A Cortically Inspired Learning Model

Atif Hashmi and Mikko Lipasti

Department of Electrical and Computer Engineering, University of Wisconsin
1415 Engineering Drive, Madison, WI - 53706, U.S.A.
ahashmi@wisc.edu, mikko@engr.wisc.edu
http://www.ece.wisc.edu/∼pharm

Abstract. We describe a biologically plausible learning model inspired by the structural and functional properties of the cortical columns present in the mammalian neocortex. The strength and robustness of our model is ascribed to its biologically plausible, uniformly structured, and hierarchically distributed processing units with their localized learning rules. By modeling cortical columns rather than individual neurons as our fundamental processing units, we get hierarchical learning networks that are computationally less demanding and better suited for studying higher cortical properties like independent feature detection, plasticity, etc. Another interesting attribute of our model is the use of feedback processing paths to generate invariant representation to robustly recognize variations of the same patterns and to determine the set of features sufficient for recognizing different patterns in the input dataset. We train and test our hierarchical networks using synthetic digit images as well as a subset of handwritten digit images obtained from the MNIST database. Our results show that our cortical networks use unsupervised feedforward processing as well as supervised feedback processing to robustly recognize handwritten digits.

Keywords: Cortical algorithms, Cortical columns, Invariant representation, Feedforward and feedback processing, Pruning, Automatic abstraction, Fault tolerance.

1 Introduction

Understanding of the structural and operational aspects of various components of the mammalian neocortex has significantly increased over the past few decades [1,2,3,4]. This has led to the development of both low level biologically realistic as well as high level biologically inspired computational models. Low level biologically realistic models include the blue brain project [5], DARPA's SyNAPSE project [6], etc. These models use neurons as their basic implementation abstraction and simulate detailed low level behavior of these neurons. Most of these models use Hebbian learning rules [7,8] along with Spike Timing Dependent Plasticity (STDP) [9] for learning and information processing. This makes them quite complex and computationally very expensive. To cope with these issues, high level learning models inspired by the neocortical properties have been proposed. These models implement various neocortical attributes like uniform structure, hierarchy, spatial pooling, temporal pooling, etc. Some of these models include ART [10], HTM [11], Bayesian networks [12], and deep belief networks [13].

K. Madani et al. (Eds.): Computational Intelligence, SCI 399, pp. 373–388.
springerlink.com

Even though these models are computationally quite efficient and implement some behavioral aspects of the neocortex, they are quite divorced from the actual biological structure and properties of the neocortex.

We hypothesize that to develop intelligent models as powerful as the brain, we must adhere to structural and functional properties of the biological example. In this article, we describe a cortical model that uses cortical columns, found in the mammalian neocortex [14], as its basic structural and functional processing units. Since cortical columns are our basic implementation abstraction, our model is inherently computationally efficient and is biologically plausible as well. Our model uses unsupervised feedforward processing and plasticity principles to learn and extract independent features from the input patterns and it uses supervised feedback processing, object permanence, and temporal associativity to develop invariant representations for variations of the same pattern. Using the feedback from higher levels, our model is also able to determine the set of independent features that are sufficient to recognize the different patterns. This is in accordance with the biological studies which show that the human brain relies on a small subset of features in the visual scene to recognize objects [15,16].

To test and validate our cortical model, we use synthetic digit images as well as a subset of handwritten digit images obtained from the MNIST database [17]. Our results show that our cortical networks learn to identify each of the unique digits present in the sample set and also pools variations of the same digit together to develop invariant representations. Moreover, using the feedback from higher levels, our model is also able to determine the set of features that is sufficient to differentiate between digits.

2 Cortical Structures, Organization, and Processing

The human brain can be divided into two main parts: the old brain and the new brain. The old brain mainly constitutes those parts of brain that developed early in evolution. These include pathways from sensory modalities to the new brain, spinal cord, and other parts that deal with instinctual behavior. The new brain, also referred to as the *neocortex*, is part of the brain which is unique to mammals and is highly developed for humans; it accounts for about 77% of the human brain (in volume) [18]. The neocortex is responsible for perception, language, imagination, mathematics, arts, music, planning, and all the other aspects necessary for an intelligent system. It contains virtually all our memories, knowledge, skills, and experiences.

A very intriguing property of the neocortex is its apparent *structural and functional uniformity* [14,19], i.e. the regions of the neocortex that process auditory inputs, appear very similar to the regions that handle visual and other inputs. This uniformity suggests that even though different regions specialize in different tasks, they employ the same underlying algorithm. In essence, the neocortex is a hierarchy of millions of seemingly-identical functional units that are called *cortical columns*. The concept of cortical columns was introduced by Mountcastle in his seminal paper in 1957 [20]. Since then, this concept has been widely accepted and studied. Later studies showed that cortical columns could further be classified into *minicolumns* and *hypercolumns* [21,14,3]. A hypercolumn contains about 50 to 100 minicolumns, and each of these minicolumns consists of around 200 to 500 neurons. The term cortical column is

sometimes used for both types of columns, though, in literature, it usually refers to hypercolumns. The minicolumns within the same hypercolumn share the same receptive field (set of input connections) and are strongly connected with each other via *inhibitory lateral connections*. Studies [21,22] hypothesize that the minicolumns use these inhibitory paths to learn unique/independent features from the input patterns. These hypercolumns are then arranged in the form of a hierarchy throughout the neocortex. Information flows up this hierarchy via *excitatory feedforward paths* and flows down the hierarchy through *feedback paths*.

It is believed that cortical regions operate by progressively abstracting and manipulating increasingly complex notions throughout the neural hierarchy [23]. For instance, from a visual scene, the visual cortex first identifies segments of different orientations, then elementary shapes such as angles and intersections, and increasingly complex combinations, such as objects found in our environment [24]. This automatic abstraction capability for various inputs (visual, auditory, olfactory) partly explains why the neocortex still outperforms traditional computers on a number of tasks, such as object recognition, language learning, and motor control. Emulating such capability is thus a major step in building computing systems that can compete with the processing characteristics of the brain.

Although there exists a monumental amount of literature explaining the aforementioned properties of the neocortex, certain alternative properties of the neocortex have not been sufficiently explored. One of these properties is the ability to determine the set of features that the neocortex uses to recognize objects. Obviously, the neocortex does not require all of the object's features to recognize it. For example, Maw et.al [15] show that while recognizing a human face, subjects tend to gaze more at the eyes, lips, and nose. Thus, out of all the features that constitute a human face the neocortex uses a small subset to process the facial recognition task. Sigala et.al [16] shows similar results in more details. Thus, within the neocortex, there exists of notion of a sets of features that are sufficient to differentiate visual patterns.

3 Cortical Model Description

3.1 Hypercolumn Abstraction

As mentioned in Section 1, we model cortical columns as our basic structural and functional implementation abstraction. Figure 1 shows the architecture of the basic functional unit in our cortical model. A hypercolumn consists of multiple minicolumns that are strongly connected with each other via horizontal inhibitory connections. All of the minicolumns within a hypercolumn share the same receptive field. A receptive field is defined as the region within sensory input that is associated to a hypercolumn.

3.2 Unsupervised Feedforward Processing and Independent Feature Learning

In our model each of the minicolumns within a hypercolumn learns to identify independent features from the input patterns using lateral inhibitory paths. This is in accordance with the biological discussion presented in Section 2. In this section, we provide detailed discussion on how each of the minicolumns learns to identify unique patterns without any supervision.

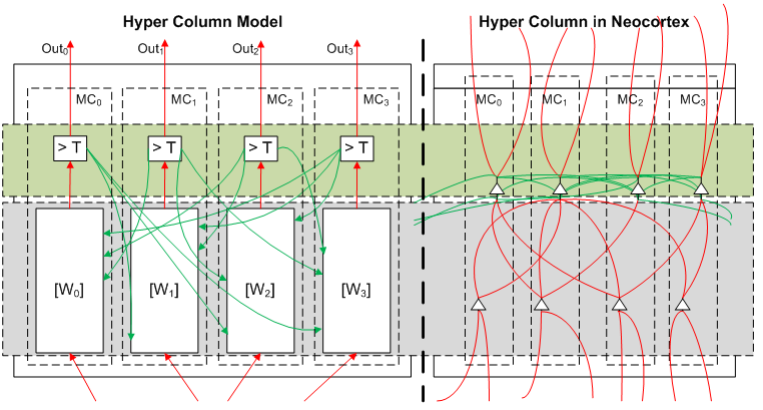

Fig. 1. Mapping between hypercolumn network and feedforward circuitry of a hypercolumn in the neocortex. Left: A hypercolumn network in our model with four minicolumns. Right: Structure of a biological hypercolumn.

Random Activations and Initial Learning. Initially all the minicolumns within a hypercolumn are initialized with very weak random weights. Thus, they show no preference for any pattern that might occur within their receptive field. Since our minicolumns also model the stochastic nature of neurons by including random neocortical firing behavior [25,26], they exhibit high activations over random intervals. When the random activation of a specific minicolumn coincides frequently with various stable occurrences of the same pattern, the minicolumn adjusts its weights so that the correlation between the weights and the input patterns increases. Thus over time, that minicolumn develops a firing preference for that specific pattern. While this random activation of minicolumns may not initially seem productive, this behavior is harnessed to make the model fault-tolerant, improves the model's training time, and mimics the behavior of its biological inspirations.

Evaluating Output of Minicolumns. Each of the minicolumns contains a set of weights W initialized to random values which are close to zero. During each training epoch, each of the minicolumns evaluates the dot-product $DP = \sum_{i=1}^{N} X_i.W_i$ between its weights $\boldsymbol{W}$ and the input $\boldsymbol{X}$. The result of the dot-product becomes the input to the activation function given by,

$$\frac{1.0}{1.0 + e^{(-\frac{DP - cutoff}{\beta})}} + \alpha \times \sum |W_i| \tag{1}$$

Here, $cutoff = \phi \times \sum |W_i|$. ϕ determines the error tolerance of the minicolumn. β defines the sharpness of the activation function while α controls the effect of weight strength of a minicolumn on its output. The minicolumn is said to fire if the value of its activation function is greater than a determined threshold.

Lateral Inhibition and Independent Feature Identification. When an input $\boldsymbol{X}$ is presented to the hypercolumn, none of the untrained minicolumns fire for that input.

However, if the random firing activity of a minicolumn coincides with the occurrence of an input pattern, that minicolumn adjusts its weights so that the dot-product between the input and the weights is improved. This is achieved by strengthening the weights corresponding to the inputs X_i that are currently active. Thus, over multiple iterations a minicolumn learns to identify a feature that initially coincided with the random activity of the minicolumn. At the same time, each minicolumn inhibits neighboring minicolumns from firing via lateral inhibitory connections for the pattern it has already learned to recognize. If multiple minicolumns fire at the same time, the one with the strongest response inhibits the ones with weaker responses. The inhibited minicolumns then weaken their weights corresponding to highly active X_i so that their dot-product with the input is minimized. As a result of this process, the hypercolumn network is able to recognize unique patterns without any supervision. A very interesting byproduct of having minicolumns learn independent features through lateral inhibition is inherent fault tolerance i.e. if a minicolumn that was firing for a feature suddenly dies (permanent hardware or software error in a future synthetic application), over time, another available neighboring minicolumn will start firing for that feature. This makes our hypercolumn structure inherently tolerant to permanent faults.

Weight Update Rules. Each time a minicolumn fires it modifies its weights so that its correlation with the input pattern that has caused it to fire increases. Weights are strengthened using the following update rule.

$$W_i = X_i \times \left(W_i + \left(C_1 + \gamma \times \frac{1.0}{1.0 + e^{(-\frac{W_i - C_2}{\beta})}} \right) \right) \tag{2}$$

Here, X_i is the input corresponding to W_i, C_1 defines the minimum amount of update added to the current W_i and C_2 defines how the present W_i will affect the weight update. In our weight strengthening rule, the update added to W_i is dependent upon the present value of W_i as well. This means that if W_i is strong it will get a higher update value. This is in accordance with biological data [26,27].

In the case when a minicolumn is inhibited, it modifies the weights using the following update rule.

$$W_i = X_i \times (W_i - \delta) \tag{3}$$

Here, δ defines the weight update rate in the presence of inhibition.

3.3 Hierarchical Arrangement of Hypercolumns

To perform complex tasks the hypercolumns can be arranged in the form of a hierarchy. Lower hierarchical levels identify simple features and communicate their output to the higher levels via feedforward paths. Each of the higher level hypercolumns receives inputs from multiple lower level hypercolumns. In this manner the activations flow up the hierarchy and the minicolumns in the top-level hypercolumns train themselves to identify each of the complex unique pattern from the input. Each level of this hierarchy behaves the same way as different levels of the visual cortex i.e. lower level hypercolumns detect edges, and the hypercolumns at the higher levels detect progressively complex features. It should be noted that our hierarchical model supports any complex hierarchical arrangement of hypercolumns.

3.4 Supervised Feedback Processing and Invariant Representations

Our feedforward learning process enables our cortical hierarchy to learn unique features from the input patterns. Each of the minicolumns can withstand and fire for patterns with small variations but patterns with significant variations are recognized as different features. This means that two variations of the same pattern might be recognized as two different features. To resolve this issue and generate invariant representation for variations of the same pattern, we make use of our supervised feedback processing algorithm.

Algorithm 1. Pseudo code for generating invariant representations within a minicolumn using supervised feedback

```
if feedback > 0 then
  if hasNotFired then
    if hasMaxFiringHistory then
      UpdateSynapticWtsExcitatory(feedback)
    end if
  else
    if hasMaxFiringHistory then
      UpdateSynapticWtsExcitatory(feedback)
      if isStable then
        for i = 1 to N do
          if IsActive(child[i]) then
            SendFBToChild(i, feedback)
          end if
        end for
      end if
    else
      UpdateSynapticWtsInhibitory(feedback)
    end if
  end if
end if
```

Lets assume that our hierarchical network has started to recognize a pattern. Now it is exposed to another variation of the same patterns that is quite different from the previous one e.g. two different variations of a handwritten digit. At this point, only some of the minicolumns within the hierarchy might fire. As a result, the top level minicolumn that is supposed to fire for that pattern might not fire. If this behavior persists, new minicolumns will train themselves to recognize features in the new variation that are quite different from the original pattern. Over time, that new variation will be identified as a new pattern. This will be marked by firing of a minicolumn in the top level of the hierarchy. At this point, the top level hypercolumn receives a feedback signal. This feedback signal forces the minicolumn firing for the original pattern to fire and also inhibits the minicolumn that is firing for the new variation. Now, the minicolumn receiving excitatory feedback also adjusts its weights so that it fires for the new variation as well while the inhibited minicolumn changes its weights so that it does not fire for that input pattern. Thus over multiple exposures, the minicolumn firing for the original pattern will also start to fire for the new variation. Once the top level minicolumn starts to give a

stable activation for both the variations, it will start to send the feedback signal down so that lower level minicolumns can also create invariant representations. The amount of feedback sent to each of the lower level minicolumns is proportional to its firing history i.e. if a minicolumn has been firing a lot in the past, it will get stronger feedback. Thus, over time most active minicolumn ends up pooling its child minicolumns to generate invariant representations and inhibits its neighbors from firing. This results in significant resource optimization. The process of generating invariant representations within a minicolumn using feedback is explained in the pseudo-code provided in Algorithm 1. In Algorithm 1, $UpdateSynapticWtsExcitatory$ models the functionality of Equation 2 while $UpdateSynapticWtsInhibitory$ models Equation 3.

3.5 Learning Spatial Correlations from Past Experiences

To determine the set of features sufficient for recognition of unique images, we consider the spatial correlations that exist among the occurrence of different independent features constructing the objects in the dataset. Since each of the minicolumns trains itself to identify independent features from its input, we can determine the spatial correlation among various minicolumns by observing their firing patterns.

Table 1. Location of minicolumns in the hypercolumn hierarchy identifying different features and shapes

Level	Hypercolumn	Minicolumn	Recognizes
0	0	0	Feature a
0	0	1	Feature b
0	0	2	Feature c
0	1	0	Feature a
0	1	1	Feature b
0	1	2	Feature c
1	0	0	Shape A
1	0	1	Shape B
1	0	2	Shape C
1	0	3	Shape D
1	0	4	Shape E

To illustrate our methodology for determining the spatial correlations, we use a simple 2-level hierarchical network shown in Figure 2. Assume that this network is trained using the dataset shown in Figure 3. Each of the four elements of the dataset is a synthetic image of size 2x4. Each image can be divided into two halves, i.e. the first 2x2 grid and the second 2x2 grid separated by the dotted line in Figure 3. Within each of the two grids, only three unique features occur: a horizontal line across the top (Feature *a*), a vertical line on the left (Feature *b*), and a diagonal line (Feature *c*). Once the network is trained with each of the 2x4 images, each of the minicolumns within the Level 0 hypercolumns start to recognize any one of the three unique features. At the same time, each of the four minicolumns at Level 1 starts to recognize any one of the four

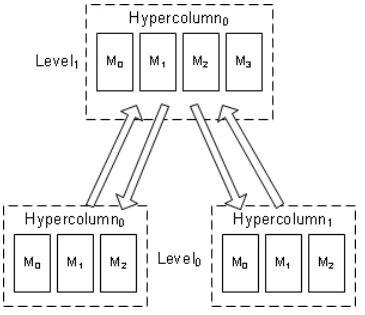

Fig. 2. Block level diagram of a 2-level hierarchical hypercolumn network

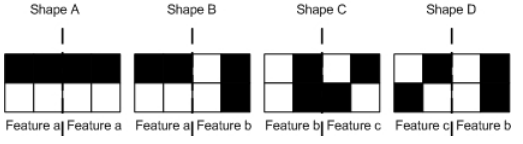

Fig. 3. Set of features used to train 2-level hierarchical network

different shapes. Table 1 shows the details about the patterns identified by each of the minicolumns in different hypercolumns in the 2-level hierarchy.

Once the 2-Level hierarchy is in a stable state i.e. all the four shapes are recognized by the Level 1 hypercolumn, the Level 1 hypercolumn starts to create spatial correlations among the occurrences of different features recognized by the minicolumns in each of the Level 0 hypercolumns. These spatial correlations help in determining the set of features sufficient for recognition of each of the unique shapes. A spatial correlation is created between two features if they co-occur frequently. Once a spatial correlation has been created, occurrence of a feature can be predicted by any occurrence of the feature it is spatially correlated to. For example, in Figure 3 feature *a* in the first 2x2 grid always occurs with features *a* and *b* in the second 2x2 grid. This means that a spatial correlation between feature *a* on the left 2x2 grid and features *a* and *b* on the right 2x2 grid is developed. Feature *b* on the left 2x2 grid co-occurs with features *a* and *c* on the right 2x2 grid. Thus, a spatial correlation is maintained among these features. Finally, feature *c* on the left 2x2 grid only co-occurs with feature *b* on the right 2x2 grid so there is just one spatial correlation for feature *c* i.e. feature *b*.

Figure 4 shows the associations among various features that are created due to the spatial correlations that exist among them. In the figure, we can see that Level 0 Hypercolumn 0 Minicolumn 0 (L0H0M0) is connected to L0H1M0 through L1H0M0 which detects Shape A. Similarly, L0H0M0 is connected to L0H1M1 through L1H0M1 which detects Shape B. Other associations in Figure 4 are also created in the same manner. Figure 4 details an association graph that shows the spatial correlations which exist among different features that are detected by the Level 0 minicolumns.

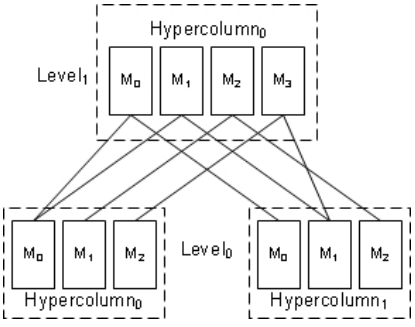

Fig. 4. A graphical representation of spatial correlations created among various features recognized by the Level 0 minicolumns after the hierarchical network reaches a steady state

4 Determining the Set of Features Sufficient for Recognition of Unique Shapes

To determine the set of features sufficient for identifying the shapes exposed to the network, we use the spatial correlation graph like the one shown in Figure 4. At each level of the hierarchy, the spatial correlations between different minicolumns are observed and the associations that provide redundant information are pruned. For example, if minicolumn A is just spatially associated to minicolumn B and minicolumn B is spatially associated to some other minicolumn a well, then the feature recognized by minicolumn A is sufficient enough to identify the shape typically recognized by minicolumns A and B together. Thus, the association from minicolumn B can be pruned. Finally, if a minicolumn has no associations coming out of it, then that minicolumn can also be pruned.

To better explain our algorithm, we apply it to the spatial correlation graph shown in Figure 4. Initially, L0H0M0 is selected. Since there are two edges out of L0H0M0 it is left as is. Then L0H0M1 is selected. Since there is just one association from L0H0M1, the association between L0H1M2 and L1H0M2 is declared useless and is pruned. For L0H0M2, there is again just one edge connecting L0H0M2 to L0H1M1 through L1H0M3. Thus, the edge from L0H1M1 to L1H0M3 is pruned because L1H0M3 can recognize Shape 4 just by using the information provided by L0H0M2. The same is true for L0H1M0. Thus, the edge between L0H0M0 and L1H0M0 is pruned. Note that in many cases the pruning algorithm has the freedom to choose which redundant connections to retain, and which to prune. After this no more changes to the network connectivity take place because all of the remaining edges provide useful information to recognize each of the unique shapes. The resultant graph after applying the spatial correlation based algorithm is shown in Figure 5.

In Figure 5, we see that after applying our algorithm, five out of ten edges between Level 0 and Level 1 are pruned. Furthermore, L0H1M2 has no more feed-forward edges to the next level of the hierarchy which renders it totally useless. Thus, L0H1M2 can

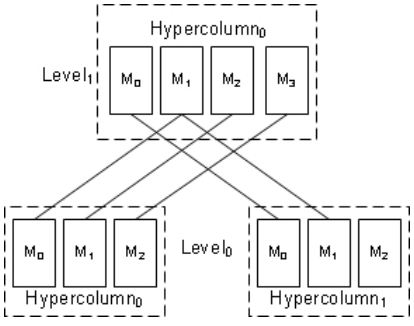

Fig. 5. Pruned graph obtained after applying the spatial correlation based algorithm on the graph shown in Figure 4

be pruned. This reduces the amount of computations required as pruning connections means less number of computations are required to recognize a shape by the mini-columns at Level 1 as the unnecessary connections are not considered for recognition.

5 Experiments and Results

To test and validate different properties of our cortical architecture and to evaluate its learning and recognition performance, we used a subset of handwritten digit images obtained from the MNIST database [17]. For this digit recognition task, we created a hierarchical network with 6 levels. We initialized this network as described in Table 2. Level 0 corresponds to the lowest level in the hierarchy. All the digits in the MNIST database are in the form of 28x28 pixel wide black and white images. Out of the 28 rows, top 2 and bottom 2 rows were always black. Thus, in our experiments, we ignored these rows to save on execution time. Each of the remaining rows becomes the input to one of the twenty four Level 0 hypercolumns.

Table 2. Detailed description of the hierarchical network created for recognition of handwritten digit images

Level	Hypercolumns (HC)	Minicolumns/HC
5	1	100
4	1	200
3	3	200
2	6	200
1	12	300
0	24	500

5.1 Experiment 1: Independent Feature Recognition

In the first experiment, we validate our feedforward information processing and learning algorithm. For this experiment, we disable the feedback processing and study how the network learns independent features from the input patterns. Since there was no

feedback, we anticipate that in Level 5 (top most level) of the hierarchy, variations of same digits will be recognized by different minicolumns. For this experiment, we took 100 handwritten digit images (10 variations of each digit) from the MNIST database and trained and tested our network with them till it achieved 100% recognition rate. In steady state, top level hypercolumn contains 89 minicolumns that learned to recognize various digit patterns present in the input dataset. 11 digit variations are pooled with some other variation of the same digit due to spatial similarities.

5.2 Experiment 2: Feedback Processing and Invariant Representation

To test how our feedback processing algorithm generates invariant representations, we use the same hierarchical network mentioned above. For the input dataset, we use the same 100 digit images (10 variations for each digit) for training as used in Experiment 1 and train the network with these images till the network achieved 100% recognition rate. At this point, we notice that there were only 10 minicolumns in the top level hypercolumn that were firing in response to the digits being exposed to the network. This means that there is just one minicolumn firing for all the different variations of the same digit. We also evaluate the resource optimization achieved through feedback processing. To do that we calculate the number of active minicolumns in the hierarchical network with and without feedback. In steady state, without feedback the network uses 3876 minicolumns while with feedback it only uses 1283 minicolumns. Thus, our feedback processing algorithm results in about 3x resource optimization.

5.3 Experiment 3: Robustness to Test Images

In this experiment, we test the robustness of our cortical network to the patterns not present in the training dataset. For this experiment we again use the same hierarchical network described above. We use 400 handwritten digits images (40 variations of each digit) training images and 40 test images (4 variations of each digit). We then train the network with the images in the training dataset till the network achieves 100% recognition rate and is in a stable state i.e. all the levels in the hierarchy have generated invariant representations for all the input digit variations. Figure 6 shows the recognition rate of the network as the number of images in the training dataset is increased from 10 to 400. For this experiment, recognition rate is defined as the percentage of the images in the test dataset that were recognized correctly.

In the future we are planning to extend our cortical architecture so that it can run on NVidia GPUs. This will let us create and test large hypercolumn based networks and will improve the recognition rates further.

5.4 Experiment 4: Inherent Fault Tolerance

This experiment validates the inherent fault-tolerant property of our cortical network. For this experiment, we use the same hierarchy as described above and use 200 handwritten digit images for training. To reduce the execution time for each epoch, we limit the feedback processing to Level 5 (top-most level) of the hierarchy only. Initially, we

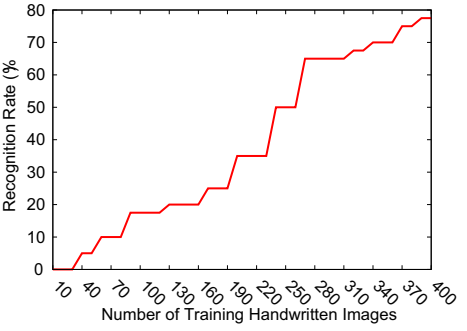

Fig. 6. Recognition rate of the network for handwritten test digit images as the number of training images is increased

train the hierarchy with all the 200 images till it achieves 100% recognition rate. At this point we corrupt 5% of the total number of minicolumns throughout the hierarchy. This was done by randomly selecting minicolumns and forcing their output to stay 0 permanently. Then we evaluate the recognition rate of the hierarchy with all the 200 training images to determine the amount of loss in recognition. Then we train the damaged hierarchy with the same training images and evaluate the peak recognition rate for the training images. We repeat this cycle multiple times corrupting 5% of the original number of minicolumns every time to observe how the hierarchy behaves as we inject more and more permanent faults. Table 3 shows the behavior of our cortical network in the presence of permanent faults.

Table 3. Evaluation of the inherent fault tolerance property of our cortical network. Initial Recognition Rate means the recognition rate (percentage) measured immediately after the faults are injected. Peak Recognition Rate means the maximum recognition rate achieved through training the damaged network.

Fault Injection Attempt	Initial Recognition Rate (%age)	Peak Recognition Rate (%age)
1	92	100
2	89	100
3	90	100
4	88	100
5	88	94
6	82	82
7	71	71
8	65	65

When Fault Injection Attempt is 5 that means that we have damaged 25% of the total minicolumns originally present in the hierarchy. For this attempt, after retraining the damaged hierarchy, it achieves the peak recognition rate of 94%. This is due to the fact that some of the hypercolumns ran out of the minicolumns that were idle.

As a result the features being recognized by the minicolumns that were damaged could not be relearned. This experiment also shows that as long as there are idle resources available in the network, it can recover from permanent faults.

5.5 Experiment 5: Determining Set of Sufficient Features

For this experiment, we create a 3 level hierarchical network. Each of the hypercolums in hierarchy is initialized to have 12 minicolumns. This network is initially expose it to synthetic digit images from 0 to 9 shown in Figure 7. Each of the rows in the 7x7 digit image is exposed to one of the hypercolumns in Level 0 of the hierarchy. For example, the highlighted row in Figure 7 is exposed to the first hypercolumn in Level 0. Similarly, the second row is exposed to the second hypercolumn in Level 0 and so on. Once the hierarchical network achieves the steady state, ten of the twelve minicolumns available in the Level 2 hypercolumn fire for a unique digit image. The remaining two minicolumns keep on thrashing because they do not have any new digit to recognize.

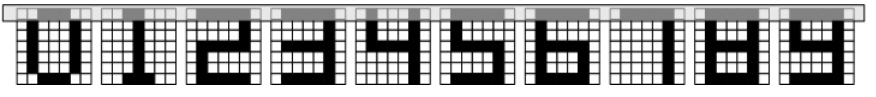

Fig. 7. Input dataset exposed to the hierarchical network to determine the set of sufficient features

Fig. 8. Set of features sufficient enough to recognize each unique shape. The features sufficient enough to identify a digit are overlaid in black on top of the actual image.

Figure 8 shows the set of features sufficient to recognize each of the ten digits. The sufficient features are shown in black. For example, to recognize a zero, the only feature that matters is the horizontal line on top of zero. Similarly, to recognize the one, the shorter horizontal line on the top is sufficient enough. In the absence of our spatial correlation based algorithm to determine the set of features sufficient for recognition of each of the digits, a total of 55 minicolumns are utilized to identify all the ten digits. With the spatial correlation based algorithm, only 33 minicolumns are required. Thus, there is an approximate 40% savings in computational resources.

Next, we again create a 3 level hierarchical network and initialize each of the hypercolumns to have 40 minicolumns. The input to this hierarchical network consists of 36 7x7 synthetic images (10 digits + 26 alphabets). To evaluate the increase in the sufficient set of features and in the number of minicolumns utilized in the presence and absence of our spatial correlation based algorithm with an increase in the unique shapes in the dataset, we exposed alphabets from A to Z to the hierarchical network along with the digits from 0 to 9.

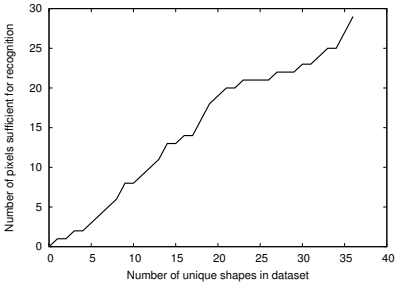

Fig. 9. Increase in the number of features sufficient to recognize all the shapes in the dataset as the number of unique shapes in the dataset is increased

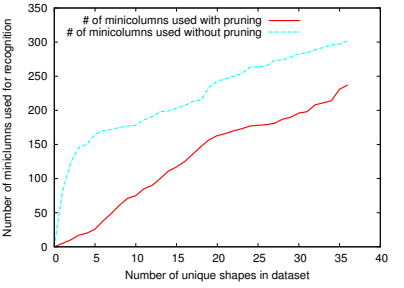

Fig. 10. Increase in the number of minicolumns required to recognize all the shapes in the dataset as the number of unique shapes in the dataset is increased

The graph in Figure 9 shows the increase in the set of features sufficient for recognition as the number of unique shapes in the dataset is increased. In this graph, the number of unique shapes in the dataset are along the x-axis while the number of sufficient features identified by the hierarchical network is along the y-axis. A linear increase in the number of sufficient features identified suggests that each of the new unique shapes adds contains at least one sufficient feature.

The graph in Figure 10 shows the increase in the number of minicolumns as the number of unique shapes in the dataset is increased. The solid line shows the number of minicolumns in all the levels of the hierarchy identified as sufficient by our spatial correlation based algorithm while the dotted line shows the total number of minicolumns used by the hierarchical network to recognize the same set of shapes if the spatial correlation based algorithm is not used. We see that the number of minicolumns required when the spatial correlation based algorithm is used is far less than the number of minicolumns used in the absence of the spatial correlation algorithm. A linear increase in the solid line in Figure 10 suggests that almost all the unique shapes in the dataset introduce a feature that is not being used by any other shape.

6 Conclusions

We describe a biological plausible learning model that implements the working of cortical columns as its basic structural and functional abstraction. We demonstrate that building models based on the properties of cortical columns can be computationally efficient as well as biologically plausible. Using these models, we can study various neocortical properties like independent feature identification, feedback, plasticity, invariant representation, and resource management. Our results show that such models are inherently tolerant to permanent faults (either in hardware or in software). Using our spatial correlation based pruning algorithm, we significantly improve the resource utilization of our hierarchical hypercolumn networks.

References

1. Nicholls, J., Martin, A., Wallace, B., Fuchs, F.: From Neuron To Brain. Sinauer Associates Ins., 23 Plumtree Road, Sunderland, MA, USA (2001)
2. Hawkins, J., Blakeslee, S.: On Intelligence. Henry Holt & Company, Inc. (2005)
3. Hirsch, J., Martinez, L.: Laminar processing in the visual cortical column. Current Opinion in Neurobiology 16, 377–384 (2006)
4. Aimone, J., Wiles, J., Gage, F.: Computational influence of adult neurogenesis on memory encoding. Neuron 61, 187–2002 (2009)
5. Markram, H.: The blue brain project. In: SC 2006: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, vol. 53. ACM, New York (2006)
6. DARPA: Systems of neuromorphic adaptive plastic scalable electronics (synapse) (2008), http://www.darpa.mil/dso/thrusts/bio/biologically/synapse/
7. Clopath, C., Longtin, A., Gerstner, W.: An online hebbian learning rule that performs independent component analysis. In: Proceedings of Neural Information Processing Systems (2007)
8. Martinetz, T.: Competitive hebbian learning rule forms perfectly topology preserving maps. In: International Conference on Artificial Neural Networks, ICANN, pp. 427–434 (1993)
9. Arthur, J., Boahen, K.: Learning in silicon: Timing is everything. In: Proceedings of Advances in Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 18, pp. 75–82 (2006)
10. Carpenter, G., Grossberg, S., Rosen, D.: Art2-a: An adaptive resonance algorithm for rapid category learning and recognition. Neural Networks 4, 493–504 (1991)
11. Hawkins, J., George, D.: Hierarchical temporal memory (2006), http://www.numenta.com/numenta_htm_concepts.pdf
12. George, D., Hawkins, J.: A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In: Proceedings of International Joint Conference on Neural Networks. IEEE International Joint Conference on Neural Network, vol. 3, pp. 1812–1817 (2005)
13. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. 18, 1527–1554 (2006)
14. Mountcastle, V.: An organizing principle for cerebral function: The unit model and the distributed system. In: Edelman, G., Mountcastle, V. (eds.) The Mindful Brain. MIT Press, Cambridge (1978)
15. Maw, N., Pomplun, M.: Studying human face recognition with the gaze-contingent window technique. In: Proceedings of the Twenty-Sixth Annual Meeting of Cognitive Science Society, pp. 927–932 (2004)
16. Sigala, N., Logothetis, N.: Visual categorization shapes feature selectivity in the primate temporal cortex. Nature 415, 318–320 (2002)
17. Lecun, Y., Cortes, C.: The mnist database of handwritten digits (1998), http://yann.lecun.com/exdb/mnist/
18. Swanson, L.: Mapping the human brain: past, present, and future. Trends in Neurosciences 18, 471–474 (1995)
19. Mountcastle, V.: The columnar organization of the neocortex. Brain 120, 701–722 (1997)
20. Mountcastle, V.: Modality and topographic properties of single neurons of cat's somatic sensory cortex. Journal of Neurophysiology 20, 408–434 (1957)
21. Hubel, D., Wiesel, T.: Receptive fields, binocular interactions and functional architecture in cat's visual cortex. Journal of Physiology 160, 106–154 (1962)

22. Hubel, D., Wiesel, T.: Receptive fields and functional architecture of monkey striate cortex. Journal of Physiology 195, 215–243 (1968)
23. Peissig, J., Tarr, M.: Visual object recognition: do we know more now than we did 20 years ago? Annu. Rev. Psychol. 58, 75–96 (2007)
24. Grill-Spector, K., Kushnir, T., Hendler, T., Edelman, S., Itzchak, Y., Malach, R.: A sequence of object-processing stages revealed by fmri in the human occipital lobe. Hum. Brain Map. 6, 316–328 (1998)
25. Freeman, W.: Random activity at the microscopic neural level in cortex ("noise") sustains and is regulated by low-dimensional dynamics of macroscopic activity ("chaos"). International Journal of Neural Systems 7, 473–480 (1996)
26. Rokni, U., Richardson, A., Bizzi, E., Seung, H.: Motor learning with unstable neural representations. Neuron 64, 653–666 (2007)
27. Seung, H.: Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. Neuron 40, 1063–1073 (2003)