Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2011-021
CBCL-298

April 14, 2011

Approximations in the HMAX Model

Sharat Chikkerur and Tomaso Poggio

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Approximations in the HMAX Model*

Sharat Chikkerur, Tomaso Poggio

Oct 2007

Abstract

The HMAX model is a biologically motivated architecture for computer vision whose components are in close agreement with existing physiological evidence. The model is capable of achieving close to human level performance on several rapid object recognition tasks. However, the model is computationally bound and has limited engineering applications in its current form. In this report, we present several approximations in order to increase the efficiency of the HMAX model. We outline approximations at several levels of the hierarchy and empirically evaluate the trade-offs between efficiency and accuracy. We also explore ways to quantify the representation capacity of the model.

1 Introduction

The HMAX model is a biologically motivated model that mimics the organization of the first few stages for the human visual pathway. More specifically, it tries to explain the purely feed forward mechanism for rapid object recognition that occurs in the first 100-150ms of visual processing. The model is organized with alternating layers consisting of simple(S) units and complex(C) units (See Figure 1). S units have specific preferred input patterns (such as oriented lines) and perform simple pattern matching. Complex units spatially pool the outputs from the S units and select the maximum, thus providing spatial invariance. Complexity of the preferred input to the S units and the degree of location and scale invariance achieved at the C units increases as we move up the hierarchy. The model requires very few training examples and competes with the state-of-the-art system in object recognition task. A detailed description of the HMAX model can be found in [2]. While the HMAX model is surprisingly accurate and a biologically faithful, it is currently bounded by computation and is not very useful for real-life application. In this report, we study a specific instantiation of the model, with two S,C layer pairs [4] that simulate visual processing in area V1/V2-V4/PIT.

1.1 Computational Complexity

- **S1 layer**: The S1 layer computation consists of performing normalized cross correlation between a bank of 64 gabor filters (16 scales and 4 directions, Figure 2) and the image. This operation is non-linear and has a computational complexity of $O(N^2M^2)$, where $M \times M$ is the size of the filter and $N \times N$ is the size of the image.
- **C1 layer**: The C1 layer is computed by performing a spatial and scale-level max pooling operation on the S1 outputs.The computational cost of this operation is $O(N^2M)$, since max operation can be performed in a separable fashion (x direction first and then y direction). In the current instantiation, the C1 layer consists of 8 bands of pooling.

*The quantitative evaluations were done during Oct 2007 and may not reflect state-of-the-art in HMAX

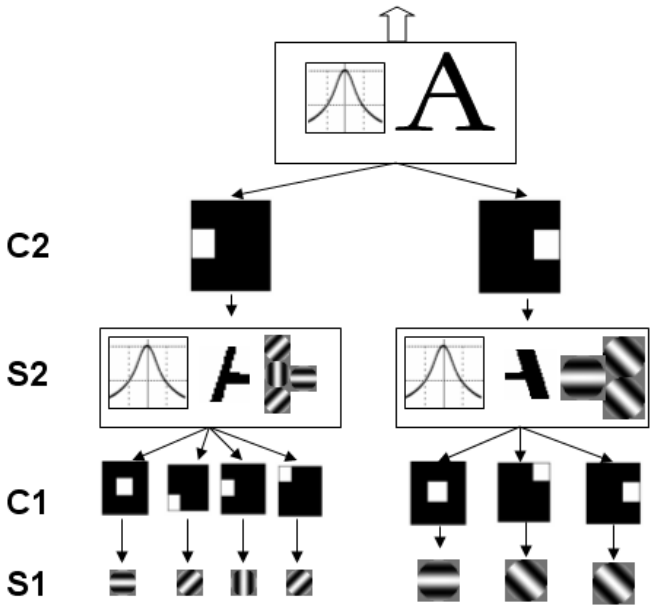

Figure 1: HMAX Model Architecture. S1 layers consists of simple gabor filter at several scales and orientation. C1 pools the filter outputs spatially and across adjacent scales. S2 is tuned to conjunctions of orientations. C2 provides further spatial and scale invariance. The C2 outputs are directly fed to a classifier

- **S2 layer**: During training, several hundred C1 patches of different sizes are randomly selected from the training images and serve as prototype features that represent a single object (class specific dictionary) or all objects (universal dictionary) [3]. Each S2 unit is designed to detect the presence of the prototype within the C1 unit responses. Each patch serves as a filter that is correlated with the 8 C1 bands to generate a tuning response map. Deriving the S2 unit outputs is the most computationally intensive part of the model, involving $O(PN^2M^2)$ computation. Here P represents the number of patches.
- **C2 layer**: The C2 layer provides the final invariance by detecting the maximum response of the corresponding S2 units over all scales and positions. There are as many C2 units as the number of patches in the representation dictionary. The computational complexity of this step is $O(N^2MP)$.

1.2 Approximations

In this report, we study several approximations to this hierarchical model. The approximations can be categorized based on which layer of the hierarchy does it effect or more logically on the nature of approximations as outlined in the following sections.

- Approximating the filters (S1)
- Approximating the features (S2)
- Approximating the outputs (C2)

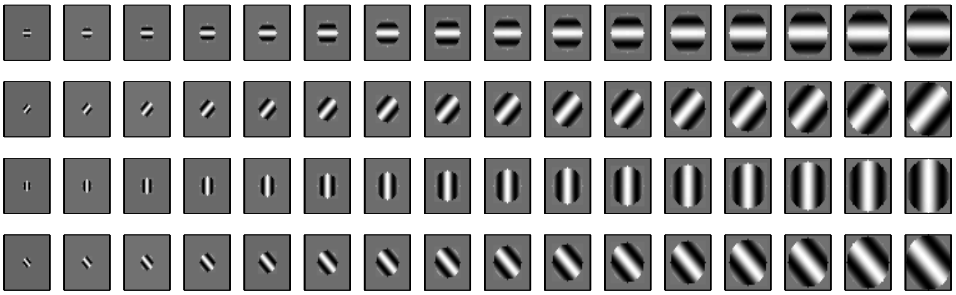

Figure 2: Bank of gabor filters used to compute S1 features. The receptive fields sizes and other parameters are outlined in [3].

2 Approximations at the S1 Layer

The S1 layer is obtained by performing pattern matching with short lines at several different scales and orientations. The oriented lines are modeled as anisotropic Gabor filters given by,

$$G(x,y) = \exp\left(-\frac{u^2+\gamma v^2}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}\right) \quad (1)$$

$$\text{where,} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (2)$$

Here γ represents the eccentricity/anisotropy of the filter, θ is the orientation and λ is the wavelength of the filter. S1 unit response $S1(s,d)$ at scale s and direction d is computed using a normalized cross correlation operation

$$S1_{(s,d)}(x,y) = \frac{\hat{G}(x,y,\theta) * I(x,y)}{I^2(x,y) * H(x,y)} \quad (3)$$

Here $\hat{G}$ is the zero mean,unit normal version of the filter and $H(x,y)$ a filter with all ones and the same size as the Gabor filter.

2.1 Separable Filters

As indicated before, the complexity of this operation is $O(N^2M^2)$, where $M \times M$ is the size of the filter and $N \times N$ is the size of the image. If the filter G(x,y) was separable ($G(x,y) = g(x)h(y)$), then we could have replace the point wise 2D convolution, with two 1D convolutions corresponding to convolution by $g(x)$ in the x-direction and $h(y)$ in the y direction. Separable convolution reduces the complexity from $O(N^2M^2)$ to $O(N^2M)$ providing substantial saving when the filter size is large. In order to reduce the complexity of computation in the S1 layer, we make the anisotropic Gabor filters separable by approximating them using its isotropic ($\gamma = 1$,circular) version. Consider the complex version of the Gabor filter given by,

$$G(x,y) = \exp\left(-\frac{u^2+v^2}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}\right) \quad (4)$$

$$= \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}(x\cos(\theta)+y\sin(\theta))\right) \quad (5)$$

$$= Re\left\{f(x)g(y)\right\} \quad (6)$$

$$f(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \times exp(ix\cos(\theta)) \quad (7)$$

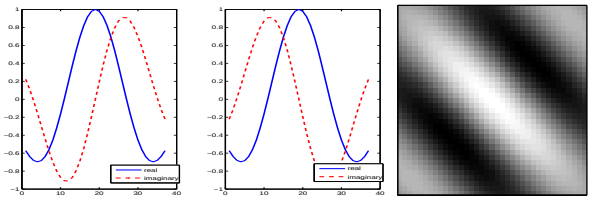

Figure 3: The figure shows f(x) and g(y) and their product. It can be seen that it approximates the oriented gabor filter fairly well

$$g(y) = \exp\left(-\frac{y^2}{2\sigma^2}\right) \times exp(iy\sin(\theta)) \tag{8}$$

$$\tag{9}$$

The convolution can now be done in a separable fashion by first convolving in the x-direction with f(x) and then convolving the result with g(y) in the y-direction. More formally, result can now be expressed as

$$I(x,y) * G(x,y) = I(x,y) * f(x)\delta(y) * g(y)\delta(x) \tag{10}$$

The cost of normalized cross correlation is now $O(cN^2M)$, $c = 8$, due to complex valued arithmetic. However, the cost of complex computations is offset by the gain obtained by separability.

2.2 SVD Approximations

One of the problems with separable filtering is convolution in the complex domain. While there are several libraries and implementation with efficient and hardware accelerated implementations of real valued 2D convolution, As an alternative approximation, We can treat the anisotropic Gabor filter G(x,y) as a two dimensional matrix that has a low rank, since the filter is fairly smooth at lower frequencies. Singular value decomposition can then be used in order to decompose the filter into a linear sum of *real* separable filters. More specifically, using SVD, we obtain

$$G = USV^T \tag{11}$$

$$G(x,y) \approx \sum_{i=1}^{K} s_i u_i(y) v_i(x) \tag{12}$$

Here, u_i, v_i are the columns of the orthogonal matrices U,V. s_i is the singular value. The convolution of the filter $G(x,y)$ with the image $I(x,y)$ can now be approximated using,

$$I(x,y) * G(x,y) \approx I(x,y) * \sum_{i=1}^{K} s_i u_i(y) v_i(x) \tag{13}$$

$$= \sum_{i=1}^{K} s_i I(x,y) * (u_i(y) v_i(x)) \tag{14}$$

$$= \sum_{i=1}^{K} s_i I(x,y) * (u_i(y)\delta(x)) * (v_i(x)\delta(y)) \tag{15}$$

The cost of convolution is now reduced to $O(KN^2M)$. In practice, $K < 4$ captures almost all the structure of the filter and hence offers better efficiency than the complex separable filter.Figure 4 shows the successive approximations achieved over a set of filters.

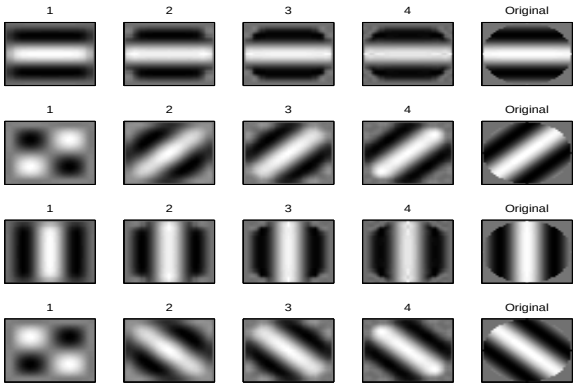

Figure 4: Successive approximations obtained through singular value decomposition

2.3 Box Filters

While the separable and SVD approximations reproduce the gabor filter, in this section we attempt to perform a grosser approximation of the filter to achieve significantly higher efficiency. Currently, the S1 layer in the model, consists of simple line detectors that are modeled using oriented Gabor filters. However, oriented lines can be represented using much simpler represented such as box filters as shown in Figure 5. These filters have very sparse derivatives allowing us to compute convolution quite efficiently. The convolution of the image with the filters $B(x, y)$ can be efficiently computed using the filter's derivatives as follows

$$I(x, y) * B(x, y) = \frac{\partial^2 B(x, y)}{\partial x \partial y} * \int \int I(x, y) \tag{16}$$

The second derivative of the box filter consists of just impulses and therefore needs to be evaluated at only four points for every position in the image. The integral image can also be efficiently computed using single pass algorithm as outlined in [5, 1]. The cost of convolution using this approximation is $O(N^2 c)$, c=4. Unlike other approximations, the computational cost does not depend on the size of the filter at all!

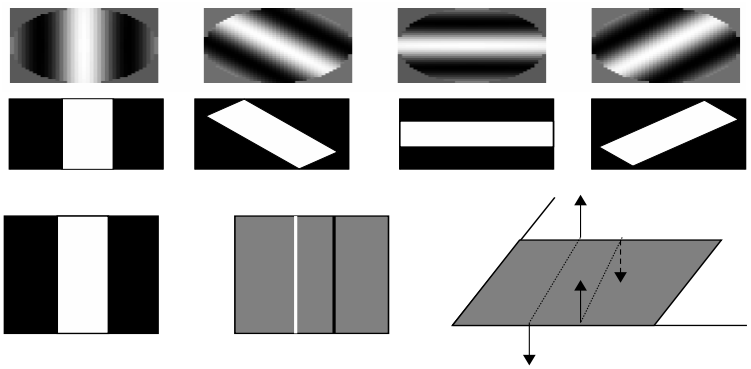

Figure 5: Box filter approximations for the oriented gabor filters and its derivatives

Image size ($H \times W$)	Baseline	Separable	SVD	Box Filter
100×100	1.0900(0.32)	0.7826(0.3172)	0.4482(0.3197)	0.3908(0.3047)
128×128	1.8090(0.4589)	1.5360(0.4568)	0.8445(0.4546)	0.5943(0.4450)
256×256	7.406(1.4424)	8.0517(1.4315)	4.2913(1.4268)	2.3997(1.4403)
512×512	31.2756(5.6481)	54.4857(5.5903)	30.0797(5.5774)	10.4270(5.6270)
640×480	37.840(6.5804)	85.277(6.5426)	47.6650(6.4964)	11.539(6.580)

Table 1: Timing comparisons for different S1 approximations. The C1 timings are almost the same for all cases and are shown in parenthesis.All units are in seconds.

Approximation	Accuracy	Std. Dev
Baseline	39.034%	0.44%
Separable	39.4920%	0.91%
SVD	39.6980%	0.91%
Box Filter	34.7720%	1.15%

Table 2: Comparative results of different S1 approximations. C1 and C2 computations were not changed for these experiments.

2.4 Empirical Evaluation

2.4.1 Timing

In order to compare the speed gain of these approximations, we measured the time taken for S1 an C1 evaluation over several image sizes. All the evaluations were done on a dual core 3 GHz machine under identical conditions. The evaluations was repeated five times. The average results are illustrated in Table 1

2.4.2 Accuracy

The different S1 approximations were evaluated on the Caltech 101 database and compared against the baseline performance. 2000 C2 (500patches $\times$ 4sizes)units and a linear SVM based classifier was used for the evaluation [1]. The S2 and C2 layers were identical in all cases. The evaluations were repeated five times. Table 2 shows the average and variance of the accuracies obtained in these cases.

3 Approximations at the C1 Layer

3.1 Quantization

In this section, we discuss the effects of quantization at the C1 level. There are two primary motivations to study approximation at this level. Firstly, it is very unlikely that neural computations occur at full precision. It is more easier to construct biophysical circuits whose internal representation uses only a few discrete levels. Secondly, discrete (integer only) representation will reduce the computational complexity and makes hardware implementation of the downstream stages feasible.

We computed the distribution of the C1 units (all 8 bands) using 500 'natural images' dataset. The empirical distributions show that outputs of the C1 unit have a heavy tail distribution (See figure 6) making uniform quantization impractical. Instead, we rely upon Max-Lloyd quantization, an optimal scheme that

[1] This is close to the baseline reported in [4]. The performance achieved by the current version of the model utilizing both C1 and C2 features is significantly higher (55%)

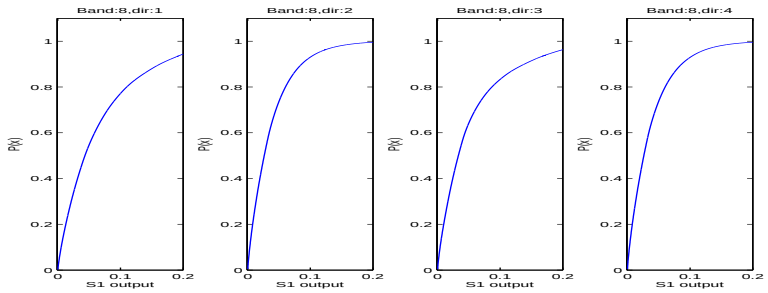

Figure 6: C1 statistics at several scales computed using 500 natural images

---

ensures a minimum variance of the quantization error, given the distribution. Given the probability density function $p(x)$ and a quantization scheme $q(x)$, Max-Lloyd quantization determines M quantization values $y_1, y_2..y_M$ and the corresponding intervals $[a_0, a_1), [a_1, a_2) \ldots [a_{M-1}, a_M]$, that minimizes the error variance given by

$$MSE_q = \sum_{i=1}^{M} \int_{a_{i-1}}^{a_i} |y_i - x|^2 p(x) dx \tag{17}$$

It can be shown that at the optimal MSE value, the quantization level y_i is the centroid of its discretization intervals $[a_{i-1}, a_i]$, and the discretization boundaries is the mid-points of adjacent levels

$$y_i = \frac{\int_{a_{i-1}}^{a_i} x p(x) dx}{\int_{a_{i-1}}^{a_i} p(x) dx} \tag{18}$$

$$a_i = \frac{y_{i-1} + y_i}{2}, i \in [1..M-1] \tag{19}$$

The boundaries a_0 and a_M correspond to the range of x. It can be seen that the solutions for y_i and a_i depend upon each other. There is no closed form solution that allows us to compute these values simultaneously. Therefore, max-Lloyd scheme depends upon an iterative scheme, where the initial values of y_i are obtained by guessing. Subsequently, the values of a_i and y_i are iteratively updated until they converge. We found that 4 level quantization is sufficient to capture the dynamic range of S1 outputs over a set of natural images (Figure 7 shows such an example). It is to be noted that the quantization levels and intervals are determined separable for each *band* and direction.

3.2 Empirical Evaluation

Figure 7 shows the results of 4 level quantization performed on a single face image illustrated earlier. We also compared the effect on accuracy by measuring the object classification performance of the system over the Caltech 101 dataset. It is interesting to observe that the overall classification rate (See Table 3) is slightly better with quantization. This is a reasonable behavior since quantization increases the invariance at the C1 level.

4 Approximations at S2

Each S2 response is computed by obtaining the Euclidean distance between the patch and the C1 outputs at each point. This procedure is repeated for every band of the C1 output and for each patch in the representation dictionary. The complexity of this computation is $O(M^2N^2P)$, where P is the number of

patches and M is the size of each patch. The C2 output is then computed by max pooling over all bands and

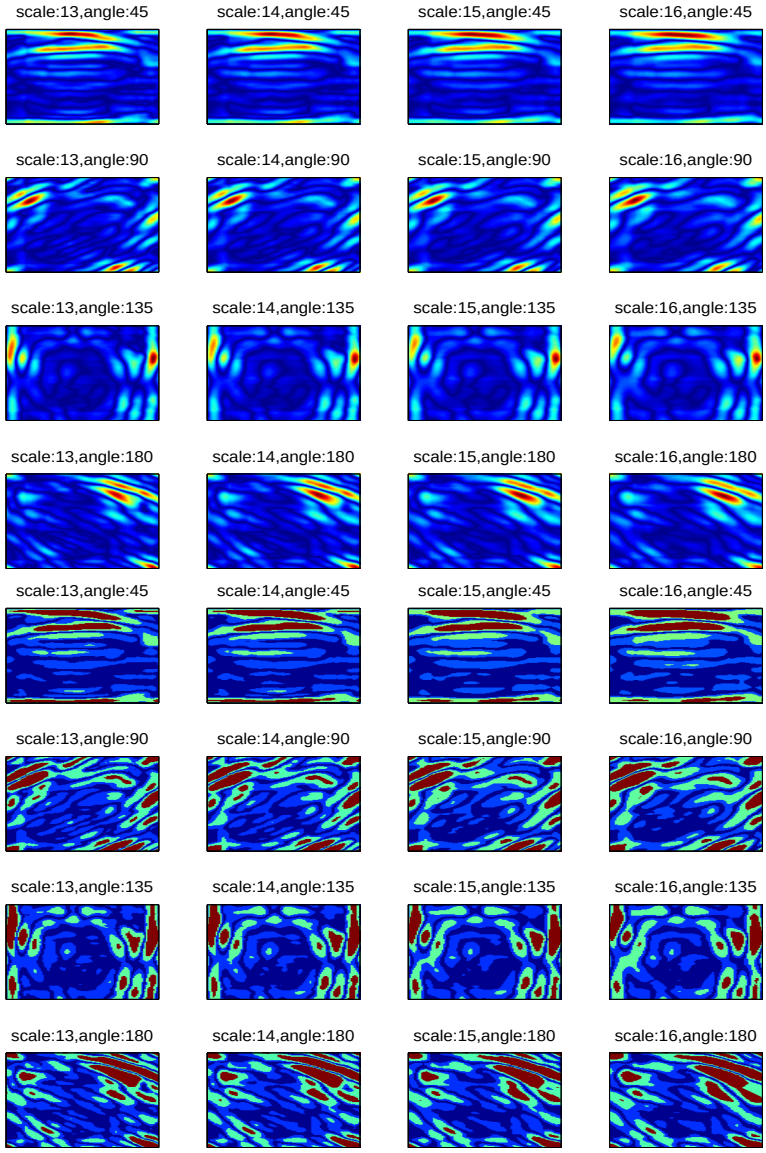

Figure 7: Results of discretizing the S1 units for the face image shown. The top four rows correspond to the baseline system and the bottom four results correspond to the quantized

Approximation	Accuracy	Std. Dev
Baseline	39.034%	0.44%
4 level quantization	40.760%	0.64%

Table 3: Comparative results of different S1 approximations. C1 and C2 computations were not changed for these experiments.

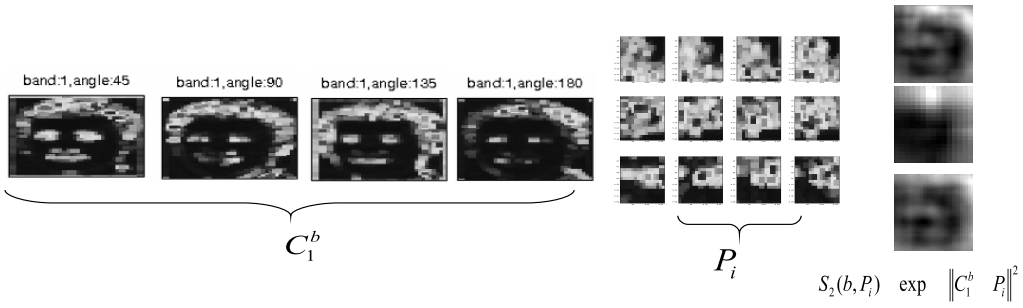

Figure 8: Overview of S2 computation. The S2 output corresponding to prototype patch P_i at band b is computed by evaluating the similarity of the patch at each position in the C1 outputs $C1_b$. The similarity is based on a Gaussian tuning curve.

computation

all positions. In effect, **only one** response is retained after all this computation. Since we are interested in finding only the maximum response, it is interesting to find out if this can be computed without evaluating the response at all positions.

4.1 Subsampling Approximations

One simple approach is to compute the responses at regular grid or a fixed number of random positions. If we subsample by a factor $s \in [0, 1]$, the cost of the computation reduces to $O(s^2N^2M^2P)$. In practice, $s \approx 0.33$, resulting in a factor of 10 reduction in complexity. However, it is to be noted, that by resorting to sub-sampling, we lost the benefit of hardware accelerated implementations that perform distance computation/convolution over the whole image. It is hoped that the order of magnitude reduction in complexity will offset this loss.

4.2 PCA Approximations

There is an S2 unit corresponding to each prototype patch selected during the training process. This representation dictionary of C1 patches is redundant by design and therefore it is likely that the S2 responses will also be redundant. We can utilize this redundancy to reduce the cost of computation. Here we investigate if it is possible to reduce complexity by approximating the patches using a common basis function. The tuning response for any given patch P_i and band b, the response is given by

$$S2_{(b,P_i)}(x,y) = \exp(-\sum_{(u,v)} ||C_b(x-u,y-v) - P_i(u,v)||^2) \tag{20}$$

$$\tag{21}$$

Let us consider the response at one point in the C1 output (x,y is fixed). The response is now written in simpler terms as $\exp(-||C_b - P_i||^2)$. If the patches P_i are expressed in terms of the principal components,

$$P_i \approx \sum_{j=1}^{K} a_j Q_j \tag{22}$$

$$||C_b - P_i||^2 \approx ||C_b||^2 + ||P_i||^2 - 2 * C_b \left(\sum_{j=2}^{K} a_j Q_j \right) \tag{23}$$

$$= \quad ||C_b||^2 + ||P_i||^2 - 2\left(\sum_{j=1}^{K} a_j(Q_j * C_b)\right) \tag{24}$$

All values of $||P_i||^2$ can be computed offline and before hand. Furthermore, all combinations of $Q_j * C_b$ can be computed before hand. The cost of computation is around $O(KMN^2)$. $K < 6$ in practice.

4.3 Empirical Evaluation

4.3.1 Timing

In order to compare the speed gain of these approximations, we measured the time taken for C2 evaluation over several image sizes. All the evaluations were done on a dual core 3 GHz machine under identical conditions. The evaluations were repeated five times. The PCA approximations were computed such that 99.9% of the original variance was retained within the templates. The average results are illustrated in Table 4. The S1 and C1 stages were all identical and is therefore not included in the timings. The gains predicted by approximations are visible only for larger size images, since hardware accelerated implementations offer sufficient efficiency at smaller image sizes. In some cases, the hardware accelerated versions actually perform better than the approximations for some sizes (e.g.. sub-sampling approximation at larger sizes). Therefore, care has to be taken to switch these approximations where they have the most impact.

Image Size	Baseline	Subsampling	PCA
100×100	7.6586	3.1336	8.3878
128×128	11.2174	5.9716	9.0476
256×256	38.3146	33.5820	14.028
512×512	153.005	163.9780	40.738
480×640	180.282	191.4970	54.109

Table 4: Timing comparisons for different S2 approximations. The S1,C1 timings are identical for all cases and are not included here.

4.3.2 Accuracy

We measure the trade-off in accuracy by measuring the performance of these approximate representations over the Caltech 101 database. The S1 and C1 layers were identical (baseline) in both cases.

Approximation	Accuracy
Baseline	39.03
Sub sampling	38.28*
PCA	35.89

Table 5: Accuracy trade-offs obtained by replacing S2 with approximations.

5 Approximations at the C2 Level

In this section, we discuss several approximations at the C2 description level. The approximations predominantly consists of scaling and quantization. The existing model relies upon continuous outputs for classification, a situation that is hard to realize using biophysically plausible circuits. It is interesting to find out as to how many levels of representation is sufficient for performing object recognition.

Approximation	Accuracy	Std. Dev
Baseline	39.034	0.44
Log Baseline [2]	40.8700	0.36
Binarization	31.428	0.98
Log scaling and Binarization	32.480	0.56
4 Level Quantization	37.3920	0.63
8 Level Quantization	38.5240	0.49
C1 and C2 Quantization [3]	32.4540	0.56

Table 6: Comparative results of different approximations. All numbers are in percentages

5.1 Binarization

Here we quantize the C2 outputs into two levels based on statistics collected over 500 natural images. For each C2 unit, we observe that the output distribution follows a **log-normal** distribution. We determined the parameters of this distribution, and discretized the C2 unit outputs into two levels ± 1 based on whether if its above or below the mean.

5.2 Multi-level Quantization

Since optimal quantization schemes like Max Lloyd work with bounded range variables, we cannot use it to optimally quantize C2 units. Instead we rely upon our previous observation that the C2 unit outputs are distributed in a log-normal fashion. Multi-level quantization is then carried out by quantizing the output at equal probability intervals.

5.3 Empirical Evaluation

The different quantization schemes were evaluated by performing classification on the Caltech 101 dataset. 2000 C2 units and a linear SVM were used for the evaluation. Identical splits (15 training and 50 testing examples) were used in all the experiments. The S2 universal patch dictionary was built by randomly selection C1 patches from 500 natural images. The evaluation was repeated 5 times. Table **??** shows the mean and std. deviation of these results.

5.4 Quantization vs. Representation

It is seen that quantizing the C2 output leads to considerable (around 7%) drop in accuracy. However, quantizing to 4 levels or 8 levels restores the classification rate to nearly baseline levels. It is also observed that the performance can be increased (not indefinitely) by increasing the number of C2 units. We are interested in finding out if the number of C2 units can be traded off with output precision. We conjecture that the performance is strongly correlated with the number of bits (levels at C2 units $\times$ number of C2 units) and hence information content. We measured the performance of the system using several discretization levels and number of C2 units. The results are shown in Table 7.

6 Conclusion

The HMAX model is a biologically inspired hierarchical model for object recognition. Due to its faithfulness to biology, it is computationally bound and is not very practical from a deployable computer vision perspective. In this report, we attempted to increase the efficiency of the model by performing approximation at several levels. We reported accuracy vs. speed trade-off results evaluated over the Caltech 101 database. Further, we discussed the implications of quantization at the various stages and in particular studied the

Patches per size	Baseline	2 Levels	4 Levels	8 Levels
500	39.034(0.44)	31.428(0.98)	37.39(0.63)	38.524(0.49)
1000	41.730(0.68)	35.21(0.63)	39.99(0.34)	40.84(0.36)
2000	42.46(0.32)	37.02(0.46)	41.37(0.35)	42.32(0.27)

Table 7: Trade off between the precision of representation at the C2 level and number of patches at the C1 level

trade-off between the number of features used for classification vs. the number of quantization levels used for the features.

References

[1] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. *Pattern Recognition*, pages 297–304, 2003.

[2] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical Report AIM-2005-36, Computer Science and Artificial Intelligence Laboratory, MIT, 2005.

[3] T. Serre, Wolf L., S. Bileschi, M. Reisenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[4] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition*, volume 2, pages 994–1000, 2006.

[5] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.