or 1), the network is counted as giving an erroneous response.

Learning in this architecture is performed in a fully supervised fashion using a backpropagation through time (BPTT) variant of gradient descent. This minimizes the squared error between the target, a unitary encoding of the desired action, and the output units.

2.3. Shaping procedure

Key to shaping is identifying the essential subcomponents in a task. These can then be separated to produce an appropriate training sequence. In case of the 12-AX task used here, the two main hierarchical components of the task are: (i) learning to memorize 1 and 2 for long periods as context markers; and (ii) learning to memorize the A or B for one step to perform the AX or BY blocks correctly.

Although the exact details of the shaping protocol are somewhat arbitrary, they do adhere to the above principles. As shown in Fig. 3, we consider a 7 stage shaping procedure divided into 3 main sections, each of which reflects the structure one of one of the subcomponents: (i) learning to store the context markers 1, 2 (stages 1-4); (ii) learning the one back characteristics of the CPT–AX (stages 5 and 6); and, finally, (iii) the full 12-AX task (stage 7).

In the first section of shaping, the networks are exposed to a task whose response is only defined by the last seen number. That is, all stimuli following a 1 require an 'L' response, whereas those following a 2 require an 'R' response (see Fig. 3a). The alphabet of possible intermediate stimuli is chosen to be a distinct set of nine further inputs that are not part of the standard 12-AX task, in order not to confound learning the storage of 1 and 2 with other aspects of the task. Note, however, that in other experiments (not shown) which used the same alphabet here as for the full 12-AX task, ultimate performance was essentially the same, so this is not a crucial parameter of the procedure.

Gating in the LSTM model is graded rather than binary (partly to ensure differentiability of the network). Making this gating be sufficiently strong requires the use of at least some long sequences, in our case including up to 60 intermediate stimuli between successive context markers. Under the shaping procedure (and indeed rather deeply embedded in the strong asymmetry in the task as a whole between the frequency of distractor and target responses), this produces very long stretches of identical responses, which themselves harm learning. Thus, to achieve acceptable performance, we segment training by presenting this section of the task in four parts (defining the first four stages of shaping), with loop lengths increasing from 12 up to the maximum of 60. Within each individual stage, the distribution of lengths follows a (renormalized) truncated exponential distribution, with the longest possible sequence being more than five times less likely than the shortest.

The second section of shaping consists of two stages, each based on remembering one of the first elements of one of the two target sequences AX and BY. Unlike the full 12-AX task, during these two stages any symbol following an A or a B respectively requires a target response. Typical sequences are shown in Figs. 3b and 3c. The final section (the seventh stage of shaping) involves learning the full 12-AX task.

Each stage is trained until the network achieves the performance criterion on that stage, or for a maximum of 500 epochs. This limit is implemented both for computational reasons as well as to remove the odd outlier network that fails to learn in a reasonable time. This limit corresponds to about three times the mean training length equivalent to several standard deviations larger than the mean of the unshaped base case.

2.4. Manual allocation

Shaping involves separate phases of training sub-tasks. As mentioned, throughout the experiments in Section 3, new network resources, i.e., new memory blocks, are allocated by hand when encountering new subtasks, ensuring the necessary separation of learned behavior across shaping stages. Memory blocks in the LSTM network are mostly independent, and so can readily be treated as separate units of resource. In Section 4, we discuss a method for automatically allocating new blocks.

For the results in Section 3, for each new sub-task during shaping, a single fresh memory block is allocated (having random initial weights), and previously used memory blocks are temporarily disabled. This results in a strict separation of "behavioral units." Each time the allocation changes, weights from the input layer and the recurrent memory layer to the output layer are reset to random

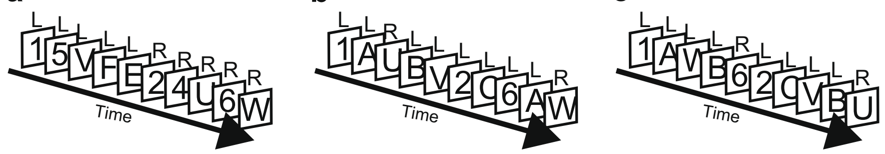

Fig. 3. Typical sequences used during the shaping procedure. (a) The first section of shaping trains the context markers 1 and 2. Responses to all stimuli following a 1 should be R and to those following a 2 should be L. The maximum length of distractor sequences gradually increases in 4 stages. (b), (c) The second section trains the unconditional gating of A and B, respectively, in a one back task. The subject has to respond unconditionally with an R to any input immediately following an A in (b) and B in in (c).

values to encourage learning into the new module. In the final stage of learning the complete 12-AX task, all four memory (three of which were present during shaping and a fourth empty block) are fully enabled, and all weights are plastic. This allows a fair comparison between the learning of the shaped network and learning of a topologically-identical, but randomly-initialized, unshaped network.

3. Results

As a baseline for performance, we train a standard (naive) LSTM network directly on the full 12-AX task. On average it acquires the task in 186 epochs. (standard error: 8.5 epochs) This is rather faster than the roughly 350 epochs that O'Reilly and Frank (2005) suggested for an unembellished LSTM network. Since the number of memory blocks (4), cells per block (2) and learning rate (0.1) are similar to those they used, this presumably results from the altered learning criterion and network parameters.

By comparison, after complete shaping, networks learn the full 12-AX task rapidly, on average in 39 epochs (standard error: 6.6), thus showing the expected large decrease in training time. As the mean can be corrupted by a few outliers, the typical learning times (median) may be more informative. The median for the shaped networks is 14 epochs; for the unshaped network the median was 162 epochs, closer to the mean. Thus median training times showed even greater advantage with about a ten fold decrease through shaping.

Of course, the full time for training should also include that devoted to the shaping itself. Calculating the equivalent number of epochs for the shaping stages based on the number of stimulus presentations, shaping takes an equivalent of 74 epochs on average (median is 61 epochs). Thus, as can be seen in Fig. 4, there remains a substantial overall benefit (significant at $p < 0.001$) for shaping. Fig. 5 shows detailed (averaged) learning curves of each of the individual stages of shaping.

To prevent any occasional bad runs from unduly corrupting average training times, outliers in the form of runs failing to learn any of the stages within 500 epochs were excluded from further analysis. This happened for both the shaped and unshaped network in about 5% of the test runs performed to gather statistics.

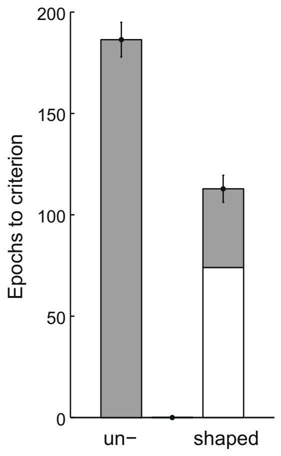

Fig. 4. Comparison of learning times between shaped and unshaped networks for the standard 12-AX task. The bars show the numbers of epochs needed to learn the task up to a set performance criterion. Grey represents the epochs trained on the 12-AX task. For the unshaped network this is the sole training received. White quantifies the cumulative training time of the shaping procedure in terms of 12-AX epoch equivalents. Error bars represent the standard error of the mean.

3.1. The necessity of resource allocation

As described in Subsection 2.4, we allocate the resources of the network, i.e., the LSTM modules, by hand during shaping, to ensure separation of the "behavioral units". We might expect allocation to play a critical role in solving the stability–plasticity dilemma (Grossberg, 1980) inherent in shaping, and so for performance without it to be severely impaired. Indeed, the idea of resource allocation, or more broadly the concept of increasing the

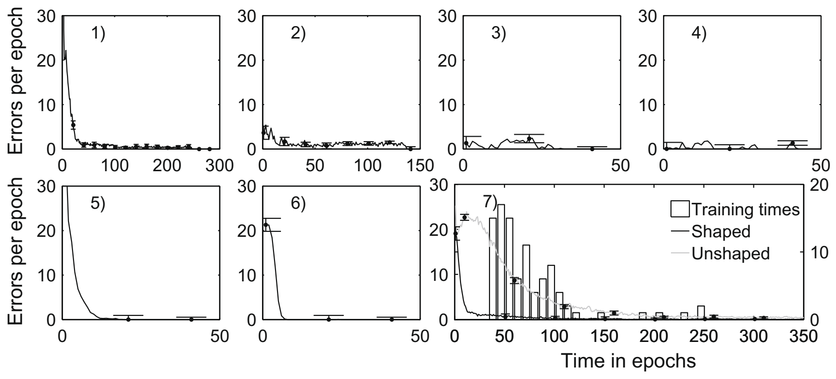

Fig. 5. Graphs show average learning curves during the individual stages of shaping. The top row shows section one training of the context markers, each presenting training times of increasing length of 12, 25, 40 and 60, respectively. The bottom left two plots show training on the unconditional one back A/B tasks. The last plot shows training times on the full 12-AX task. The dashed line shows average learning for the unshaped network. The histogram (scale on the right) visualises the distribution of times required for the 6 stages of shaping. Plots follow each other sequentially. Error bars represent the standard error of the mean, and are only shown for selected points for clarity.

capacity of a learning agent in the event of sudden sequential change in the task to prevent interference of learning has been proposed previously as a solution to similar problems, including Redish, Jensen, Johnson, and Kurth-Nelson's (2007) reinforcement learning model of extinction and renewal learning, which we discuss below.

One way to test the importance of resource allocation for shaping is to perform the same learning procedure (the shaping stages followed by the full 12-AX task) but with all the memory modules being fully active throughout learning. Doing this on average required 235 epochs (median 227 epochs) for the final stage (the 12-AX task) alone. This is longer not only than the network which does involve resource allocation, but also than the baseline case of the unshaped network. This problem is only partially solved by increasing the capacity of the network, which should reduce the pressure to find one specific solution. Although testing a version with twice as many memory blocks showed a slight reduction in the number of epochs for the final 12-AX stage (mean 210, median 200), doubling the number of memory units once more to give four times the initial capacity, led to worse results again.

3.2. Robustness to irrelevant additional structure

That 12-AX is the first task seen by our network is another potential confound, since, in reality, this task is likely to be only the latest in a very large set of tasks that the subjects will face. The possibility of generalising from these previous tasks to produce even better performance is an important, but hard, question that we discuss later, but cannot yet simulate. However, it is straightforward to test the robustness of the learning procedure to extra irrelevant behavioral units associated with other, independent, tasks.

We consider two possible confounding structures, both involving extra, irrelevant, LSTM memory units. Shaping remained identical, with all extra modules fully disabled during the first six stages of 12-AX task shaping, but enabled and plastic during the final, complete, 12-AX stage. One set of memory units comes from solving a similar task defined on the same alphabet of symbols, but with different context and block markers. The other comes from using LSTM modules with random weights drawn independently from exponential distributions matched to the marginal distribution over weights as arises during normal 12-AX training. Distributions are matched separately for the different classes of weights.

In neither case does learning performance using shaping of the 12-AX task differ much from that of the simple shaped network (means 25, 19 and medians 12 and 10, respectively, for similar and random structures).

3.3. Scaling behavior

The ability to cope with increasingly complex tasks and to identify and learn patterns even over extended time periods is a key aspect of cognitive flexibility. However, long temporal credit assignment paths render rapid learning infeasible for many traditional neural network learning algorithms. By contrast, we predict that a network containing the sort of additional task information associated with learning through shaping, should experience fewer problems.

In this case, the most important form of temporal complexity is the number of inner loops contained within each outer loop. We can therefore test the prediction about the benefit of shaping by training both shaped and unshaped networks on 12-AX tasks with outer loops varying in length from 1 to 40 sequence pairs, i.e., 2–80 interleaving stimuli. Since the more complex tasks take longer to converge, we also raise the cut-off criterion from 500 to 1000 epochs. All other parameters, including the shaping procedure, are identical to those described above.

Fig. 6 confirms that whereas the training time of the unshaped network rises very steeply with longer outer loops (up to 715 epochs on average for 40-length outer loops), the shaped network actually uses *fewer* epochs. This apparently paradoxical result comes from the fact that the number of pattern presentations per epoch increases (from 75 to 1050 on average). This shows the unshaped network in a particularly unfavourable light.

3.4. Computational generalisation

Another critical issue is the ability of the procedure to create appropriate computational mechanisms that can generalize along appropriate dimensions by effectively abstracting the statistics of the task away, focusing only on the underlying rules.

In this case, the key dimension is again the length of the outer loop, and so we test the ability of the networks, having learnt from outer loops of one length (up to 4 inner loops), to generalize to longer outer loops, without further learning. The rules of course remain the same for each of these tasks. Thus, if they are represented abstractly, in a way that generalizes proficiently, then the only extra errors should come from the requirement to retain working memory for more extended periods. Therefore, this parameter manipulation acts as a proxy of one type of rule abstraction. Note that this is a quite different question from that in Section 3.3, which concerned learning rather than generalisation.

Fig. 7 shows that the error rate of the unshaped network increases much more sharply than that of the shaped network, particularly for long outer loops. Fig. 8 breaks this down by the number of inner loops since the last context marker (i.e.,'1' or'2'). The devastating sensitivity of the unshaped network to this factor is starkly evident. Shaping creates a more abstract computational mechanism that generalizes appropriately beyond the original training.

3.5. Symbol generalisation

A more direct measure of the generalisation capabilities of the network is its ability to handle previously unseen test sequences. To test the shaped and unshaped networks in this way, we withheld two of the inner loop sequences during training on the base 12-AX task. After successful training to criterion on this reduced task, we measured the error rate on the full task. In order to determine a reliable average error rate, learning in the network is disabled for all cases during the test phase.

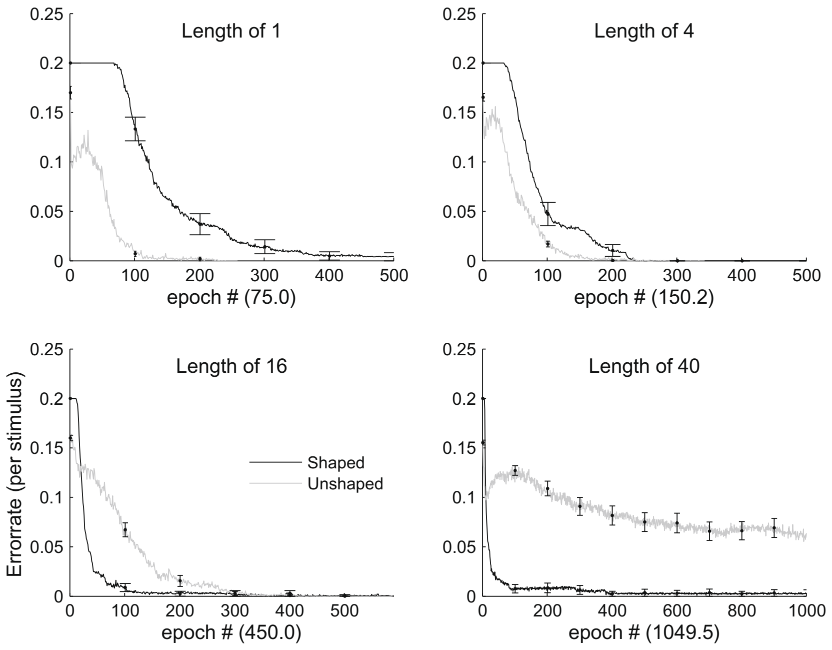

Fig. 6. Shaping as a function of complexity: Each plot shows learning curves for the `12-AX` task with loop lengths of 1, 4, 16 and 40. Training times of the shaped network are offset by the number of steps (in average epochs) needed to learn the complete shaping procedure. Average numbers of presentations per epoch are indicated in brackets below each plot. The performance of the unshaped network is shown by the dashed line, the shaped network by the solid line. Error bars represent the standard error of the mean.

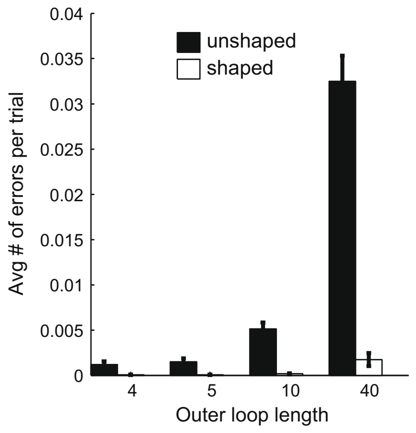

Fig. 7. Computational generalisation: Shaped and unshaped networks are trained on the base `12-AX` task with a maximum loop length of 4. Without further training, performance is measured on the extended `12-AX` tasks. The loop length of 4 is included as a control and represents the successful training criterion. This graph shows the average number of errors per presentation on each of the different loop length tasks.

As there is only a very limited number of sequences (nine) altogether, the choice of which sequence is left out can significantly influence the results. Obviously the two target sequences `AX` and `BY` cannot be left out, as these are unique and therefore not possible targets of generalisation. Furthermore, each stimulus is only present in 3 sequences. Hence, if two of these containing the same stimulus are withheld, then again, no generalisation is possible. Therefore, we choose to withhold `AZ` and `CX`. Even so, only two exemplars are left according to which rules can be inferred, making this test quite hard. With `AZ` and `CX` withheld, the remaining non-target sequences are `AY CY CZ BX BZ`. Therefore, when looking at the `AZ` sequence, the `A` appears once in a target and once in a non-target sequence, whereas the `Z` never appears in a target sequence. The `CX` sequence in contrast has the opposite weighting, with the first element of the sequence, the `C`, never being part of a target and the `X` being part in half of the sequences. This choice of withheld patterns therefore allows us to identify if there are differences as to which element is more important for generalisation to a network.

The results of this experiment are presented in Fig 9. First, when trained on the full `12-AX` task, i.e., with no pattern being withheld during training, neither shaped nor unshaped networks have large variations across individual inner loop pairs, showing they have indeed learned the task correctly. This is to be expected. By contrast, those networks trained on the restricted training set show more substantial variations. Although both training methods end up performing worse on the withheld data, either shows generalisation and on average only makes on the order of one in 10 mistakes for the withheld sequences. Nevertheless again the shaping seems to benefit and the overall error rate is down compared to the unshaped training. More interestingly however, the patterns of errors on the withheld patterns differ significantly between the shaped and unshaped networks. Whereas the unshaped network frequently wrongly generalises the `CX` pattern as a target, shaping reduces this type of error. In contrast however, the shaped network generalises worse on the `AZ` pattern,

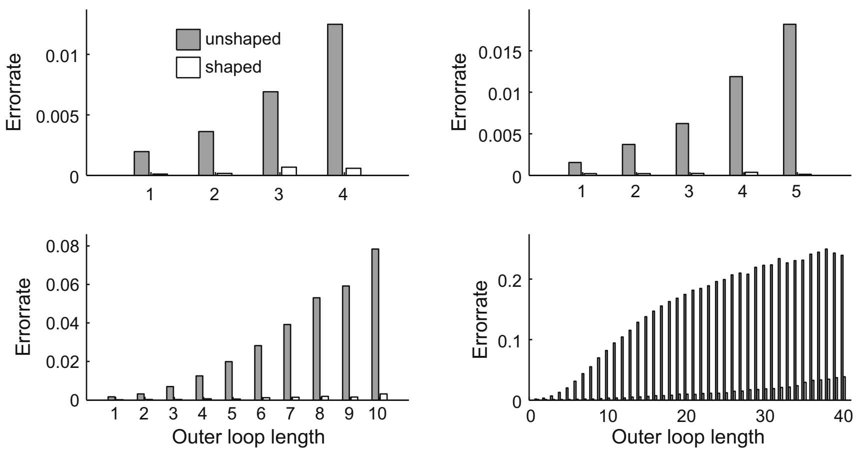

Fig. 8. Computational generalisation: The graphs show the probabilities of making a mistake on a stimulus *n* duplets into the outer loop (normalized to the number of presentations of each kind) for shaped and unshaped networks tested on outer loops of lengths 4, 5, 10 and 40.

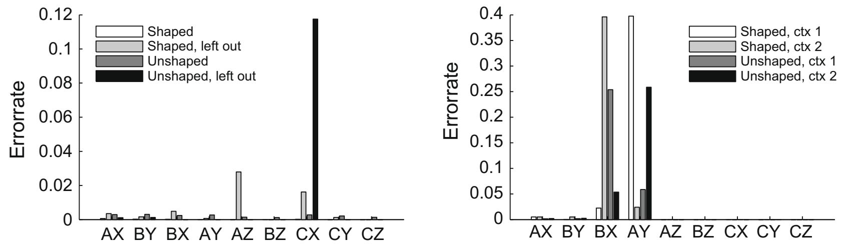

Fig. 9. Symbol generalisation: These graphs show symbol generalisation in the form of steady state error rates after training on a reduced set of candidate sequences. (a) Candidate sequences AZ and CX were never presented during training. For comparison, the results of the fully trained 12-AX are included. (b) Candidate sequences AY and BX are not shown during training. Errors are separated according to the context in which they occur.

occasionally classifying it wrongly as a target. Here the unshaped networks have no problems.

By symmetry, withholding sequences BZ and CY should lead to the same result. Indeed, shaped networks made more errors on BZ (shaped: 2.5%; unshaped: 0.0%) whereas the unshaped networks made more errors on CY (shaped: 3.4%; unshaped: 11.4%).

These results suggest that the training method influences the nature of generalisation, with the shaped networks over-emphasising the first element of the sequence and the unshaped networks generalising along the second element of the sequence. The extensive pretraining on A and B during shaping leads to a prominent representation of these stimuli in the network, and so determines the nature of the generalisation.

To verify this interpretation, we ran a second set of experiments in which a different class of sequences was withheld (AY and BX). By contrast with the previous sequences, both the first and second stimulus of the sequences are each part of one of the target sequences. Thus, from the analysis above, we would expect both types of networks to have problems with generalising these sequences. However, differences can still be seen coming from the nature of the context dependency of the 12-AX task. As shown in Fig. 9(b), when splitting the errors by context, the varying types of errors can easily be seen. The shaped network again generalises along the A and more often incorrectly attributes the AY a target in the 1 context and the BY in the 2 context. Conversely, the unshaped network more often incorrectly responds to the AY in the 2 context and the BX in the 1 context. Overall, for both networks, the error rate is much higher with this set of withheld sequences, as each part of the sequence during training is only part of one target and one non-target sequence making generalisation more ambiguous.

3.6. *Reversal learning*

An important experimental test of flexibility concerns the effect on the speed of learning of successive alternating reversals in the contingencies in the experiment (Butter, 1969; Iversen & Mishkin, 1970; Jones & Mishkin, 1972). We therefore define a reversed task (AB-X1), in which the rules are the same, but the context markers and the inner loop target sequences are upside down. We compare the particular impact of reversals with that of alternating learning of a new task called 45-DU (shifting), whose rules are the same, but involving different, non-overlapping, symbols.

3.6.1. *Reversal tasks*

These set of experiments involve two new tasks, the AB-X1 task for the reversal experiment and the 45-DU task for the shifting experiment. All three tasks share the same rules (either one of two target sequences is active, depending on the most recent context marker) and thus have the same hierarchical nature as the 12-AX task, however which stimulus is a context marker and which belongs to one of the target sequence differs between the tasks. In the AB-X1 task, the same alphabet

(1, 2, 3, A, B, C, X, Y, Z) is used with A and B being the new context markers and X1 and Y2 the new target sequences. The 45-DU task instead employs a non-overlapping alphabet (4, 5, 6, D, E, F, U, V, W) with 4 and 5 the context markers and DU and EW the respective target sequences. By the fact that the AB-X1 uses the same alphabet as the 12-AX task, and the tasks are learned sequentially the switch between these two tasks requires a certain amount of unlearning or at least suppression of the stimulus action mappings. In contrast the non-overlapping alphabet of the shifting task results in less interference due to the strong external indicator of the used stimulus set.

In order to accommodate these additional tasks, some changes to the network and the shaping procedure are required. First, the new symbols require the input layer to be extended by further 9 units. Second, in order to allow the shaped network to be able to build upon prior structure during reversals in the same way as in the base 12-AX task, the shaping procedure is extended to include equivalent training for the AB-X1 task. The additional shaping is performed using 4 additional memory modules, so the network had a total of 8. For a fair comparison, the unshaped network is also given 8 modules.

The full shaping procedure of components of both tasks is completed before the first exposure to either the 12-AX or the AB-X1/45-DU task, by which time resource allocation is finished and all memory cells are fully enabled. Thus, at the time of first exposure to the full tasks, the shaped and unshaped network are again identical apart from the structure in the weights established through shaping. Further, as always, the activations of the network are reset at the end of each epoch, whether or not a reversal occurs.

The actual reversal task consists of 5 reversals between 12-AX and AB-X1. A reversal occurs either after a network has reached its performance criterion of learning the task, or after a maximum of 500 epochs. The time of reversal is not cued in any way.

3.6.2. *Reversal results*

The large panels in Fig. 10 show averaged learning curves for the first 6 reversals. Networks with either training method show substantial difficulties with learning the initial reversal of the task. In fact, the difficulties are such that a large fraction of the training runs fails to achieve the learning criterion within the preset number of training epochs (500). Even for the shaped cases this fraction is at about 50%; however, performance of the unshaped network is particularly devastated by the reversal, with close to all (95%) of the networks failing to converge appropriately. In the graphs showing the error rates averaged over all runs, bar those few failing to learn even on the initial 12-AX, this failure to learn can be seen in the elevated asymptotes; a somewhat orthogonal feature to the steepness of the initial learning. Nevertheless, over the successive reversals, both networks improve in both measures; but the shaped networks still significantly out-perform the unshaped networks throughout each AB-X1 task. Shaping does not, however, eliminate the path dependence of the learning of the full task (given the lack of resource allocation at this point); there is typically a very fast phase of initial learning each time the 12-AX task repeats.

In the shifting experiment (45-DU task), compared with the above reversals, switching the contingencies does not lead to such poor performance. The small panels of Fig. 10 show that there is a clear learning effect already by the second shift. By the third shift, both shaped and

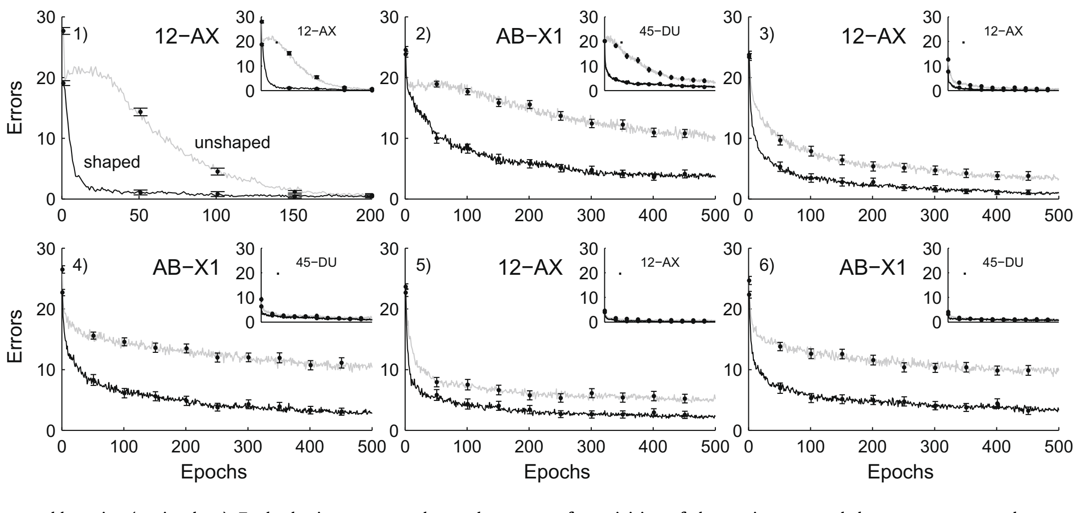

Fig. 10. Reversal learning (main plots): Each plot in sequence shows the course of acquisition of alternating reversals between 12-AX and AB-X1 versions of the task, for shaped and unshaped networks. No changes to the networks and their weights are performed between tasks. The shaping procedure preceding the first presentation of the 12-AX task is not shown. The inset plots show the equivalent for shifts between 12-AX and 45-DU tasks. Error bars show the standard error of the mean.

unshaped have mostly learned the shift and perform well immediately after the switch. However, again, the shaped network does substantially better with the median learning time of 1 epoch for both the last shifts, compared to 14 and 16 for the unshaped network.

4. Automatic allocation

In all the simulations so far, we have employed a *deus ex machina*, in which shaping has been associated with the manual allocation of resources. Indeed, Section 3.1 showed that shaped networks perform substantially worse than the unshaped networks without allocation. We did this to focus on the potential benefits of shaping in computational modeling, rather than particular implementation details. However, if there was no way of realizing these benefits without such an external intervention, then shaping would not be a viable solution. Thus we now suggest one simple mechanism for automatic resource allocation, as a proof of principle.

4.1. Unexpected uncertainty as a trigger for resource allocation

A simple possibility for automatic allocation is to specify a mechanism that detects the onset of each new sub-task. An obvious candidate for this is unexpected uncertainty (Yu & Dayan, 2005; Dayan & Yu, 2006), i.e., the unexpected drop in the performance of the network that happens when each new sub-task is introduced. Indeed, unexpected uncertainty, and its noradrenergic neural representation, have been been implicated in the nature and speed of learning in reversal tasks, which involve similar contingencies. Another example of using a form of unexpected uncertainty to detect boundaries is the model of Zacks et al. (2007), Reynolds et al. (2007). In this model of event segmentation in a sequential perception stream, segmentation is based upon the assumption that the predictability of the next percept significantly drops at the boundaries of events, beyond the expected uncertainty within the event, driving a gating signal in working memory. Equally, in the work of Redish et al. (2007), the contingency of extinction is detected by a sudden drop in reward, at which point the state space is duplicated and thus new resources are allocated.

Along similar lines, we propose that times of heightened unexpected uncertainty trigger events in the network. Rather than only affecting network activity through gating, however, we propose a change of the network topology itself (hard resource allocation) or more realistically through meta-plasticity in the form of differential changes in learning rates (soft resource allocation).

For the want of a full model of uncertainty in these simulations, we make the assumption (which in this case is true) that, having learned the deterministic 12-AX task, an agent would make no errors. Thus, the expected uncertainty is close to zero. As such, the current error rate can be treated as a direct measure of unexpected uncertainty. We model allocation as being triggered by a threshold crossing of unexpected uncertainty or error rate increase, a technique both Redish et al. (2007) and Reynolds et al. (2007) employed in their models. As in Section 3, when allocation occurs, a single new module of the LSTM gets activated. However, unlike for manual allocation, the plasticity of the existing modules is strongly reduced (by a factor of 20) rather than completely abolished, and indeed these modules continue to contribute to network activity in a normal manner. This is essential for a solution to a task to involve a recombination of elements of the solutions to previous tasks.

Although according to the model, the number of modules should be able to grow arbitrarily large, for practical reasons we cut-off new allocation after 12 units, which happened once in the hundred runs.

As no outside intervention was performed in the network during the automatic allocation experiments, the network topology at the time of first encounter with the 12-AX is no longer identical to either the unshaped network or the manually allocated shaped networks. Not only may the number of memory blocks differ due to the continuing allocation of new blocks, but also, once performing the 12-AX task, not all blocks are equally plastic. Indeed, there is no longer any distinction between epochs involving shaping and those involving the full 12-AX task. Appendix B provides details of this procedure.

4.2. Results

In order to test the effectiveness of this simple automated allocation mechanism, we repeated the basic shaping experiment. Fig. 11 shows that this network is a clear improvement on a network without any allocation. Although the mean number of epochs required for learning (173 epochs for the combined training, 87 epochs for 12-AX task alone) is greater with automated than manual

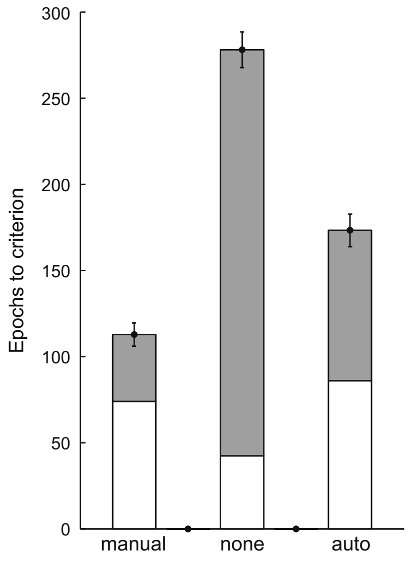

Fig. 11. Automatic allocation: This graph shows learning times for the shaped 12-AX task given manual allocation, no allocation, and automatic allocation. As above, the white bars show the epochs devoted to shaping; the grey bars show those devoted to the full task.

allocation, it is still less than for learning without shaping ($p < 0.05$, single tail t-test). Furthermore, much of this apparent decreased learning speed originates from slightly less robust learning, which makes it harder to achieve the stringent performance criterion. This can be seen particularly clearly in the actual learning curve (Fig. 12-7), which differs little from the case of manual allocation in Fig. 5, and shows a large improvement over no shaping. The reduced robustness is apparent in the heavier tail of the learning curve.

Another way of analysing the effects of the automatic allocation is to look at the distribution of points at which new blocks were allocated. On average, in each run, 4.5 allocations occurred. The majority of runs contained 4 allocations. Although only 3 allocations were necessary, and thus all but one run showed at least one spurious allocation, the majority of these occurred during the early part of the first stage of shaping, at which point the error rate was still highly variable. Looking closer at the allocation during the final stage of learning the full 12-AX task, we can see that all runs contained at least one allocation during this stage and 76 runs detected the task switch within the first epoch of changing to the 12-AX task.

In a second set of experiments, the reversal tasks of Section 3.6.2 were repeated using the automated allocation procedure. Here, the automatic allocation actually performs *better* than manual allocation. This comes from its ability to allocated new blocks throughout the reversals, thus helping separate out modules associated with 12-AX and AB-X1 tasks (see Fig. 13).

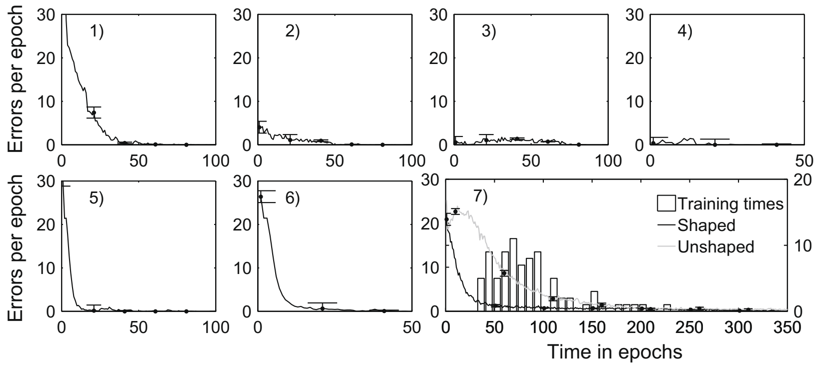

Fig. 12. Automatic allocation: Averaged learning curves for shaping with uncertainty-based allocation. The structure of the graphs is the same as in Fig. 5. (Panels 1–6) show the stages of shaping. (7) Learning the full 12-AX. Error bars represent the standard error of the mean.

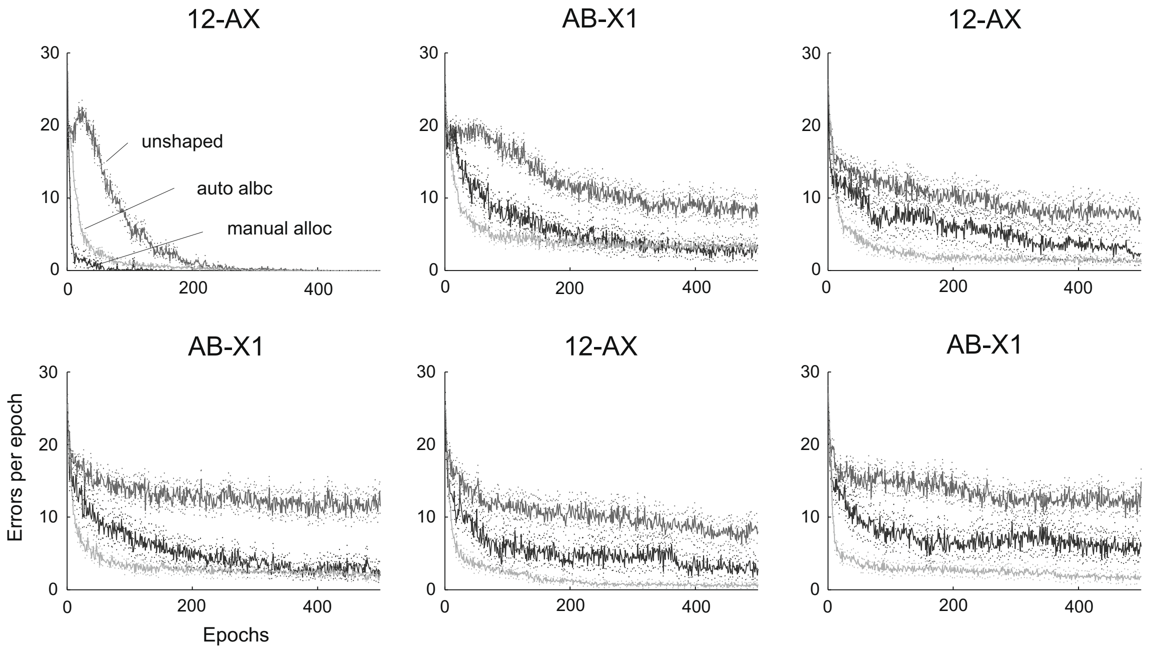

Fig. 13. Automatic allocation: Reversal learning with the help of allocation. This graph shows the learning curves for an unshaped network, a shaped network with manual allocation and a shaped network with auto-allocation. Standard errors of the mean are shown as dotted lines for each of the 3 learning curves.

5. General discussion

Although shaping is widespread as a method of training subjects to perform complex behavioral tasks, its effects in computational models have not been extensively investigated. We studied shaping in a moderately complex hierarchical working memory task, showing that it offers significant benefits for learning in the face of medium-term to long-term temporal demands. Speed is not the only benefit of shaping – we showed that it also leads to a solution of the task that generalizes better over time, and is also more flexible in the face of task changes such as reversals.

There is not yet a clear computational theory that provides constraints on appropriate ways of designing shaping protocols. Indeed, the rapid variation in methods of teaching, suggests that there may also be a dearth of clear psychological constraints. It does seem evident that finding the separable hierarchical parts of a problem is key (Watkins, 1989; Singh, 1992; Parr & Russell, 1997; Barto & Mahadevan, 2003), but if there is more than one way of decomposing a task, or indeed more than two levels in its underlying hierarchy, then further experimentation may be necessary. We did refute the possible hypothesis that any way of making a task progressively more complex would be equally valuable, by showing that training simply with successively longer outer loops without the initial hierarchical decomposition did not show the full benefits, such as generalisation. Further, more subtle changes to such aspects of the procedure as the distribution of inner loop lengths during shaping could result in slightly different failure modes of shaping. In the future it will be important to try and characterize such differences and test the resulting predictions experimentally.

Unfortunately, there appear to be no published data on the learning performance of human subjects on the `12-AX` task, let alone on the effect of shaping. It would be interesting to use a model like ours to make behavioral predictions based on different ways of decomposing and shaping a task. One direction to turn for this is transfer learning, an example of which is a recent study by Dahlin, Neely, Larsson, Backman, and Nyberg (2008). Here, subjects that were first trained upon a letter memory task, performed a 3-back working memory task more proficiently. This result is of particular interest because of the suggested involvement of a process of striatally-influenced updating, which is somewhat analogous to our PBWM-based gated working memory. The ability to filter or gate has actually been shown to correlate with performance in working memory tasks (McNab & Klingberg, 2008). Although not a direct model, our simulations may show related effects, in that shaping can enhance a marker of working memory. During symbol generalisation, the shaped network showed stronger reliance on the one back stimulus than the immediate stimulus, an effect linked with increased working memory.

From a neural perspective, there is ample evidence for the involvement of structures in the lateral prefrontal cortex (PFC) in hierarchically structured cognitive tasks including the `12-AX` task (Koechlin & Summerfield, 2007; Reynolds, 2007), and indeed O'Reilly and Frank's (2005) PBWM model is focused on PFC, together with its dopaminergic inputs and connections with the basal ganglia. We are not aware of any data on the effects of different sorts of shaping on the neural activity in, or the Blood-oxygen-level dependent (BOLD) signal measured by functional magnetic resonance imaging (fMRI) from, different PFC areas. In tasks more complex than `12-AX`, it might be possible to use the ideas about processing decompositions from Koechlin et al. (2003) and make predictions about the effects of different methods of shaping that could be tested using imaging.

Our simulations showed that shaping alone without the support of an allocation mechanism, can perform worse than no shaping. This could be because it poses more acute stability–plasticity dilemmas (Grossberg, 1980). Although Section 4 presents a simple algorithm for automatic allocation, it is not a complete solution, and substantial further work will be needed to make this procedure robust and general. In particular, to accommodate asymptotic errors in probabilistic tasks, it will be necessary to incorporate expected as well as unexpected uncertainty (Yu & Dayan, 2005). Reynolds et al. (2007) and Redish et al. (2007) showed that using a rapid change in the running average of the error rate can account for some probabilistic effects. However, this fails to accommodate more sophisticated fluctuations in expected uncertainty, and these authors also see a significant drop in performance of their automatic model compared to the manual one.

The obvious alternative is to have the subject explicitly model the uncertainty. In the wider field of machine learning, a popular way of handling the equivalent of allocation involves mixture models, in which underlying modules compete (Jacobs, Jordan, Nowlan, & Hinton, 1991) to explain inputs based on their own so-called *generative, predictive,* or *forward* models. The MOSAIC framework (Haruno, Wolpert, & Kawato, 2001) for mixture model learning for motor control is a good example of this, and could admit a generalisation to shaping.

Another, related, possibility is to consider resource allocation itself as a recursive instance of gating, now at the level of strategy learning rather than read-in to working memory. In our models we chose a resource allocation that was exclusive and hence resulted in local representation of tasks. This type of allocation lends itself comfortably to the simple and abstract model of LSTM. However, these results will hopefully extend at least qualitatively to the more distributed representations that are likely to be found in natural neural networks. The need for explicit resource allocation is also likely to extend to such networks.

We focused on shaping in operant conditioning. Three additional forms of shaping are also important, and would be interesting targets for future studies. First, in sculpting animal behavior, it has been observed that it is beneficial, or even essential, to start from the actions intrinsically emitted as Pavlovian responses to predictions. Breland and Breland (1961) provide some striking and memorable examples of this maxim, and, as in negative automaintenance (Sheffield, 1965; Williams & Williams, 1969; Dayan, 2006), the deleterious consequences of ignoring it.

The second form of shaping is self-shaping, in which subjects themselves may explicitly simplify tasks, by omitting certain features (Duncan, 1995), either deliberately, or perhaps just by not understanding them. Elman (1993) shows the potential benefits of this in the case of grammar learning. However, explicit self-shaping requires the automated discovery of the hierarchical structure of tasks, which is highly non-trivial itself. Such hierarchical structure discovery has been an active, though not currently strikingly productive, focus of work in machine learning (McGovern & Barto, 2001; Barto & Mahadevan, 2003; Bakker & Schmidhuber, 2004).

Finally, shaping can involve the formation through training of more abstract representations of input that can be used to speed the subsequent learning of complex behaviors. In the field of machine learning, this is the standard view of the interaction between unsupervised learning, which represents inputs according to the underlying structure of their statistical distribution, and supervised learning, which uses these representations to perform tasks well (Hinton, Osindero, & Teh, 2006; Hinton & Ghahramani, 1997). However, it is also possible to generalize the use of representations learnt directly to solve one task to other tasks. For instance, following on from Premack (1983), Thompson, Oden, and Boysen (1997) showed that chimpanzees that had been trained on a sophisticated task of determining the identity of two inputs, thus putatively building a new, abstract, representational unit, could more readily learn a separate task, in which the identity or otherwise of two initial inputs determined what actions should be subsequently executed.

In conclusion, the critical role that shaping plays in the genesis of complex cognitive behavior has not been adequately reflected in computational models. Although our account is incomplete, notably because we have not specified a complete and fully robust method for resource allocation, we have shown the large potential benefits of simple forms of shaping, and provided some insights into the provenance of these advantages.

Acknowledgements

This work was supported by the Gatsby Charitable Foundation. We are grateful to Paul Burgess, Tony Dickinson, John Duncan, Sam Gilbert, Randy O'Reilly, Jeremy Reynolds and Tim Shallice for many helpful comments and for sharing data and ideas prior to publication.

Appendix A. Network rules

This section describes the detailed equations used in simulating the LSTM network. These come from the original paper describing LSTM with forget gates Gers et al., 2000.

The LSTM network consists of five types of units: Input units with unary (0/1) activations i_l based on the stimulus presented; output units o_R, o_L, indicating which choice the network makes; and then each memory block (see Fig. A.1) has three different sorts of gating unit $g^{\mathrm{I}}, g^{\mathrm{F}}, g^{\mathrm{O}}$, and three units for each cell c within a block, input x_c, memory s_c and output y_c. Sigmoid and tanh functions are used as activation functions throughout, based on linearly weighted inputs. The activity of the memory cells at one timestep act as inputs to all the relevant units at the next timestep. We describe the equations for a single memory block below. When the network contains multiple memory blocks, all the units in each block receive inputs from all the other blocks.

Gating units: Gating term $g^{\mathrm{I}}(t)$ is calculated as

$$g^{\mathrm{I}}(t) = \sigma\left(\sum_l m_l^{\mathrm{I}} y_l(t-1) + \sum_l w_l^{\mathrm{I}} i_l(t)\right) \tag{A.1}$$

where $\sigma(z) = 1/(1+\exp(-\beta z))$ is a standard logistic sigmoid function, m_l^l are recurrent weights from the memory cells y_l, and w_l^l are feed–forward weights from the input

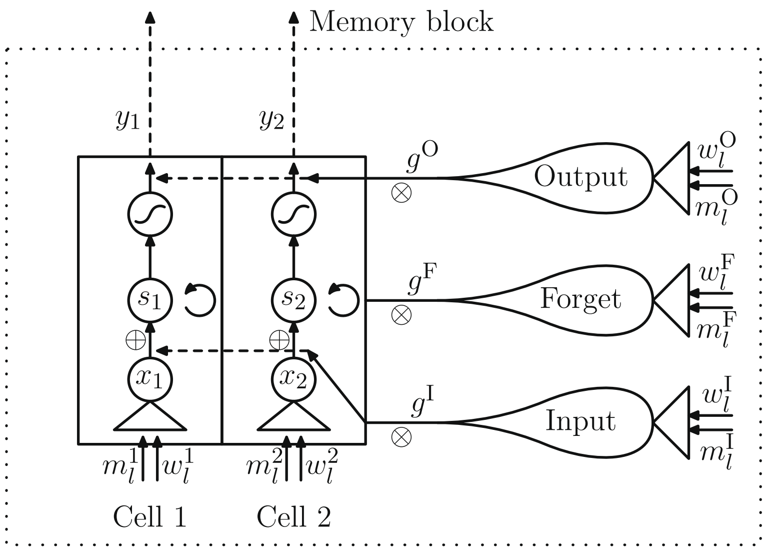

Fig. A.1. A single memory block of the LSTM together with the notation used in this Appendix.

units i_l. $g^F(t)$ and $g^O(t)$ are calculated similarly. β was chosen to be 1.5.

Memory input units: The input to memory cell c is

$$x_c(t) = 2\tanh\left(\sum_l m_l^c y_l(t-1) + \sum_l w_l^c i_l(t)\right) \tag{A.2}$$

Storage unit: The stored content of memory cell c at time t ($s_c(t)$) is calculated as its content at the previous time step combined with the gated input. At the beginning of each epoch, the memory cells' contents are reset to zero $s_c(0) = 0$

$$s_c(t) = s_c(t-1) \times g^F(t) + x_c(t) \times g^I(t) \tag{A.3}$$

Memory output unit: The output depends on another nonlinearity

$$y_c(t) = \tanh(s_c(t)) \times g^O(t) \tag{A.4}$$

Choice units: Finally, the choice of the network is based on the two output units o_L and o_R, which are calculated as

$$o_R(t) = \sigma\left(\sum_l m_l^R y_l(t-1) + \sum_l w_l^R i_l(t)\right) \tag{A.5}$$

Although this form of output is used for learning, in evaluating the performance of the network, we binarise the units output, equivalent to setting $\beta = \infty$. Activations of [1, 0] correspond to a left response, [0, 1] to a right response, and both [0, 0] and [1, 1] are considered invalid responses, and are counted as errors.

Appendix B. Automatic allocation

Automatic allocation is based on detecting a sudden increase in error rate. For this, after each trial a smoothed success rate is calculated that extends over the last 300 trials

$$\text{Suc} = \frac{1}{z} * \sum_{i=0}^{299} c(t-i) * e^{-0.01i} \tag{B.1}$$

where $c(t)$ denotes if the trial at time t was correct and $z = \frac{1-e^{-3.00}}{1-e^{-0.01}}$ is a normalisation constant to constrain the score between 0 and 1. New allocation is only possible once the results of previous allocation have surpassed a success rate threshold of 0.99, and there is then a sudden increase in error rate above 10%.

An allocation step consists of enabling a new memory block with randomly initialised weights. The learning rate of this new block is set to $\alpha = 0.1$ and the learning rate of all previously allocated blocks is reduced to $0.05 * \alpha$ to prevent forgetting of previously learned tasks.

References

Badre, D., Poldrack, R. A., Pare-Blagoev, E., Juliana amd Insler, R. Z., & Wagner, A. D. (2005). Dissociable controlled retrieval and generalized selection mechanisms in ventrolateral prefrontal cortex. *Neuron, 47*(6), 907–918.

Bakker, B., & Schmidhuber, J. (2004). Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In F. Groen, N. Amato, A. Bonarini, E. Yoshida, & B. Kroese, (Eds.), *Proceedings of the 8th conference on intelligent autonomous systems*, Vol. IAS-8, pp. 438–445.

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications, 13*(4), 341–379.

Breland, K., & Breland, M. (1961). The misbehavior of organisms. *American Psychologist, 16*(11), 681–684.

Brown, S. L., Joseph, J., & Stopfer, M. (2005). Encoding a temporally structured stimulus with a temporally structured neural representation. *Nature Neuroscience, 8*(11), 1568–1576.

Butter, C. M. (1969). Perseveration in extinction and in discrimination reversal tasks following selective frontal ablations in Macaca mulatta. *Physiology & Behaviour, 4*(2), 163–171.

Dahlin, E., Neely, A. S., Larsson, A., Backman, L., & Nyberg, L. (2008). Transfer of learning after updating training mediated by the striatum. *Science, 320*(5882), 1510–1512.

Dayan, P. (2006). Images, frames, and connectionist hierarchies. *Neural Computation, 18*(10), 2293–2319.

Dayan, P., & Yu, A. J. (2006). Phasic norepinephrine: A neural interrupt signal for unexpected events. *Network: Computation in Neural Systems, 17*(4), 335–350.

Dorigo, M., & Colombetti, M. (1998). *Robot shaping: An experiment in behavior engineering*. MIT Press/Bradford Books.

Duncan, J. (1995). Attention, intelligence, and the frontal lobes. In M. S. Gazzaniga (Ed.), *The new cognitive neurosciences* (pp. 721–733). The MIT Press.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition, 48*(1), 71–99.

Frank, M. J., Loughry, B., & O'Reilly, R. C. (2001). Interactions between frontal cortex and basal ganglia in working memory: A computational model. Technical Report, Department of Psychology, University of Colorado.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation, 12*(10), 2451–2471.

Gilbert, S. J., Frith, C. D., & Burgess, P. W. (2005). Involvement of rostral prefrontal cortex in selection between stimulus-oriented and stimulus-independent thought. *European Journal of Neuroscience, 21*(5), 1423–1431.

Grossberg, S. (1980). How does a brain build a cognitive code? *Psychological Review, 87*(1), 1–51.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Computation, 13*(10), 2201–2220.

Hazy, T. E., Frank, M. J., & O'Reilly, R. C. (2007). Towards an executive without a homunculus: Computational models of the prefrontal cortex/basal ganglia system. *Philosophical Transactions of the Royal Society B. Biological Sciences, 362*(1485), 1601–1613.

Hinton, G. E., & Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society B, 352*(1358), 1177–1190.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Iversen, S. D., & Mishkin, M. (1970). Perseverative interference in monkeys following selective lesions of the inferior prefrontal convexity. *Experimental Brain Research, 11*(4), 376–386.

Jacobs, R., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*(1), 79–87.

Jones, B., & Mishkin, M. (1972). Limbic lesions and the problem of stimulus–reinforcement associations. *Experimental Neurology, 36*(2), 362–377.

Koechlin, E., Ody, C., & Kouneiher, F. (2003). The architecture of cognitive control in the human prefrontal cortex. *Science, 302*(5648), 1181–1185.

Koechlin, E., & Summerfield, C. (2007). An information theoretical approach to prefrontal executive function. *Trends in Cognitive Sciences, 11*(6), 229–235.

McGovern, A., & Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th international conference on machine learning* (pp. 361–368). San Francisco, CA: Morgan Kaufmann.

McNab, F., & Klingberg, T. (2008). Prefrontal cortex and basal ganglia control access to working memory. *Nature Neuroscience, 11*(1), 103–107.

Newport, E. L. (1988). Constraints on learning and their role in language acquisition: Studies of the acquisition of american sign language. *Language Sciences, 10*(1), 147–172.

Newport, E. L. (1990). Maturational constraints on language learning. *Cognitive Science, 14*(1), 11–28.

O'Reilly, R. C., & Frank, M. J. (2005). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation, 18*(2), 283–328.

Parr, R., & Russell, S. (1997). Reinforcement learning with hierarchies of machines. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems*. The MIT Press.

Peterson, G. B. (2004). A day of great illumination: B.F. Skinner's discovery of shaping. *Journal of the Experimental Analysis of Behavior, 82*(3), 317–328.

Premack, D. (1983). The codes of man and beasts. *Behavioral and Brain Sciences, 6*, 125–167.

Redish, A. D., Jensen, S., Johnson, A., & Kurth-Nelson, Z. (2007). Reconciling reinforcement learning models with behavioral extinction and renewal: Implications for addiction, relapse, and problem gambling. *Psychological Review, 114*(3), 784–805.

Reynolds, J. (2007). Computational, behavioral, neuro-imaging methods investigating the hierarchical organization of pfc and goal-oriented behavior. unpublished/NIPS workshop on hierarchical organization of behavior.

Reynolds, J. R., Zacks, J. M., & Braver, T. S. (2007). A computational model of event segmentation from perceptual prediction. *Cognitive Science, 31*(4), 613–643.

Rohde, D. L. T., & Plaut, D. C. (1999). Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition, 72*(1), 67–109.

Saksida, L. M., Raymond, S. M., & Touretzky, D. S. (1997). Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems, 22*(3), 231–249.

Savage, T. (1998). Shaping: The link between rats and robots. *Connection Science, 10*(3), 321–340.

Savage, T. (2001). Shaping: A multiple contingencies analysis and its relevance to behaviour-based robotics. *Connection Science, 13*(3), 199–234.

Shallice, T., & Burgess, P. W. (1991). Deficits in strategy application following frontal lobe damage in man. *Brain, 114*(2), 727–741.

Sheffield, F. (1965). Relation between classical conditioning and instrumental learning. In W. F. Prokasy, (Ed.), *Classical conditioning: A symposium. Appleton-Century-Crofts*, 302–322.

Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning, 8*, 323–339.

Skinner, B. F. (1938). The behavior of organisms: An experimental analysis. *Appleton-Century-Crofts*.

Sowell, E. R., Thompson, P. M., Holmes, C., Jernigan, T. L., & Toga, A. W. (1999). In vivo evidence for post-adolescent brain maturation in frontal and striatal regions. *Nature Neuroscience, 2*, 859–861.

Thompson, R., Oden, D., & Boysen, S. (1997). Language-naive chimpanzees (Pan troglodytes) judge relations between relations in a conceptual matching-to-sample task. *Journal of Experimental Psychology: Animal Behavior Processes, 23*(1), 31–43.

Watkins, C. J. C. H. (1989). Learning from delayed rewards. Ph.D. Thesis, Cambridge, UK.

Williams, D. R., & Williams, H. (1969). Auto-maintenance in the pigeon: Sustained pecking despite contingent non-reinforcement. *Journal of the Experimental Analysis of Behavior, 12*(4), 511–520.

Yu, A. J., & Dayan, P. (2005). Uncertainty, neuromodulation, and attention. *Neuron, 46*, 681–692.

Zacks, J. M., Speer, N. K., Swallow, K. M., Braver, T. S., & Reynolds, J. R. (2007). Event perception: A mind–brain perspective. *Psychological Bulletin, 133*(2), 273–293.