Distal Dendrite Feedback in Hierarchical Temporal Memory

Adam Kneller and John Thornton

Institute for Integrated and Intelligent Systems
School of ICT, Griffith University, Queensland, Australia
Email: adam.kneller@griffithuni.edu.au
j.thornton@griffith.edu.au

***Abstract*—Recent theories have proposed that the unifying principle of brain function is the minimisation of variational free energy and that this is best achieved using a hierarchical predictive coding (HPC) framework. Hierarchical Temporal Memory (HTM) is a model of neocortical function that fits within the free energy framework but does not implement predictive coding. Recent work has attempted to integrate predictive coding and hierarchical message passing into the existing suite of HTM Cortical Learning Algorithms (CLA) producing a PC-CLA hybrid. In this paper we examine for the first time how such hierarchical message passing can be implemented in a *pure* HTM framework using distal dendrite structures that are already implemented in the CLA temporal pooler. We show this approach outperforms the more simplistic proximal dendrite structures used in the PC-CLA hybrid and also that the new CLA hierarchy is effective for anomaly detection and image reconstruction problems that are beyond the reach of the existing single-level CLA framework.**

I. Introduction

Recent developments in computational neuroscience suggest that the primary function of the neocortex is the formation of predictions concerning future input. This is underpinned by Friston's free-energy principle [1] which asserts that the brain learns to predict sensory input by minimising the variational free energy of its internal states – thereby minimising 'surprising' interactions with the world. According to the Bayesian brain hypothesis [2], the brain achieves this by constructing a generative probabilistic model of the causes of sensory input which it optimises by testing the model against sensory input and updating beliefs following Bayesian principles.

Friston (and others) currently propose hierarchical predictive coding (HPC) as the most promising means for the brain to implement the free-energy principle [3]. Under HPC, higher-level regions of the brain convey predictions or expectations of future activity to lower-level regions via top-down (feedback) connections. If predictions conflict with lower-level activity the difference is returned as *prediction error* [4]. Higher regions then resolve prediction error by altering their models of the causes that generated the error.

Hierarchical Temporal Memory (HTM) is an alternative model of neocortical functioning that also fits within the free-energy framework. In contrast to HPC, HTM is based on sequence learning rather than predictive coding. This is implemented by means of hierarchically structured cortical regions learning their common spatial input patterns and (as in HPC) using this information to generate downward projecting predictions concerning the future activity of lower-level regions. However, in HTM this is not in response to prediction error but is rather a 'confirmation' or 'reinforcement' of bottom-up activity.

The broad outline of the HTM model was first proposed by Jeff Hawkins in 2004 [5]. Subsequently several HTM components, known as Cortical Learning Algorithms (CLA), have been developed by Hawkins and his colleagues at Numenta Inc. [6]. The current generation CLA comprise two main components: the *spatial pooler* (SP), and the *temporal pooler* (TP). Further research has extended the Numenta SP to handle real-valued [7] and multi-channel input [8]. The TP has also been compared to hidden Markov models for noisy sequence learning [9] where it has shown promising results.

Until recently, the CLA components have not been combined into a hierarchy. The first such attempt [10] involved integrating the SP and TP into a hierarchical predictive coding framework producing a Predictive Coding-CLA hybrid (PC-CLA). Here the sequence learning of HTM is retained, but top-down and bottom-up message passing now occurs between hierarchical levels, conveying predictions and prediction error respectively. Although the benefit of passing bottom-up prediction error (as opposed to total activity) in HTM has yet to be established, the authors demonstrate that hierarchical message passing improves performance over a single-level CLA [10], [11].

In this paper, we investigate (for the first time) the construction of a *pure* HTM hierarchy that follows the original intent of the HTM model. To this end we experiment with two possible hierarchical message-passing strategies, one (following [10]) that uses bi-directional proximal dendrites (adapted from the CLA SP) and the other using distal dendrite structures (adapted from the CLA TP). Our results show the distal dendrite strategy is more effective than the PC-CLA approach. We also investigate the usefulness of the new CLA hierarchy by showing that it is effective for both anomaly detection and image reconstruction problems that lie beyond the reach of the existing single-level CLA framework.

II. Hierarchical Temporal Memory

The HTM model has three primary structural components: regions, columns, and cells. *Regions* are self-contained blocks that constitute the main components of the hierarchy, with regions that are higher up the hierarchy receiving input from several regions in the level below (Fig. 1). Each region comprises a number of *columns* arranged in a two-dimensional grid with each column containing a number of *cells* (Fig. 2). The column is the main functional unit of the HTM with the

978-1-4799-1959-8/15/$31.00

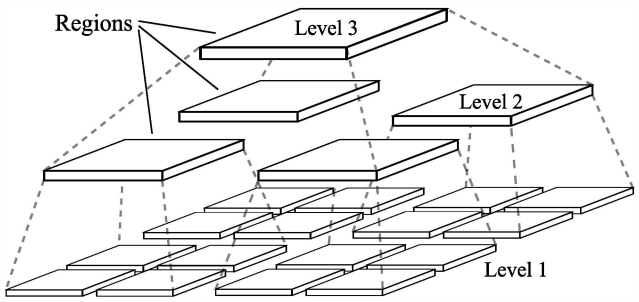

Fig. 1. A three-level HTM hierarchy.

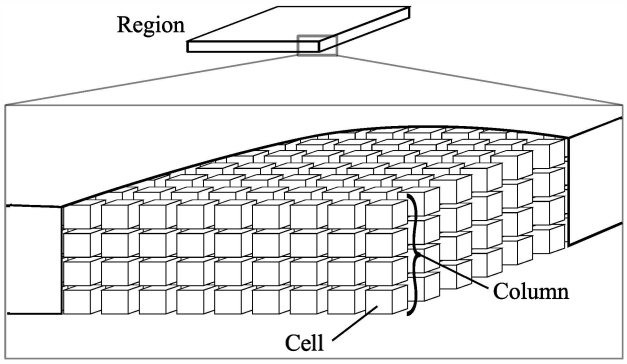

Fig. 2. The internal structure of a region. Each column in this region contains four cells.

spatial pooler (SP) operating on the column as a whole unit and the *temporal pooler* (TP) operating on its constituent cells.

The connection between a column or cell and its input is called a *synapse*, each of which has a real-valued *permanence*. The permanence differs from a weight in a traditional neural network in that it does not multiplicatively alter the input. Instead it is compared to a *connection threshold* which, if exceeded, causes the synapse to become a *connected synapse* that passes its input on unaltered; otherwise the synapse is a *potential synapse* and the input is ignored. In this sense HTM synapses may be considered to have binary weights.

The CLAs combine to form an online system that receives a continuous stream of input. Each level completes running the SP and TP for all regions before the algorithm moves to the next higher level. Training of the HTM can also be conducted offline using a fixed-size data set, as shown in Fig. 3. Here, online operation would correspond to a continuous repetition of Lines 13–15 for all levels from 1 to L.

We now provide more detail on the functioning of the SP and TP with an emphasis on the TP, as our feedback mechanism draws heavily from the concepts embodied in it. Note also that although the CLAs are designed to handle any modality of data, our description will focus on processing images. A summary of HTM structures is provided in Table I.

A. Spatial Pooler

The SP is a sparse feature detector that learns the common spatial patterns in a region's input and, at every time step, produces a sparse distributed representation (SDR) [12] of the current input. It has been extended from the original algorithm [6] into the Augmented Spatial Pooler (ASP) [7]

```
1:  procedure OFFLINETRAINING(htm, data[1..T], C)
2:      levels[1..L] ← htm.levels
3:      for ℓ ← 1 to L do
4:          regions ← levels[ℓ].regions
5:          for all r in regions do
6:              repeat
7:                  for t ← 1 to T do
8:                      spatialPooler(r, data[t])
9:                  end for
10:             until SP has converged for r
11:             for c ← 1 to C do
12:                 for t ← 1 to T do
13:                     spatialPooler(r, data[t])
14:                     SDR ← activity of all columns
15:                     temporalPooler(r, SDR)
16:                 end for
17:             end for
18:         end for
19:     end for
20: end procedure
```

Fig. 3. Offline training of the CLAs. C is the number of iterations of the data to run through the TP. Training can be sped-up by caching the output of the SP (when training the TP), or the level below (when $\ell > 1$), for all data instances so that they do not need to be re-computed.

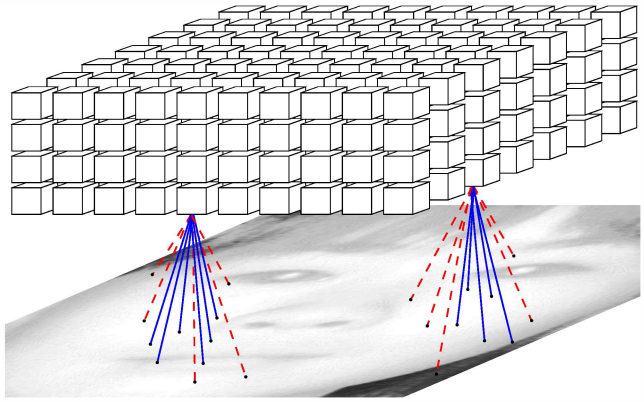

Fig. 4. Columns' synapses with their input. Solid lines indicate *connected synapses*, while dashed lines represent *potential synapses*.

that is capable of handling non-binary input, multiple channels, and is designed to converge onto a stable set of codes without learning being artificially terminated.

During initialisation, each column possesses a *proximal dendrite* which forms *proximal synapses* with a random subset of the region's input. These synapses represent the subset of the input space to which the column has access (Fig. 4). Proximal synapse permanence values are randomly initialised, but are related to the topographic distance of the input pixel from the column and can be determined by several methods (e.g. linearly or via a Gaussian [13]).

HTM columns are deterministic binary units whose feed-forward (bottom-up) state variable is called $stateFF$. The SP begins with all columns summing the inputs from their connected synapses. This sum is then used in a lateral inhibitory competition where the more active columns (higher summed input) within a predefined radius inhibit the less active ones. User defined parameters determine

TABLE I. SUMMARY OF HTM STRUCTURES

Existing HTM structures	
Regions	Main component of the hierarchical structure. Composed of a 2-D grid of columns. Regions at the same level in the hierarchy do not interact.
Columns	Primary functional unit in a region, contains a number of cells and a proximal dendrite. The SP acts on columns forming SDRs of active columns. Has binary state variables for bottom-up ($stateFF$) and top-down ($stateFB$) activation.
Proximal Synapses	Attached to a column's proximal dendrite and form its feedforward connection to input. Each has a real-valued *permanence* between 1 and 0. May be *connected* with binary weight 1, if permanence exceeds a *connection threshold*, otherwise will be *potential* with binary weight 0.
Cells	The TP acts on a column's cells in order to encode the column's activity in different temporal contexts. Has binary state variable $activeState$, ternary state variable $predictiveState$ (with values *off*, $next$, $future$), and also contains a number of distal dendrite segments.
Distal Dendrite Segments	Attached to cells and used by the TP for determining a cell's $predictiveState$. Each cell will have many distal dendrite segments. If the number of *connected* synapses on the segment (to other active cells) exceeds an *activation threshold*, then the segment becomes an *active segment*, causing its cell to enter a $predictiveState$.
Distal Synapses	Attached to distal dendrite segments and form lateral connections to other cells in the *same* region.
Proposed additional structures for top-down feedback	
Feedback Dendrite Segments	Attached to columns and used to determine top-down feedback. As with distal dendrite segments, if the number of active synapses exceeds the activation threshold then the segment becomes active.
Feedback Synapses	Attached to feedback dendrite segments, form connections to columns in the region at the next *higher* level in the hierarchy.

what percentage of columns within the radius remain active. Columns that win have $stateFF \leftarrow on$ and all others have $stateFF \leftarrow$ *off*. The pattern of active and inactive columns forms the SDR that is passed to the TP (i.e. $SDR = [stateFF_1, stateFF_2, \ldots, stateFF_N]$).

Learning in the SP is a Hebbian-like process where columns increment the permanence of their proximal synapses to non-zero inputs, and decrement the permanences of all other synapses. At any time step only the active columns perform this learning, inactive columns do not alter their synapses (for further details of the SP and ASP including the parameters controlling activation and learning, see [6]–[8], [13]).

B. Temporal Pooler

The TP aims to discover common sequences of region activity by having each column learn to predict when it will become active, based on the region's activity in preceding time steps. The SDR from the SP represents the sparse features that are currently present in the region's input and this is placed in a temporal context by the TP selecting a combination of cells within active columns to also become active. This allows the TP to distinguish the *same* pattern of active columns in *different* contexts, i.e. a set of n active columns each containing m cells can represent the same pattern in n^m different contexts. Active cells then remember the pattern of activity in the previous time step, thus allowing them to predict their own activity and enabling the TP to learn sequences.

Each cell has one binary state variable, $activeState$, and one ternary state variable, $predictiveState$, with values *off*, $next$, and $future$. Cells also have a number of *distal dendrite segments,* and each such segment has a number of *distal synapses*. As with SP proximal synapses each distal synapse has a *permanence* value that is compared to a *connection threshold* to determine whether or not it is connected.

Cells use their distal segments as a means of remembering active-cell patterns. Whenever a cell enters an active state at time t it chooses one of its segments and new synapses are added to the segment; these synapses connect to cells in other columns that were in an active state at time $t-1$. In future time steps a cell will compare the current pattern of active cells to all of the patterns remembered by its segments. If any segment closely matches the current active-cell pattern, as determined by an *activation threshold*, then this indicates that the cell expects to become active in the near future.

The TP has a three phase learning algorithm: calculating a cell's active state, predictive state, and learning.

1) Phase I – Calculating Active State: The TP receives an SDR from the SP, specifying $stateFF$ for all columns. For every active column, if any cell i in the column had $predictiveState_i(t-1) = next$ then that cell now enters an active state ($activeState_i(t) \leftarrow 1$). If no cells in an active column were in a predictive state at the end of the previous time step then all cells now enter an active state, i.e.:

$$\forall\, i \in column.cells,\ activeState_i(t) \leftarrow 1$$

2) Phase II – Calculating Predictive State: At the end of Phase I some columns will have one or more active cells and a few may have all their cells active. Now the TP determines its belief about which cells will become active in future time steps: For every distal dendrite segment on every cell in every column (active or inactive), determine the number of *active synapses*. An active synapse is a connected synapse that connects to a cell with $activeState(t) = 1$. If the number of active synapses on a segment exceeds an *activation threshold* then it is considered an *active segment* and the cell enters a predictive state ($predictiveState(t) \leftarrow \{next, future\}$). Whether $predictiveState$ gets $next$ or $future$ depends upon the particular active segment, which will be discussed in Phase III. At the end of Phase II all cells in the region will have a value for $activeState$ and $predictiveState$, the majority of which will be inactive due to the sparse nature of the algorithm.

3) Phase III – Learning: In this phase all cells may potentially update their distal dendrite segments and synapses based upon the accuracy of their predictions. When a cell predicts that it will become active ($predictiveState \neq$ *off*) but remains inactive for a certain number of time steps then all distal

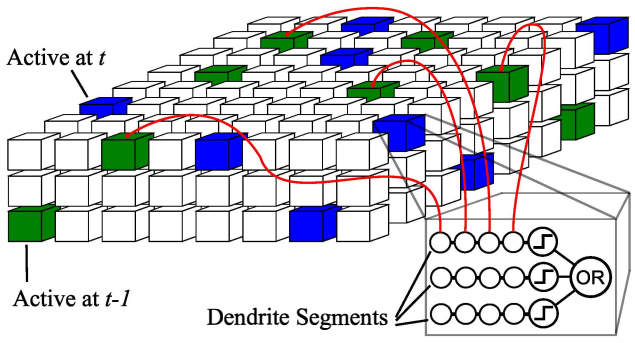

Fig. 5. A cell growing a new dendrite segment. The blue cells are active at time t, and the green cells were active at time $t-1$. The active cell in the lower right corner grows new synapses to cells that were active at $t-1$.

synapses contributing to that prediction (i.e. active synapses on an active segment) have their permanences decremented by a fixed amount. Where a prediction turns out to be correct the synapse permanences are incremented by a fixed amount (possibly different from the decrement).

Where a column became active unexpectedly (i.e. all of its cells became active in phase I) then one cell in the column is chosen randomly to grow a new distal dendrite segment. This segment forms new synapses to cells that were active in the previous time step (Fig. 5). This type of segment is known as a *sequence segment* and when it becomes active it causes the cell to enter $predictiveState(t) \leftarrow next$.

Cells learn to predict their activity further and further into the future by queueing *possible* new segments and synapses that are then added if and when a cell's predictions are proved correct. When a cell enters a predictive state (either $next$ or $future$) it queues a possible new *non-sequence* segment that (when active) will cause its cell to enter $predictiveState(t) \leftarrow future$. If a cell enters an active state at some future time step, all of its queued segments are added. If the cell ever drops out of a predictive state without becoming active ($predictiveState(t) = off$ AND $activeState(t) = 0$) then its queued segments are deleted.

As with the SP, the TP has various parameters that control, for example, the magnitude of the permanence increments and decrements, the number of synapses to add to a new distal dendrite segment, the connection threshold, the activation threshold, etc. For full details see [6].

III. Feedback

This section presents the main contribution of the paper: the *Distal Feedback* method for implementing feedback in HTM. The purpose of top-down message passing in hierarchical predictive coding (HPC) is to convey higher level predictions about activity at lower levels [4]. In temporal contexts this would be a prediction of activity in the next time step [14]. However, this is of limited benefit in HTM as sequence learning and prediction are *already* occurring within a region without the need for feedback from above (through the activity of the TP). Thus our feedback is primarily designed to convey higher level synthesised information about the activity of neighbouring regions at a lower level, thereby allowing a lower level region to resolve ambiguities in its predictions given contextual information concerning other regions.

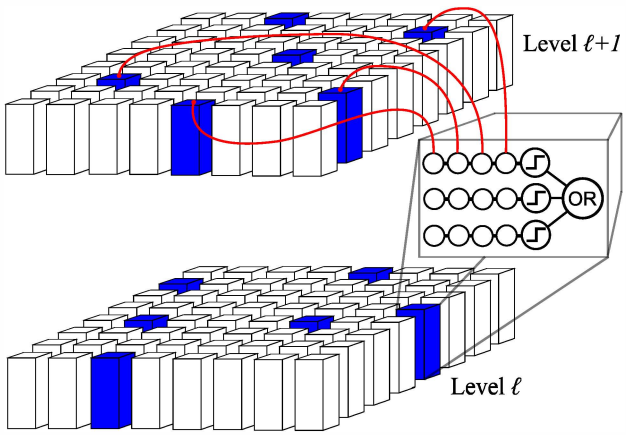

Fig. 6. Columns at level ℓ grow feedback synapses, in the current time step, to active columns in the region above (level $\ell+1$).

A. Feedback Implementation

The feedback technique we are proposing is based on the TP distal dendrite structure. Using this structure, we create an additional set of distal (feedback) dendrites that receive descending prediction signals from a higher region (rather than laterally from within the same region). The key implementation changes are that feedback segments are attached to columns (instead of cells), and that synapses are formed to active columns in the region above in the *current* time step (see Fig. 6). The pseudocode for this DISTALFEEDBACK method is shown in Fig. 8, its role in the overall functioning of the hierarchy is shown in Fig. 7 and is explained further below:

The calculation of feedback occurs after a higher level region has completed running the TP. First every column determines a new feedforward state $stateFF$ by taking the boolean OR of its cells' $activeState$ and $predictiveState$:

$$stateFF = \begin{cases} on & \text{if } \exists\ i \in cells \text{ s.t. } (activeState_i = 1 \\ & \text{OR } predictiveState_i \neq off) \\ off & \text{otherwise} \end{cases}$$

overriding the $stateFF$ calculated at the end of the SP. Once all columns have redetermined their $stateFF$, feedback determines the value of a new binary state variable $stateFB$.

Given a region and the activity at the higher level DISTALFEEDBACK loops through every column and determines its expectation about top-down activity. First FINDBESTSEGMENT (Line 3) returns the feedback dendrite segment[1] with the greatest number of active synapses, provided it exceeds the activation threshold. It may return NULL indicating that feedback does not predict that the column should be active (Line 4). From here there are four scenarios that may occur (Lines 6–12):

1) In the simple case where FINDBESTSEGMENT returns NULL and $stateFF = off$ no action is taken.
2) FINDBESTSEGMENT returns a non-NULL segment and the column's $stateFF = on$. This indicates that the top-down expectation is in agreement with bottom-up activation and therefore the segment is reinforced, i.e. has its synapse permanences incremented (Line 7). Only

[1]To distinguish distal dendrite feedback segments from TP segments we call them *feedback dendrite segments* and their synapses *feedback synapses*.

```
1: procedure OFFLINETRAINING(htm, data[1..T], C, F)
2:     levels[1..L] ← htm.levels
3:     for ℓ ← 1 to L do
4:         regions ← levels[ℓ].regions
5:         for all r in regions do
6:             repeat
7:                 for t ← 1 to T do
8:                     spatialPooler(r, data[t])
9:                 end for
10:            until SP has converged for r
11:            for c ← 1 to C do
12:                for t ← 1 to T do
13:                    spatialPooler(r, data[t])
14:                    SDR ← activity of all columns
15:                    temporalPooler(r, SDR)
16:                end for
17:            end for
18:        end for
19:        if ℓ > 1 then
20:            for all r in levels[ℓ − 1].regions do
21:                for f ← 1 to F do
22:                    for t ← 1 to T do
23:                        distalFeedback(r, levels[ℓ])
24:                    end for
25:                end for
26:            end for
27:        end if
28:    end for
29: end procedure
```

Fig. 7. Modified offline training of the CLAs. Where F determines the number of feedback learning iterations taken through the data. Lines 1–18 are identical to those in Fig. 3 while lines 19–27 implement feedback training.

```
1: procedure DISTALFEEDBACK(region, level)
2:     for all column in region.columns do
3:         best ← findBestSegment(column, level)
4:         if best ≠ NULL then
5:             column.stateFB ← on
6:             if column.stateFF = on then
7:                 adaptSegment(best, positive)
8:             else
9:                 adaptSegment(best, negative)
10:            end if
11:        else if column.stateFF = on then
12:            addNewFeedbackSegment(column)
13:        end if
14:    end for
15: end procedure
```

Fig. 8. Pseudocode for feedback calculation.

active synapses have their permanences' increased. The segment may also add a new synapse: if it has not reached the maximum allowable number of synapses, and if there are active higher level columns to which the segment does not already connect.

3) FINDBESTSEGMENT returns a non-NULL segment but the column's $stateFF = off$. Instances where bottom-up and top-down activity do not match may be due to incorrect predictions, noisy bottom-up activation, or incomplete learning. In this case the feedback segment has its active synapse permanences decremented (Line 9). The rationale is that if the feedback segment encodes useful information then it will match feedforward activity more often than not, and therefore decrementing the synapses will not be detrimental in the long run.
4) FINDBESTSEGMENT returns NULL but the column's $stateFF = on$. We assume that this scenario is due to incomplete learning, and so the column adds a new feedback dendrite segment (Line 12). This will be the majority case in the first training cycle, as columns begin with no feedback segments and must learn them all initially. There are parameters defining the number of feedback synapses that new segments begin with, the maximum number of synapses a segment may have, and their initial permanence values. The accuracy and generalisability of feedback may be adjusted through these parameters.

Currently feedback has no effect on learning or inference. This will be a focus for future work.

B. Feedback in PC-CLA

In contrast to our method, PC-CLA uses the same proximal synaptic connections for top-down feedback as it does for bottom-up SP activation, effectively treating them as bi-directional. For this reason we term PC-CLA approach as *Proximal Feedback*. Proximal Feedback is calculated by determining the columns in the higher level region that are predicted to be active in the next time step and then looping through these columns' proximal synapses, ensuring that every connected synapse sends a signal down to the connected column in the lower level indicating that it is expected to become active in the next time step [15]. This can lead to many false positives, such as when a column in a higher region becomes active and does not 'use' all of its proximal synapses; in this case sending a signal back down all synapses will lead to over-prediction. It is also known from neuroanatomy that the cortical connections carrying bottom-up and top-down information are separate and originate in different layers of the neocortex [16], [17]. It therefore seems unlikely that the basic principles of cortical feedback would be captured in a bi-directional model.

IV. EXPERIMENTS

A. Dataset and Pre-processing

In order to evaluate Distal Feedback we performed three experiments using the Hong Kong PolyU NIR Face Database [18] following a similar methodology to that used by Li *et al.* for anomaly detection [19]. The NIR Face Database comprises approximately 100 near-infrared 768×576 pixel images from each of 350 subjects. These images were pre-processed by histogram equalisation, detecting the person's face, straightening, centring and cropping, and then rescaling the cropped face to 32×32 pixels. Following preprocessing we randomly selected 100 images from each of 55 randomly selected subjects. This dataset was then used as input to a two-level HTM with four regions at level 1, and one region at level 2, with each region comprising $1,024$ columns. All images were split into four separate sets of 16×16 pixel images, each of which was associated with one of the regions at level 1 (Fig. 9). The dataset was further split into training and testing sets with the training set comprising 80 images per person ($4,400$ total). The testing set was constructed as

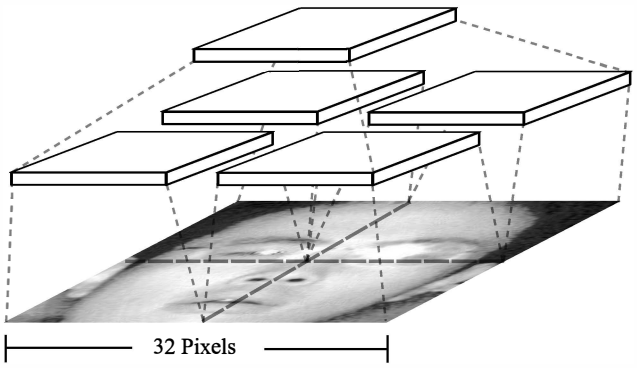

Fig. 9. 32×32 pixel images are split into 4 separate 16×16 images: one for each of the regions at level 1.

Fig. 10. Example anomalous face constructions: the original face is on the top row, the face from which the anomalous quadrant is taken is on the middle row and the final face with anomalous quadrant is on the bottom row.

follows: for each person anomalous instances were created by selecting 40 images from that person's training set, and replacing one of the four quadrants in the image (i.e. top-left, top-right, bottom-left, bottom-right)[2] with the corresponding quadrant randomly selected from a training image of one of the other 54 persons (some examples are shown in Fig. 10). The remaining 20 images per person represent the non-anomalous instances, giving a total test set of $3,300$ images.

B. Comparison to Proximal Feedback

First we compared Distal Feedback and Proximal Feedback's ability to accurately predict lower level activity. Here we implemented PC-CLA's proximal feedback mechanism *within* an HTM framework as our intention was to evaluate the feedback mechanisms on equal terms (leaving the question of the efficacy of predictive coding versus a pure HTM implementation for future work). We performed the prediction accuracy test using the training set and HTM setup described above, where each column contains four cells, each of which may have a maximum of 256 distal dendrite segments. In addition, each distal segment was given 20 synapses initially, with a maximum set at 25, and an activation threshold of 12. The system was trained offline using the procedure from Fig. 7 with $C = 9$ and $F = 6$[3].

Testing was performed by disabling learning and representing the training set. For each instance we used both Distal and Proximal Feedback to predict which columns would become active in the next time step. At the next time step this prediction is compared to actual active columns. We used the F_1 score[4] to compare methods: columns that are bottom-up active but have no top-down prediction are considered false negatives, and columns with top-down prediction but no bottom-up activation are false positives.

Distal Feedback achieved an F_1 score of 0.729 ± 0.004 compared to Proximal Feedback's score of 0.548 ± 0.005. From these results we can see that Distal Feedback is considerably more accurate than Proximal Feedback and we therefore did not include Proximal Feedback in any further testing.

C. Anomaly Detection

The anomaly detection task we present here is based on that presented in [19] but modified to accommodate the partition of the data into four subsets. As anomalies are only being detected in static images (rather than image sequences) temporal predictions are not relevant and the TP was disabled for the entire test. We again used offline training, this time testing varying settings for the number of feedback training cycles. The anomaly score for each region was calculated as $1 - F_1$ score[5]. As described above, the anomalous instances were composed of a person's face, where one of the quadrants is replaced with part of another person's face. The goal of top-down feedback is to provide each region with contextual information as to what is happening in other regions at the same level. Each region will learn the approximate patterns of higher level activity that correspond with different bottom-up activation. When these do not match it is an indication that something is wrong, hence we use the percentage of unpredicted columns as the anomaly score.

The experiment was set up such that each column in the HTM had a maximum of $1,536$ feedback dendrite segments. New feedback segments were initialised with 18 synapses, with each synapse having an initial permanence of 0.25, and segments having a maximum of 22 synapses. The synapse permanence increment was 0.01, and the decrement was 0.02. We then tested different settings for the activation threshold.

D. Face Reconstruction

As a further demonstration of the effectiveness of Distal Feedback we reconstructed one of the faces from the dataset using top-down activation. We used a 320×320 pixel image with no pre-processing (Fig. 12) from which we randomly sampled $10,000$ patches and implemented the same two-level HTM structure used in the anomaly detection experiment (four regions at level 1) with each image patch divided into four regions (as in Fig. 9). Reconstruction was performed as in [7], by taking the linear superposition of connected proximal synapses from active columns. The only difference here is that active columns are determined by feedback state instead of feedforward state (i.e. columns for which $stateFB = on$ instead of columns where $stateFF = on$). We also present reconstructions for corrupted images (one with a black square in the middle, the other with black lines through it, Fig. 13). In this case the HTM was first trained on the original image and

[2]We positioned the anomalies to exactly match one region's input in order to determine Distal Feedback's maximum effectiveness.

[3]All final parameter settings were chosen because they produced promising results during preliminary testing.

[4]The F_1 score is the harmonic mean of the *precision* and *recall*. Precision is the percentage of instances identified as positive that were actually positive and recall is the percentage of positive instances that were identified as positive.

[5]We compute results using the `perfcurve` function from the MATLAB Statistics Toolbox, which requires a higher score for instances that are more likely to be anomalous.

TABLE II. AUC ANOMALOUS QUADRANT RESULTS

	Feedback Training Cycles					
Activation Threshold	1	3	5	7	9	11
5	0.8340	0.8813	0.8946	0.9006	0.9031	0.9050
6	0.8684	0.9009	0.9088	0.9118	0.9135	0.9144
7	**0.8907**	**0.9072**	**0.9123**	**0.9135**	**0.9143**	**0.9148**
8	0.8886	0.9021	0.9059	0.9050	0.9045	0.9045
9	0.8587	0.8832	0.8896	0.8871	0.8858	0.8853

TABLE III. AUC ANOMALOUS FACE RESULTS (FROM LI *et al.* [19])

Method	RNN	One-class SVM	Manifold Clustering	Li *et al.*
AUC	0.7530	0.6893	0.7066	0.7898

reconstruction was performed using patches from the corrupted image, with both feedforward and feedback activation tested. Using feedback activation on the corrupted images allows us to visualise what the system 'believes' it should be seeing based on what it has been presented in the past (i.e. the uncorrupted image) and the activity of other regions at the same level.

V. RESULTS AND DISCUSSION

Table II presents the anomalous quadrant detection results as the area under the ROC curve (AUC) for different settings of the activation threshold and number of feedback training cycles.[6] Here the best result of 0.9148 was achieved using an activation threshold of 7 and 11 feedback training cycles. This result demonstrates that Distal Feedback *works* and also that it works relatively *well*. To provide context for this claim, Table III shows state-of-the-art results from Li *et al.* [19] on a related anomaly detection experiment using the same image database and sampling method. In this case the task was to discriminate, for each of the 55 subjects, between full-face images of a chosen subject and anomalous images of other subjects. Here the best AUC probability was 0.7898 for Li *et al.'s* specialised SVM algorithm. While we cannot compare directly with this result, the related nature of the experiments suggests that reaching over 90% accuracy in discriminating anomalous quadrants is at least a non-trivial achievement.

Fig. 11 further graphs a selection of the results from Table II and illustrates a general trend of increasing performance for lower valued activation thresholds (from 5 to 7) as the number of training cycles is increased. This is to be expected as more training cycles allow feedback segments to refine the patterns they learn by growing and pruning synapses. However, we can also see the beginnings of the effects of over-fitting for the higher-valued activation thresholds, with those set at 8 and 9 both peaking after five training cycles and then starting to decline. Given this effect and the flattening out of performance of lower valued activation thresholds we do not expect that further training cycles would significantly improve our results.

Overall, the anomaly detection results show that an HTM region equipped with Distal Feedback is able to effectively generate models of the activity of its connected lower-level regions. Specifically, we have shown how such generative

[6]AUC represents the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance [20].

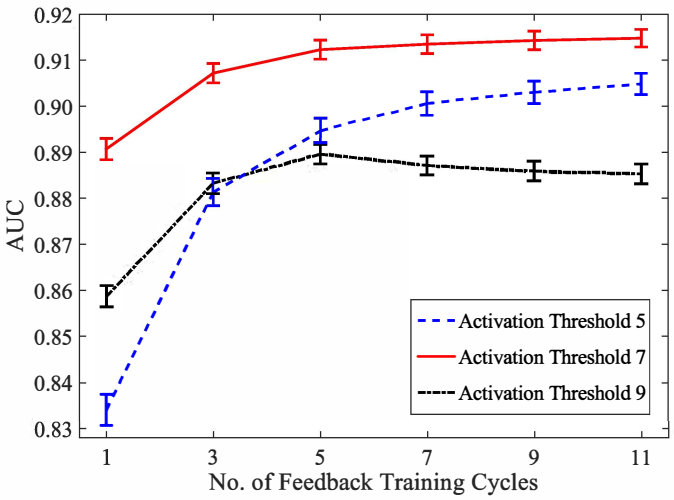

Fig. 11. Performance for activation thresholds of 5, 7, and 9 over different numbers of feedback training cycles. AUC improves with more training cycles but begins to asymptote.

Fig. 12. Original image (left) and reconstruction from feedback (right).

feedback can effectively and accurately detect the presence of an anomalous lower-level region purely on the basis of the non-anomalous activity of surrounding lower-level regions. Here the mismatch between the Distal Feedback and the anomalous lower-level activity represents the system's *surprise* at the anomalous activity. Such a feedback-enabled HTM embodies just the kind of generative model required for Friston's free-energy formulation [1] and achieves this using relatively simple, biologically plausible Hebbian learning principles.

However, implementing Distal Feedback is only a *first step* in developing a hierarchical CLA framework. The next question is how feedback and feedforward information can be combined to probabilistically alter the states of both higher and lower level columns. The face reconstruction experiments begin to answer this question by showing how top-down feedback can be used to override cases of obvious image corruption: Fig. 13 shows that *without* feedback (middle) the original corruptions are reproduced, whereas *with* feedback (bottom) the corruptions are 'repaired' (to a certain extent). This further illustrates how Distal Feedback extends the capability of a single-level HTM, and also how feedback *could* be used to override 'noise' in lower-level column activity. However, this does not answer the more challenging question as to *when* such feedback should be trusted.

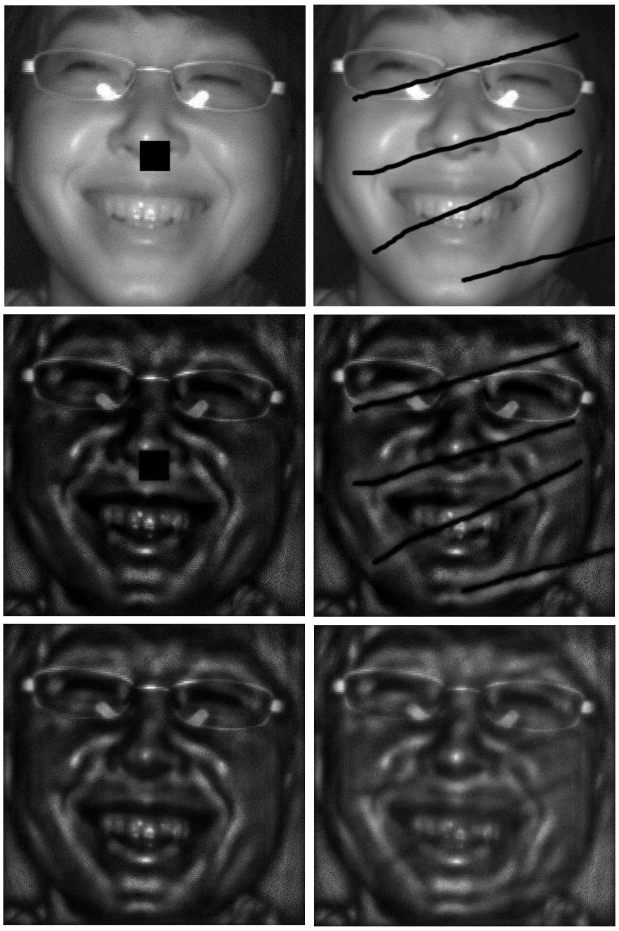

Fig. 13. The corrupted images (top) and reconstructions produced using feedforward activation (middle) and feedback activation (bottom).

VI. Conclusions and Future Work

In this paper we have presented a new Distal Feedback method that implements message passing within an HTM hierarchy and have shown that this method is considerably better than PC-CLA Proximal Feedback at predicting lower-level column activity on a database of facial images. We have further demonstrated the effectiveness of Distal Feedback in providing global contextual information to lower level regions concerning the behaviour of their neighbouring regions. Lastly, we have shown how Distal Feedback can be used to reconstruct corrupted images.

The significance of the paper lies in our having, for the first time, successfully implemented hierarchical message passing within a pure HTM framework and also having demonstrated that such message passing provides useful information that otherwise would be unavailable to a single level system. More broadly, we have proposed a feasible and biologically plausible way in which feedback can occur within the neocortex, and we have developed an HTM hierarchy that can now be evaluated against the more mainstream HPC framework.

Future work will focus on integrating feedback into CLA learning, i.e. by using feedback *and* feedforward information to determine whether or not a column becomes active. In addition we will investigate the effectiveness of only passing prediction errors in feedforward messages (as opposed to the CLA method of passing total activation), and of halting SP learning before convergence.

References

[1] K. Friston, "The free-energy principle: a unified brain theory?" Nature Reviews Neuroscience, vol. 11, no. 2, 2010, pp. 127–138.

[2] D. Knill and A. Pouget, "The Bayesian brain: the role of uncertainty in neural coding and computation," Trends in Neuroscience, vol. 27, 2004, pp. 712–719.

[3] A. Clark, "Whatever next? Predictive brains, situated agents, and the future of cognitive science," Behavioral and Brain Sciences, vol. 36, no. 3, 2013, pp. 181–204.

[4] R. P. N. Rao and D. H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extra classical receptive field effects," Nature Neuroscience, vol. 2, no. 1, 1999, pp. 79–87.

[5] J. Hawkins and S. Blakeslee, On Intelligence. New York: Henry Holt, 2004.

[6] J. Hawkins, S. Ahmad, and D. Dubinsky, "Hierarchical temporal memory including HTM cortical learning algorithms," Numenta Inc., Redwood City, CA, Tech. Rep., 2011.

[7] J. Thornton and A. Srbic, "Spatial pooling for greyscale images," International Journal of Machine Learning and Cybernetics, vol. 4, no. 3, 2013, pp. 207–216.

[8] B. Cowley, A. Kneller, and J. Thornton, "Cortically-inspired overcomplete feature learning for colour images," in PRICAI 2014: Trends in Artificial Intelligence, ser. LNCS, D.-N. Pham and S.-B. Park, Eds., vol. 8862. Heidelberg: Springer, 2014, pp. 720–732.

[9] D. E. Padilla, R. Brinkworth, and M. D. McDonnell, "Performance of a Hierarchical Temporal Memory network in noisy sequence learning," in 2013 IEEE International Conference on Computational Intelligence and Cybernetics. IEEE, 2013, pp. 45–51.

[10] R. McCall and S. Franklin, "Cortical Learning Algorithms with predictive coding for a systems-level cogitive architecture," in Proceedings of the Second Annual Conference on Advances in Cognitive Systems, 2013, pp. 149–166.

[11] P. Agrawal and S. Franklin, "Multi-layer Cortical Learning Algorithms," in 2014 IEEE Symposium Series on Computational Intelligence, Orlando, FL, December 9–12 2014.

[12] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" Vision Research, vol. 37, no. 23, 1997, pp. 3311–3325.

[13] L. Main, B. Cowley, A. Kneller, and J. Thornton, "Evaluating sparse codes on handwritten digits," in AI 2013: Advances in Artificial Intelligence, ser. LNCS, S. Cranefield and A. Nayak, Eds., vol. 8272. Heidelberg: Springer, 2013, pp. 396–407.

[14] R. P. N. Rao, "An optimal estimation approach to visual perception and learning," Vision Research, vol. 39, no. 11, 1999, pp. 1963–1989.

[15] R. McCall, "Fundametal motivation and perception for a systems-level conitive architecture," Ph.D. dissertation, University of Memphis, Memphis, TN, August 2014.

[16] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," Cerebral Cortex, vol. 1, no. 1, 1991, pp. 1–47.

[17] D. Mumford, "On the computational architecture of the neocortex," Biological Cybernetics, vol. 66, no. 3, 1992, pp. 241–251.

[18] B. Zhang, L. Zhang, D. Zhang, and L. Shen, "Directional binary code with application to PolyU near-infrared database," Pattern Recognition Letters, vol. 31, no. 14, 2010, pp. 2337–2344.

[19] S. Li, M. Shao, and Y. Fu, "Locality linear fitting one-class SVM with low-rank constraints for outlier detection," in 2014 International Joint Conference on Neural Networks (IJCNN). IEEE, 2014, pp. 676–683.

[20] T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters, vol. 27, no. 8, 2006, pp. 861–874.