Supplementary Methods

Randomization and Selection of Self-Calibrating Binary Encoders in Neural Systems

Alan Herbert

alan.herbert@insideoutbio.com

1. 617.584.0360

Abstract

Zebrafish larvae are able to track and capture prey a few days after birth, consistent with a genetic specification of pattern recognition by the nervous system (PRNS). Computational approaches to model PRNS use neural network designs that rely on multiple fully connected layers between inputs and outputs. Whereas the biology suggests that PRNS can be specified genetically with a small subset of genes, the computational approach requires thousands to millions of parameters to achieve high accuracy. For example, the convolutional neural network (CNN) LeNet5 [1] uses 6 neural layers with 60,000 trainable parameters and multiple rounds of tuning to accurately classify handwritten digits, while competing designs employ over 12,000,000 [2,3] variables. We propose a more biologically plausible approach using what we call self-calibrating randomly initiated binary encoders (SCRIBE). Self-calibrated neural nets (SCN) based on SCRIBEs require as few as 300 parameters and only a single round of selection to achieve accuracies similar to CNNs. Unlike CNNs, SCNs can easily be recalibrated to perform multiple recognition tasks. SCNs could be specified biologically with just a few genes.

We provide a description of a process based on set of randomly generated self-calibrating neural ensembles (SCN) that can achieve highly accurate PRNS (Fig. 1). SCN are population based and are not concerned with adjusting the strength of individual neural connections within a population – that is why convolutional neural networks (CNNs) require optimization of many parameters in their design, often rendering them task specific. Both SCNs and CNNs require a classifier to optimize accuracy of class assignment by assigning a positive, negative or zero weight to each the output from their respective nets. SCN classification is based on the binary output of calibrated arrays that are pooled and weighted according to whether the output is inhibitory (assigned a negative sign) or stimulatory (assigned a positive sign), The matrix of outputs is further reduced by L1 normalization to identify and weigh output that produce accurate classification of inputs.

SCNs have a flexible design. Input is preprocessed, locally convoluted, smoothed, sampled and then pooled prior to classification of input (Fig. 1). They differ from other approaches to PRNS in that the algorithms used at each step can be selected independently of those employed in other steps, generating a range of potential architectures for adaptation by organisms to a particular environment. In the examples used here, pre-processing of hand written digits (Fig. SF1a) is restricted to image rotation. This design feature is based on the mammalian visual cortex where elements respond preferentially to image orientation and is aimed at conferring rotational invariance on the classifier. In the next step, local convolution is by a subset of kernels, termed a tranche, selected from the randomly generated population specified by the RAN(2,-1) array (Fig. SF1b). The RAN(2,-1) kernels are of low complexity with two discrete elements (2 and -1), representing the binary output of SCRIBES (as described in the manuscript), weighted by whether outputs are inhibitory or stimulatory. The approach is consistent with the observed ratio of excitatory to inhibitory synapses in the CA1 area of the rat hippocampus[4]. In contrast, CNN kernels are drawn from a continuous random distribution[1]. The output from convolution with each RAN(2,-1) kernel is divided into patches and the pixels within each patch are pooled to score the signal to noise ratio (SNR) over each input image (Fig. SF2). As with CNN, patch pooling confers translational invariance on the classifier[1].

The classifier is calculated using multinomial models to predict class membership. The models use an objective function regularized with either a lasso (LR) or a group lasso (gLR) as implemented in the GLMnet software package to assign a weight to each input from the net [5]. GLMnet also allows for cross-validation of models. Lasso regularization minimizes the number of parameters (i.e. the number of non-zero coefficients) in the final model. GLMnet employs a cyclic coordinate descent algorithm to improve speed. The results obtained depend on the particular regularizer used, with a trade-off between the higher classifier accuracy obtained with the

gLR and the smaller number of parameters with LR (Fig. SF4) where the weights of classifiers from each approach are plotted for each column of the randomly generated and selected kernel in the tranches used in the analyses.

The combination of a tranche, an image patch vector and a stack array is referred to here as a net (boxed in Fig. 1). SCNs consist of a net connected to a single classifier component (Fig. 1). The output of a net may be sampled using multiple classifiers with each combination of net/classifier defining a unique SCN. In the work described here, training of each SCN is performed with a labeled set of images. The accuracy of each SCN classifier is then easily scored using a test set of data.

We analyzed the performance of SCNs and compared the results with previous work[1], as summarized at http://yann.lecun.com/exdb/mnist/. SCNs were tested using either the MNIST or USPS hand written digit test and training sets (Fig. SF1a). The MNIST 28 x 28 pixel handwritten dataset includes 60,000 training images and 10,000 test images. The best neural network learning machines requires up to 12 million weights [6] and have an error rate of ~0.23% (range 0.23-4.7%) [1,3], close to the 0.2% rate estimated for practiced humans. The USPS images are notoriously difficult to recognize and consist of a set of 7290 16 x 16 pixel images for training and 2007 for testing. The current benchmark for neural network solutions equal the human error rate of ~2.5% [7,8].

We assessed the performance of SCN. A series of random SCN, sampled from a set of 64 (3 x 3) kernels was generated. Each kernel was then individually tested to assess its accuracy as a stand-alone classifier. The kernel was fed images that had been rotated through 0^0,30^0,60^0 and 90^0 (r(0,90,30) of the form r(start, finish, increment value)). The mean classification accuracy with LR of each kernel tested separately for the MNIST dataset was 96.94% (range 92.40%-98.43%, SD=1.30%). The performance is comparable to some of the non-neural network approaches previously described (http:// http://yann.lecun.com/exdb/mnist/). The mean classification accuracy with LR for the USPS datasets was 93.92% (range 90.23%-95.21%,SD=1.04%), reflecting the greater degree of difficulty in classification of images in this collection.

We then assessed SCNs made by combing individual kernels into tranches, varying the tranche size from 12 to 30 kernels. The classification accuracy of each tranche was assessed using the MNIST dataset. An accuracy of greater than 99% was observed with a single iteration under nearly all conditions tested (see Range-Test Table 1), with little difference between tranches with 10 kernels compared to those with 30. The use of multiple kernels in each SCN increases accuracy, even when randomly constructed and randomly selected.

We also tested other strategies to improve SCN classification accuracy. An ensemble approach based on SCNs trained independently of each other (see methods) resulted in only small gains in accuracy (Table 1, Column 11). Further small improvements in accuracy (<0.1%) were obtained when the number of rotations used in image pre-processing was increased or when gLR was used instead of LR (Table 1). The minimal gain is offset by the increase in the computational burden required by such schemes. When cross-validation was used for training the classifier, the number of non-zero coefficients in the final model was lowest (Table 1, Groups 2 and 6), but was higher when gLR rather than LR was used. Overall, these results demonstrate that highly accurate SCNs can be made using low complexity kernels organized in tranches to analyze the MNIST dataset, with only small improvements gained by increasing the complexity involved in the design. They also show that accuracy is high when only a single round of training is involved. The total number of parameters that specify such SCNs is much lower than reported for other proposed solutions for the accurate, efficient classification of handwritten digits by at least two orders of magnitude.

We then evaluated performance of SCNs on the USPS dataset. Increasing the number of pre-rotations improved accuracy overall with RAN(2,-1) kernels and was optimal for rotations spaced at either 30^0 or 20^0, but fell off at 10^0 when the number of columns in the stack array was greater than the number of samples in the training set (Table 2, Fig. 2b). With no rotation, the accuracy obtained was 95.91% (Table 2, lines 1 and 2). Overall, accuracy was increased by 0.5% with gLR compared to LR, while the number of non-zero coefficients increased up to 5-fold, from 650 to 2800 (Table 2, groups 5 and 6). As with the MNIST dataset (Table 1), the number of non-zero coefficients was greatly reduced when cross-validation was used in training (Table 2, line 10). Jointly the MNIST and USPS datasets illustrate that the trade-off for SCNs between computational cost and accuracy depends on the dataset and its size.

To assess the effect of training set size, we varied the number of images used for calibration of the classifier by randomly sampling of the standard 60,000 digit MNIST training set (Fig. 2a). With as few as 3,000 images, accuracy was 98.58% (sem=0.03%, r(0,300,60), t=10 , gLR), increasing almost linearly as more images were added (Fig 2a). The lower bound was limited by the GLMnet gLR algorithm which requires that the array stack column number be smaller than the total number of images in the training set. When the analysis was performed using LR, accuracy was diminished (Fig. 2a). Even with this limitation, it was possible to achieve an accuracy of 97.49% (sem=0.02%) using only 1000 images. These results illustrate that SCNs can produce highly accurate results after calibration with only a small set of training images. As the number of images increases, so does the accuracy.

With both the MNIST and USPS datasets, accuracy was maintained when only the best 3 SCNs were scored (Table 1, Column 11). This result suggests that a PRNS based on SCNs could evolve by using additional rounds of selection based on the best performing

SCNS in the previous round. By choosing optimal SCNs and minimizing the number required for a particular task, the efficiency of PRNS would improve over time by reducing the computational burden.

We tested other classes of random kernels to test whether would improve the accuracy of classification using the USPS dataset. First, omitting convolution reduced accuracy to 91.73%, even with pre-rotation of the images (r=r(0,180,30),k=12,s=12,N=5). Second, convolution sets based on kernel selected from RAN(1,-1) and RAN(1,1) arrays produced accuracies of 94.07% and 89.38% (r=r(0,180,30),k=12,s=12,N=5). Third, alternatives to random kernels for convolution selected from discrete cosine transformations (DCT) or Hadamard-Walsh (HW) transform arrays were examined. For these experiments, images were pre-processed using a series of rotation sets of different size, covering the range from 0^0 to 330^0 (Fig. 2b). Each new rotation set was generated by adding another 30^0 rotation step to that found in the preceding set. The maximum accuracy obtained for the USPS dataset using a single DCT(2,4) tranche of kernel size 14 was 97.78% (r(0,180,30), Figs. 2b, 2c). Kernels from a DCT(2,3) array yielded an accuracy of 97.45% with gLR, similar to the best results obtained with RAN(2,-1) kernels (Fig. 2b), even though the DCT(2,3) array consists of only 9 elements (Fig. SF5). The best individual result was obtained for a DCT(2,4) matrix with an accuracy of 97.81% (gLR) when the tranche contained 12 kernels. Each kernel class produced a different accuracy profile when tested with images pre-processed with different rotation sets (Fig 2b). The DCT(2,4) kernels had a defined maximum at 180^0, yielding an accuracy of 97.69% (sem=0.03%, t=12). Hadamard-Walsh kernels of size 4 and size 8 yielded accuracies of 96.86% and 97.21% with tranche sizes of 4 and 8 kernels respectively and gLR. SCNs design thus allows the use of different kernel arrays to optimize classification accuracy under different conditions.

Interesting properties of SCNs emerged when we introduced filters into the net to reduce computational burden. We tested a filter deigned to sample the output after convolution but prior to patch pooling (Fig. 2d). Classification accuracy was preserved when only 10 (62.5%) pixels per row of 16 were sampled and diminished only when a few elements per row were sampled (for example, an s=5 filter had accuracy of 98.92%). Unlike CNNs, a fully connected system is not an essential prerequisite for SCNs. Different kernel/filter combinations were found to produce the same patches. This outcome allows a strategy for the unsupervised classification of images based on clustering those kernel/filter combinations that produce identical patches. The strategy is not used here.

We also tested sampling by filters applied after patch pooling, using the USPS dataset. When sampling of columns from the stack array was random, accuracy of classification was reduced. To avoid this problem, we used the scale-free robust Rousseeuw–Croux estimator of dispersion Qn to order columns by variability without regard to the kernel that produced them [9]. This approach allowed us to test two different sampling schemes. The first scheme was motivated by the reduction in overall computational cost possible with a

divide and conquers strategy. We broke the sorted stack array into equally sized groups of columns from left to right and analyzed each in parallel using separate classifiers for each group. We then combined the output from each classifier to produce the final result. This design decreased the total number of calculations, which scales with an upper bound of $\mathbf{c^2}$ (**c**=number of columns from the stack array used by the classifier for **c**<**n**, otherwise it scales with $\mathbf{c^3}$ [10]). Dividing the columns into 3, 4 or 5 groups (i.e. reducing computations by 3/9, 4/16 and 5/25) yielded accuracies of 97.51%, 97.41% and 97.35% for the USPS dataset respectively. These schemes allow reduced computational coast with a slight decrease in accuracy.

In the second approach, we sampled the sorted columns using a power of 2 series to select the columns used for classification (e.g. half of the columns, a quarter, an eight…). The design was motivated by the variation observed in the dendritic trees of neurons that double in size at each node, augmenting the number of branches receiving input. When every 256th column across the variance distribution (i.e. 22 columns out of 5616) was selected, we obtained an accuracy of 93.12% using the parameters given in Fig. 3. When the interval sampled from the stack array was halved, accuracy increased linearly, reaching a maximum value of 97.65% when all columns were included (Fig. 3). The same linearity obtained when accuracy is plotted against sampling interval (i.e. versus $1/2^n$) was also present when the pooling size was increased with 4 x 4 patch pooling (blue line Fig. 3) or when the image was pre-processed by rotation through a different set of angles (orange line Fig. 3) or when the MNIST dataset was tested (magenta line Fig. 3). Line fitting to each result gave an estimate of the maximum accuracy of a SCN design and allows each SCN design/dataset combination to be characterized by two parameters: intercept and gradient (Fig. 3). While classifier accuracy is a linear function of the information sampled, the computational cost in these examples increases non-linearly with $\mathbf{c^2}$. The trade-off is between accuracy and speed.

SCNs that sample the same stack array at different densities offer interesting design possibilities. Denser sampling modes are more accurate than sparser ones but require greater computational time, regardless of whether samples are taken over intervals of space or measures of time. The faster, less accurate mode allows rapid assessment of input. This creates a prior set of probabilities that can be updated by the slower, more accurate mode to produce more reliable estimates, as described by Bayes Theorem. Such an arrangement facilitates responses to features that vary rapidly over time, such as harmony or motion, with the faster mode anticipating the outcome of the slower analysis. The generation of multiple priors through a range of different sampling intervals affords the slower mode a series of perspectives for improving the classification of objects. With SCNs, new sampling modes evolve efficiently and independently by optimizing the current sampling scheme through selection of tranches and also by finding novel ways to connect the stack array to different classifiers. Sampling by multiple SCNs of the same information stream has no equivalent in existing CNN implementations. SCNs also open up the possibility of using swarm classifiers that avoid local energy minima and are time efficient in evaluating alternative fits. [11]

SCNs have similarity to both Importance Sampled Learning Ensembles (ISLE) [12] and Compressible Sensing algorithms (CSA) [13]. ISLE use sampled data or a subset of attributes to train ensemble predictors. With SCNs, all randomly selected tranches are used for building the classifier. Sampling is used to reduce the number of computations while still maintaining accuracy. SCNs and CSA both aim to produce sparse representations to increase efficiency. CSA necessitates a space where the representation of data is reduced while preserving the distance relationships between pairs of points that allows reconstruction of the original image[14]. With SCNs translational and rotational invariance of images is achieved at the expense of losing those features necessary to remake an individual image. However, SCN output does enable feed-forward to sensory layers to enable their input to reconstruct features of the image classified, a concept captured by a scheme based on Markov blanket. [15] With SCNs there is no weighting of individual connections within the net or any need for fully connected elements. Highly accurate PNRS is achieved through selection of tranches from low complexity neuronal ensembles and calibration of the classifier requires only a small number of parameters.

SCNs are scalable. The scheme used here is based on a small set 3 x3 separable matrices produced from arrays with 2 random elements that model the weighted output of the SCRIBEs described in the manuscript. The design reduces computation by requiring 6 steps per convolution (i.e. 2xk, k=dimension of the kernel) rather than 9 (k^2) as with other designs. The kernel set is small and can be generated in its entirety so that the whole space can be explored in selection of the optimal tranches for a classification task (Fig. SF1b). These compact kernels with LR are faster than other solutions and can be used with cross-validation to produce models that contain only a few non-zero coefficients. Larger kernel sets made from separable matrices with more than 2 elements or with more dimensions or from a different type of finite group may be more useful for other problems. Even then, a subspace, such as one that contains orthogonal subsets, could be extensively explored to select the best set of tranches for a particular classification task. The dynamic nature of array selection means that the initial set of tranches produced need not be optimal, nor the only solution possible.

Biologically, SCNs only need to function well enough until more successful outcomes are generated. A diversity of solutions is possible because elements of SCN can change independently of the other processing steps. For example, the sensors involved in pre-processing, the type of convolution kernel selected and the class of classification algorithm can vary by classification task. Each SCN component can be encoded by a small set of genes. For example, the architecture of SCRIBEs could be specified in similar ways to which neural innovation of tissues occurs, with connection of outputs to higher order arrays similarly specified. Genes similar to those that specify tiling of dendritic arbors could specify classifier connectivity: more branching, denser sampling. Random variation in wiring can also affect how inputs are filtered and processed and allow detection by SCRIBEs of feature attributes as id described in the manuscript. Excitatory and inhibitory neurons associated with receptive fields then account for synaptic weights such as the +2 and -1 used here. Classifiers may depend on neurotransmitters produced by the neurons with which SCRIBE outputs synapse.

Additional development effects also can play a role. For example, the GABA receptor reportedly can switch from excitatory to inhibitory during development and thus are capable of playing a role in classification by SCN[16]. Due to their simple modular design, SCN balance speed of processing with accuracy of classification. The SCRIBE architecture on which they are based can be genetically encoded using those interactions already known to developmental biologists.

Methods

Each image was pre-processed by rotation through a set of angles to produce the input image set using R software environment for statistical computing and graphic (http://www.r-project.org/) and the BiOps package (V 0.2.2). The rotation is selectable and 10^0, 20^0 , 30^0 ,40^0 or 60^0 steps were used in various tests. For local convolution, we used small 3 x 3 separable kernel constructed from discrete elements (rather than a continuous random variable as used in CNN). The kernels are the inner product of two 1x3 vectors that represent the left and right half of a 1x6 vector randomly generated from the set (2,-1) (Fig. SF1b). There are 64 different possible 1x6 vectors available using this scheme ($64=2^3x2^3$). Each kernel is normalized by the absolute sum of the matrix elements. A randomly selecting a subset of kernels is then selected to form a tranche, with each kernel used for local convolution of the input image set. In some cases, each kernel was paired with a unique, randomly generated filter. The filter was applied after convolution but prior to patch pooling. The filter allowed selection of a specified number of elements from each convoluted row for further processing. Otherwise, all elements of the output from convolution were used. The kernel/filter pair was applied to all members of the training and testing input image sets. To create translation invariance, outputs were then down sampled to produce a reduced representation of the image. For example, 6 x 6 overlapping regions were pooled using a neighborhood function to calculate a signal to noise ratio (SNR). The SNR function normalized the sum of squares of the selected pixels by the mean sum of squares for the entire image to produce a 6 x 6 output matrix for each random kernel/filter pair. Each matrix was reshaped into a 1 x 36 vector that was concatenated with those from other kernels in the tranche to produce an output vector, one for each image rotation (length= 36 x number of random kernels used). Subsequently, the vector was concatenated with those generated using the other image rotations (length= 36 x number of random kernels x number of rotations)). The concatenate for each image was stacked on an array with each row corresponding to a separate image and with the output from each kernel vertically aligned with that of other images. These stacks lack any of the horizontal connectivity present in CNN. The columns represent the variables used by the GLMnet software package to build a classifier using either a lasso (LR) of group lasso (gLR) for regularization, both based on L1 minimization [5]. gLR ensures that the same coefficients are used as predictors for all classes by minimizing the L2 column norm of the entire stack while LR minimizes a different subset of coefficients for each class. The GLMnet solver is based on a cyclic coordinate descent algorithm that optimizes weights one column at a time by treating all other pixel column stacks as fixed variables with fixed weights. A column weight is set to zero when information

from that stack fails to reduce classification error. This process continues in a reiterative fashion across all stacks until convergence occurs. Where indicated, 10-fold cross validation was performed with the training set. The GLMnet term α was set to 1 except where noted. The performance of each classifier was checked on the test set of images. Figures were produced using the Fields package [17]. The program is implemented in R.

References

1 LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**, 2278-2324, doi:10.1109/5.726791 (1998).

2 Ciresan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Computation* **22**, 3207-3220, doi:Doi 10.1162/Neco_a_00052 (2010).

3 Ciresan, D. C., Meier, U. & Schmidhuber, J. in *2012 IEEE Conference on Computer Vision and Pattern Recognition.* 3642–3649.

4 Gulyas, A. I., Megias, M., Emri, Z. & Freund, T. F. Total number and ratio of excitatory and inhibitory synapses converging onto single interneurons of different types in the CA1 area of the rat hippocampus. *J Neurosci* **19**, 10082-10097 (1999).

5 Friedman, J., Hastie, T. & Tibshirani, R. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* **33**, 1-22 (2010).

6 LeCun, Y. *et al.* in *INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS.* 53-60.

7 Seewald, A. K. On the Brittleness of Handwritten Digit Recognition Models. *ISRN Machine Vision* **2012**, 10, doi:10.5402/2012/834127 (2012).

8 Simard, P. Y., LeCun, Y. & Denker, J. S. in *Advances in Neural Information Processing Systems 5, [NIPS Conference]* 50-58 (Morgan Kaufmann Publishers Inc., 1993).

9 Rousseeuw, P. J. & Croux, C. Alternatives to the Median Absolute Deviation. *J Am Stat Assoc* **88**, 1273-1283, doi:Doi 10.2307/2291267 (1993).

10 Raz, R. On the complexity of matrix product. 144, doi:10.1145/509907.509932 (2002).

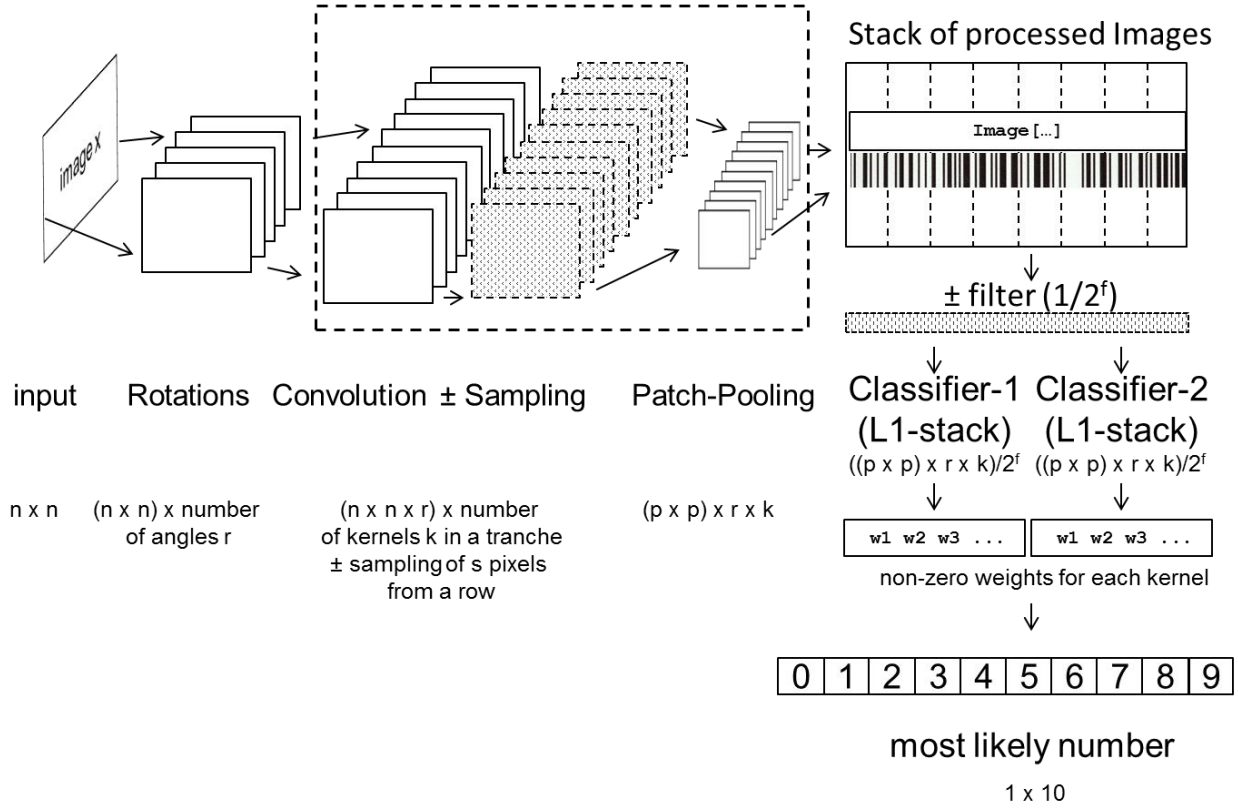
11 Chakraborty, A. & Kar, A. K. in *Nature-Inspired Computing and Optimization Modeling and Optimization in Science and Technologies* Ch. Chapter 19, 475-494 (2017).

12 Friedman, J. H. & Popescu, B. E. Importance Sampled Learning Ensembles. 1-32 (Department of Statistics, Stanford University, 2003).

13 Tao, T. & Candes, E. Near Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? *IEEE Transactions on Information Theory* **52**, 5406–5425 (2006).

14 Gao, J. B., Shi, Q. F. & Caetano, T. S. Dimensionality reduction via compressive sensing. *Pattern Recogn Lett* **33**, 1163-1170, doi:DOI 10.1016/j.patrec.2012.02.007 (2012).
15 Friston, K. Life as we know it. *J R Soc Interface* **10**, 20130475, doi:10.1098/rsif.2013.0475 (2013).
16 Ganguly, K., Schinder, A. F., Wong, S. T. & Poo, M. GABA itself promotes the developmental switch of neuronal GABAergic responses from excitation to inhibition. *Cell* **105**, 521-532 (2001).
17 Fields Development Team. *fields: Tools for Spatial Data*, <http://www.cgd.ucar.edu/Software/Fields> (2006).

Figure 1: Dual SCNs

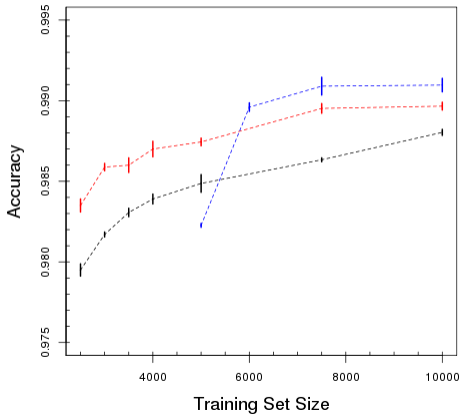


The flow of information from image to a single classifier involves one step of pre-processing that involves rotation of an image (total number =n) through a number of different angles (number of rotations =r). The transforms are then convoluted with a tranche of kernels (number=k). Different types of kernels can be used, including random kernels composed of [2,-1] elements that represent the pooled inhibitory or stimulatory binary output from SCRIBEs, or those that performed Direct Cosine Transforms (DCT), or Hadamard transforms. The output from each convolution is then divided into a patchwork and each patch p is pooled to provide a measurement of the signal to noise ratio in that neighborhood. The result is linearized and stacked into an array (shown as a barcode in the figure)

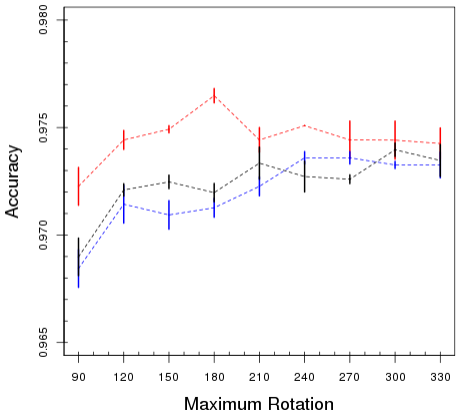
with each row representing a single image after processing and its class label. Up until this stage, all steps are performed in parallel with no transfer of information between each of the processes. A classifier builds a prediction model for each class based on column values. Here two independently built classifiers are shown that sample the stack array, potentially optimized by different algorithms. The lightly stippled areas represent optional filters designed to reduce the computational burden. The s-filter samples the result of convolution. The f-filter samples the output from the stack array so that $1/2^f$ columns are used in model building by the classifier. The dotted box contains the elements that are reported by others in characterizing their neural network components and are referred to here as a net. Image pre-processing machinery such as SCRIBES are outside this box, as is the classifier machinery. The combination of an input processing layer, a net and a single classifier is referred to as a SCN. Two SCNs are shown, one that uses classifier-1 and the other classifier-2.

Figure 2

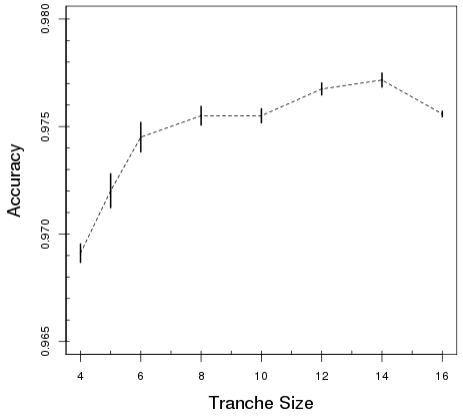
A.



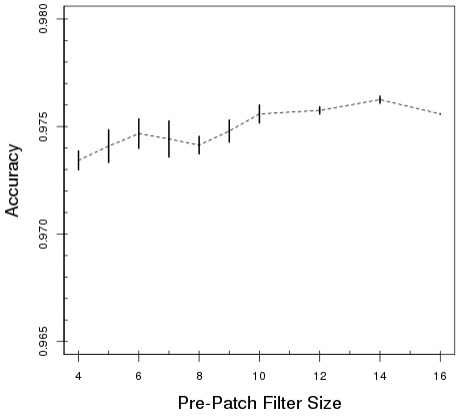
B.



C.



D.



a. Training size and accuracy: The MNIST dataset was sampled and used to calibrate a multinomial classifier. Local convolution was performed with tranches of DCT(2,4) kernels with parameters r= r(0,300,60), t=10, s=20, n=5 (R=pre-rotation of images, T=kernels in tranche, S= sample size per row, pre-patch pooling, n=number of nets. The red and black lines compare models regularized with either gLR (red) or LR (black)). The blue line represents results from DCT(2,4) kernels with r= r(0,320,40), t=16, s= 20 with gLR. In this case, the gLR accuracy decreases when the number of columns in the stack array is equal to or greater than the number of training samples.
b. Pre-rotation of images tested against different kernels: Images from the USPS set were pre-rotated before local convolution with DCT(2,4) kernels (red line, t=16,s=12), RAN(2,-1) kernels (black line, t=12 , s=12) or DCT(2,3) kernels (blue line ,t=9, s=9). Each rotation set started at 0^0, and increased by 30^0 to the maximum value shown on the x-axis. The results are for a minimum of n=3 replicates against the test set (standard errors of the mean are shown).
c. Tranche size and accuracy. Results from the USPS dataset for DCT(2,4) kernels, r=r(0,180,30) , s=16, n=3 for tranches containing varying numbers of kernels.
d. Pre-patch sample size and accuracy. Results from the USPS dataset for DCT(2,4) kernels, r=r(0,180,30) , t= 16 kernels, n=3. The x-axis lists the number of elements s per row that were sampled subsequent to convolution but prior to patch pooling. Accuracy is shown on the y-axis.

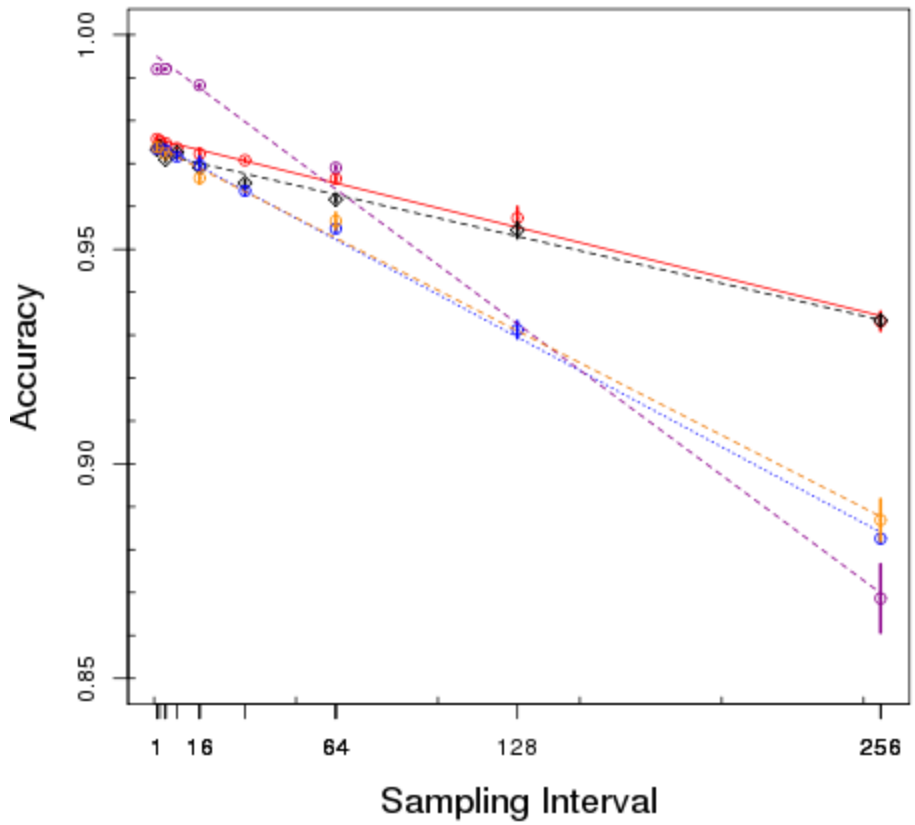


Figure 3: Accuracy versus sparsity for the USPS and MNIST datasets. Output from the stack array from both the training and test sets was sampled at the shown intervals before being passed to the classifier. The magenta line is obtained with the MNIST dataset (DCT(2,4), t=5, k=12, r=r(0,320,40), patch size p=6x6). The other lines use the USPS dataset. The dashed black line is for the RAN(2,-1),t=11, k=12, r=r(0,360,30), p=6x6. For comparison, the solid red line displays results with DCT(2,4) kernels, leaving other

parameters unchanged (i.e. t=11,k=12, r=r(0,360,30), p= 6 x 6). The dotted blue line is for DCT(2,4) kernels with a smaller patch size (P=4x4), with other parameters unchanged (i.e. t=11,k=12,r= r(0,360,30)). The dotted orange line is for smaller DCT(2,3) kernels with t=11, k=12, r(0,300,60), p= 6 x 6. The lines were fitted using a linear model. The intercepts for the lines top to bottom are 0.9954, 0.9757, 0.9750, 0.9473 and 0.9725. Errors bars represent the standard error of the mean determined from 3 trials. The upwards ticks on the x-axis are spaced at intervals of 50.

Table 1: Tranche size and analysis of the MNIST dataset

Group	SCNs	Type	k	Pre-R	gLR	α	s	CV	Mean-Test	Range-Test	Mean-NZ	Range-NZ	Stack	Mode-T	Top 3
1	5	RAN(2,-1)	10	r(0,320,40)	no	0.75	13	no	0.9908	0.9904-0.9912	**1951**	1916-1986	3240	0.9921	0.9919
2	5	RAN(2,-1)	10	r(0,320,40)	no	0.75	13	yes	0.9908	0.9896-0.9918	**324**	324-361	3240	0.9919	0.9916
3	5	RAN(2,-1)	10	r(0,320,40)	no	1.00	13	no	0.9904	0.9898-0.9907	**1720**	1484-1802	3240	0.9910	0.9914
4	5	RAN(2,-1)	10	r(0,320,40)	no	1.00	13	yes	0.9900	0.9885-0.9910	**331**	307-352	3240	0.9916	0.9918
5	5	RAN(2,-1)	20	r(0,180,30)	no	1.00	20	no	0.9913	0.9910-0.9917	**2089**	1932-2234	5040	0.9924	0.9922
6	5	RAN(2,-1)	20	r(0,180,30)	yes	1.00	20	no	0.9924	0.9915-0.9932	**4966**	4922-4999	5040	0.9928	0.9932
7	7	RAN(2,-1)	10	r(0,90,30)	no	1.00	20	yes	0.9903	0.9893-0.9919	**304**	292-310	1440	0.9917	0.9921
8	7	RAN(2,-1)	30	r(0,90,30)	yes	0.75	13	no	0.9917	0.9912-0.9925	**4268**	4212-4295	4320	0.9925	0.9928
9	7	RAN(2,-1)	12	r(0,360,30)	yes	0.75	13	no	0.9918	0.9910-0.9929	**5539**	5268-5596	5616	0.9930	0.9925
10	7	RAN(2,-1)	20	r(0,360,30)	yes	0.75	13	no	0.9926	0.9921-0.9934	**8745**	8171-9273	9360	0.9933	0.9936
11	5	DCT(2,4)	16	r(0,180,30)	no	1.00	20	no	0.9909	0.9901-0.9915	**2034**	1969-2099	4032	0.9912	0.9920
12	5	DCT(2,4)	16	r(0,180,30)	yes	1.00	20	no	0.9919	0.9916-0.9921	**4015**	4010-4024	4032	0.9921	0.9921
13	5	DCT(2,3)	9	r(0,180,30)	no	1.00	20	no	0.9910	0.9908-0.9915	**1583**	1568-1593	2268	0.9919	0.9914
14	5	DCT(2,3)	9	r(0,180,30)	yes	1.00	20	no	0.9917	0.9913-0.9919	**2262**	2255-2267	2268	0.9921	0.9923

SCN were made using different kernel sets. The number of SCN used to generate each result is given in column 2 and the type of kernel in column labeled “k”. The number of kernels k per tranche was varied, as was the pre-rotation (Pre-R) of the image (shown as r=r(start, finish, size of steps in between) in column 4). The number of elements prior to patch pooling is shown in column “s” and whether cross-validation was used during training is indicated in column “CV” For each condition, the mean and the mode of for the classifier accuracy is given as well as the mean and range for non-zero (NZ) coefficients in the final model. The mode of the top 3

SCNs for each condition is also presented. For these tests GLMnet variable α is given and use of grouped Lasso is indicated by “yes” in the column labeled “gLR”.

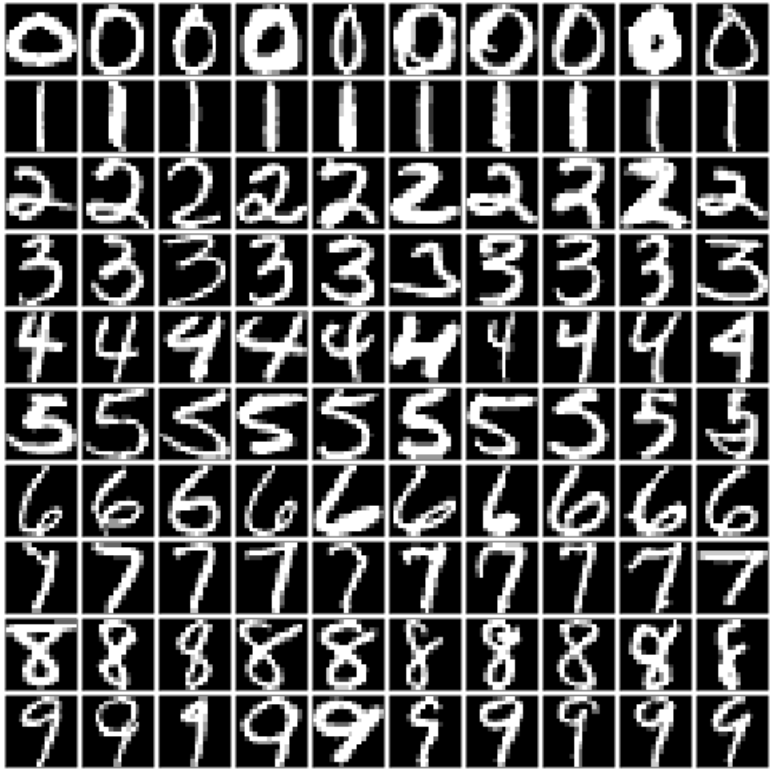
Table 2: Effects of image pre-rotation on classification accuracy with the USPS dataset

Group	Pre-R	gLR	CV	Mean-T	Range-T	Mean-NZ	Range-NZ	Stack	Mode-T	Top 3
1	0	no	no	0.9509	0.9437-0.9562	263	253-272	288	0.9591	0.9576
2	0	yes	no	0.9514	0.9457-0.9562	277	214-288	288	0.9591	0.9572
3	r(0,90,30)	no	no	0.9633	0.9571-0.9651	485	446-532	1152	0.9661	0.9671
4	r(0,90,30)	no	yes	0.9631	0.9591-0.9661	74	67-85	1152	0.9676	0.9661
5	r(0,90,30)	yes	no	0.9632	0.9591-0.9661	1073	975-1145	1152	0.9666	0.9671
6	r(0,90,30)	yes	yes	0.9634	0.9562-0.9686	1114	1046-1145	1152	0.9661	0.9696
7	r(0,180,30)	no	no	0.9609	0.9552-0.9646	579	558-618	2016	0.9631	0.9686
8	r(0,180,30)	yes	no	0.9676	0.9621-0.9716	1728	1414-1971	2016	0.9696	0.9726
9	r(0,360,40)	no	no	0.9687	0.9671-0.9701	562	480-562	2880	0.9711	0.9716
10	r(0,360,40)	yes	no	0.9690	0.9646-0.9721	1945	1543-2530	2880	0.9716	0.9736
11	r(0,360,30)	no	no	0.9633	0.9571-0.9676	646	608-702	3744	0.9656	0.9686
12	r(0,360,30)	no	yes	0.9633	0.9601-0.9666	86	80-97	3744	0.9691	0.9696
13	r(0,360,30)	yes	no	0.9741	0.9681-0.9761	2784	1968-3574	3744	0.9736	0.9756
14	r(0,360,30)	yes	no	0.9656	0.9656-0.9721	2506	2361-2740	3744	0.9731	0.9741
15	r(0,360,20)	no	no	0.9656	0.9616-0.9676	687	610-746	5472	0.9681	0.9691
16	r(0,360,20)	yes	no	0.9721	0.9681-0.9756	2870	2230-3609	5472	0.9741	0.9756
17	r(0,320,40)	no	no	0.9662	0.9646-0.9686	643	582-697	3888	0.9681	0.9696
18	r(0,320,40)	no	yes	0.9657	0.9641-0.9676	82	77-85	3888	0.9691	0.9686
19	r(0,320,40)	yes	no	0.9714	0.9701-0.9731	2245	1641-3201	3888	0.9736	0.9736
20	r(0,320,40)	yes	yes	0.9724	0.9711-0.9736	2419	2101-2706	3888	0.9736	0.9746
21	r(0,360,10)	no	no	0.9553	0.9487-0.9606	319	291-358	10656	0.9616	0.9626
22	r(0,360,10)	yes	no	0.9641	0.9606-0.9686	318	290-371	10656	0.9661	0.9691

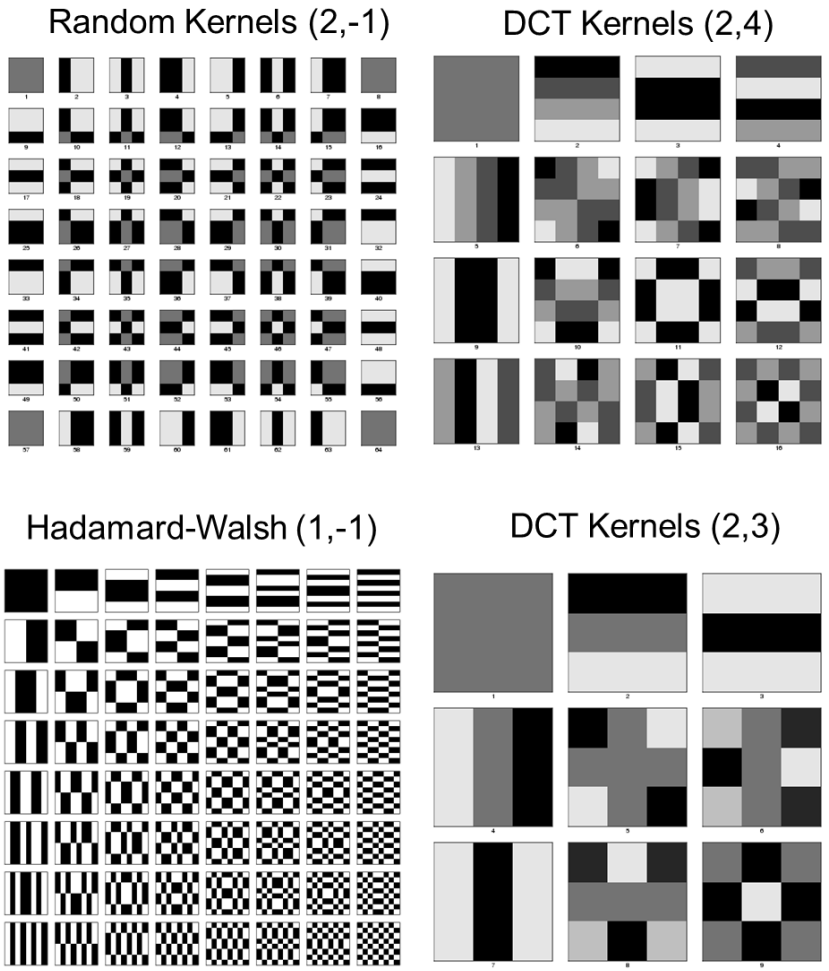
Image rotation is coded as r= r(start, finish, size of steps in between). Results are based on 11 SCNs per group constructed by randomly selecting 8 RAN(2,-1) kernels in each tranche. Grouped refers to whether gLR was used as a regularizer by the classifier, with “no” indicating that LR was used instead. Classifiers were trained without cross-validation and with α set to 1. The mean proportion of correct calls for the test set by the ensemble along with the range for individual SCNs is given. The mean number of non-zero (NZ) coefficients after coordinate descent and the range for individual SCNs is tabulated. The total number of columns in the array is in the column labeled “Stack” and represents the length of the image vector after pre-rotation, local convolution , patch pooling and linearizing. Scoring is based on the mode of the calls by the 11 SCNs (Mode-T) and by top three SCNs (Top 3).

Supplementary Figure SF1

a.



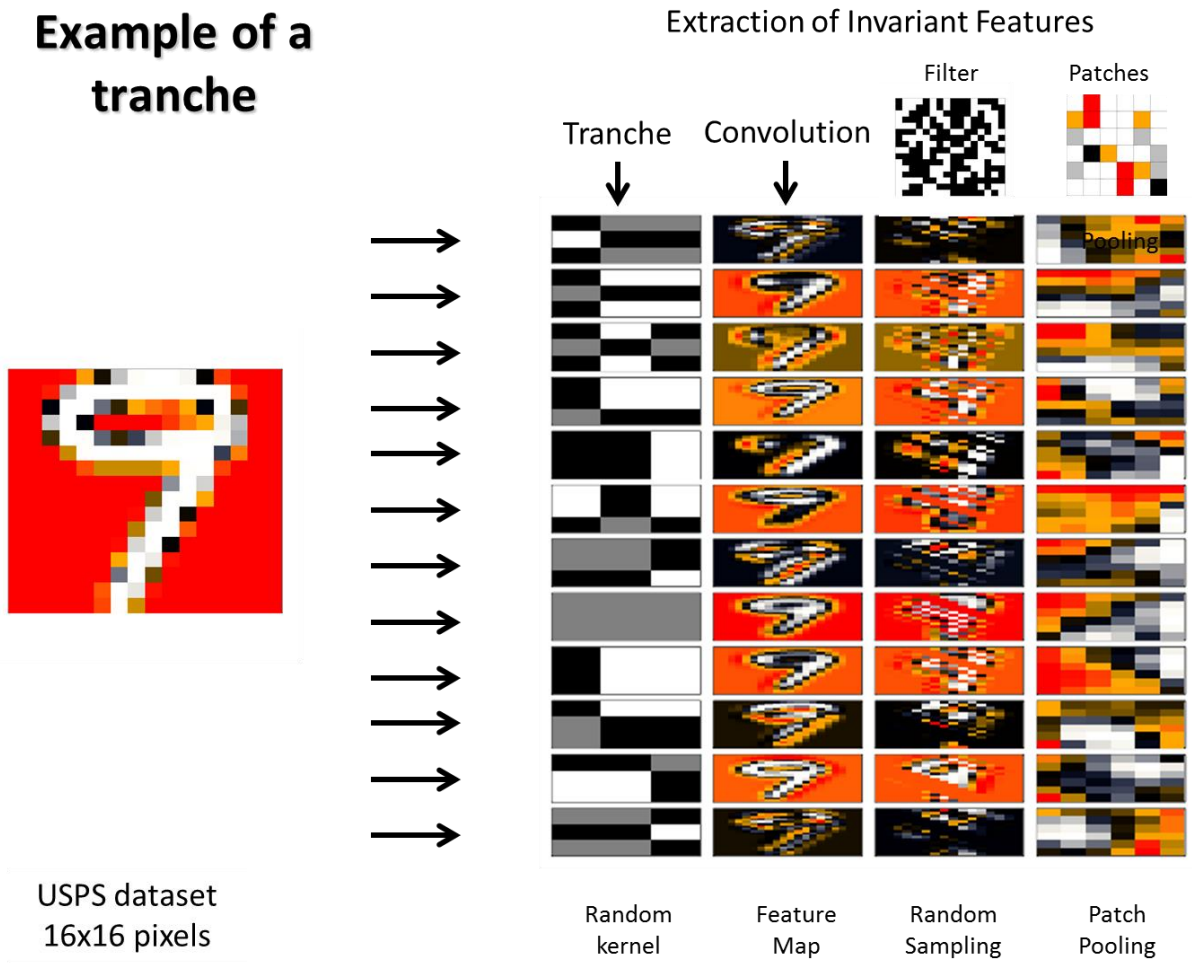
b.



a. A small sample of hand written digits from the USPS dataset used to test classification by SCNs.

b. The convolution kernels are of low complexity. The random kernels (RAN(2,-1)) are the inner product of 3 tuples constructed from the left and right half of 6 tuple vectors representing all 64 possible combination of the elements (2,-1). The discrete cosine transform kernels (DCT(2,3)) and (DCT(2,4)) are the outer product of a type 2 Discrete Cosine Transform 3 x3 and 4 x 4 matrices. The Hadamard Walsh kernels (HW(1,-1)) are the outer product of size 8 Hadamard matrices with elements (1,-1) ordered by sequency.

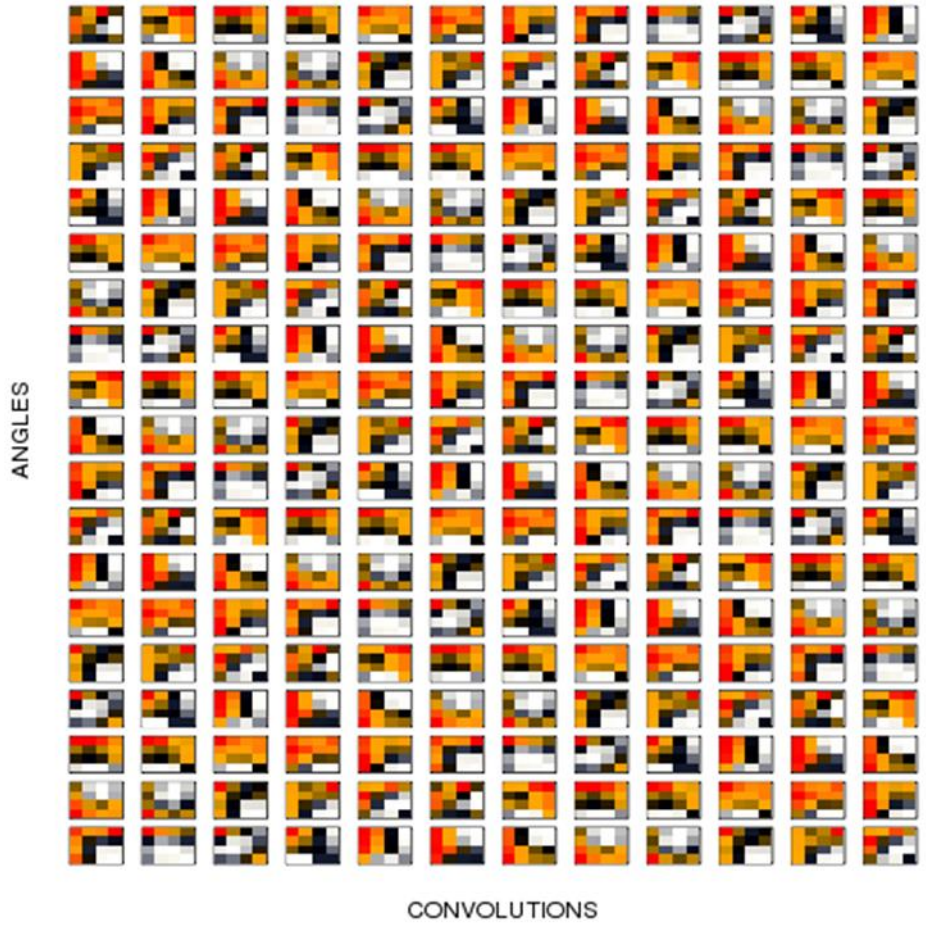
Supplementary Figure SF2



An example of image processing, proceeding left to right. The image of the digit “9” undergoes local convolution with a tranche of 12 randomly selected kernels from set RAN(2,-1) that are shown in Figure SF1b to produce a set of feature maps that are randomly

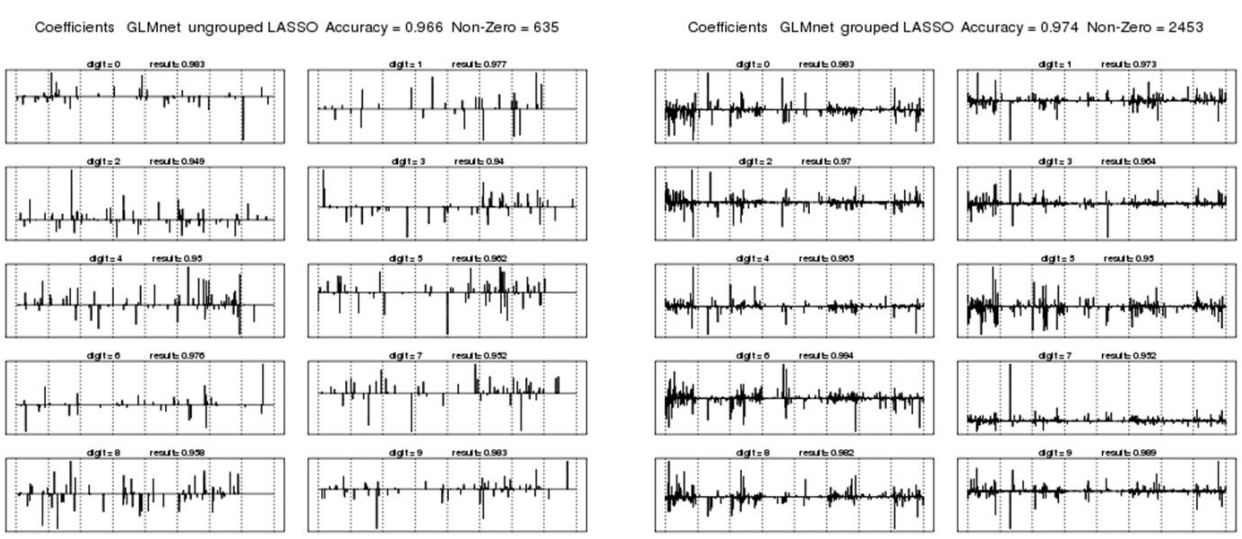
sampled and then patch pooled using 6 x 6 overlapping tiles to produce the outputs that are destined for the classifier. This process is repeated for all the rotations of that image.

Supplementary Figure SF3



Input to the classifier showing the output from processing of one image after 19 angles of rotation with R=(r(0,260,20))and using a tranche made with t=12 RAN(2,-1) kernels.

Supplementary Figure SF4.



Coefficient Plot for classifiers for each number generated using a lasso regularizer LR (left) or a group lasso regularizer gLR(right). The USPS dataset (16 x 16) , rotation set r= r(0,360,30), tranche with kernel size k=8 and random sample s=9 with patch size p=6 were used to process the images. The 10-fold cross-validation accuracy and number of non-zero coefficients is shown at the top of each figure. The dotted lines in each panel separate the output from each randomly selected kernel.