StrongNet: mostly unsupervised image recognition with strong neurons

Sergey Bochkanov, ALGLIB Project,

with special thanks to Elvira Illarionova.

SUMMARY: This technical report proposes two innovations in the design of neural networks: **(a)** *strong neurons* – highly nonlinear/nonsmooth neurons with multiple outputs and **(b)** *mostly unsupervised architecture* – backpropagation-free design with all layers except for the last one being trained in a completely unsupervised setting.

The new neural design (called StrongNet) was tested on a well-known MNIST benchmark. This demonstrated that StrongNet is able to capture structural information about images in the "mostly unsupervised" setting. Three-layer StrongNet has a test set error on MNIST as low as 1.1%. It outperforms 11 out of 14 two/three-layer networks cited on the MNIST homepage (as of Aug 2014).

The most important result achieved by us is that StrongNet – architecture that includes only one supervised layer – successfully competes with "completely supervised" shallow networks, which have up to three supervised layers. This result demonstrates the feasibility of the new approach and suggests the performance of deeper StrongNets should be investigated.

PUBLISHED: August 13, 2014.

1. Introduction

The recent decade has shown great practical and theoretical advances in neural designs: optimized implementations (parallel and distributed training frameworks), improved training algorithms (unsupervised pretraining), and architectural improvements (convolutional networks) (see [12]).

The performance of the new methods on the well-known MNIST dataset improved from year to year – from several percentage points to ~1% (best two/three-layer networks [1], [12]), then down to ~0.5% (convolutional networks [1]), then down to 0.25% (committees of convolutional networks [2]).

However, almost all successful methods were complex multilayer backpropagation models explicitly fitted to the data. They have been successful, but they do not reproduce the internals of the best pattern recognizer of all times – the visual cortex.

Several points should be noted:

(1) Backpropagation is biologically implausible. It is known that biological neurons may carry some information in the backward direction (so called "neural backpropagation," see [14]), but it is unlikely that this mechanism can propagate an error signal through multiple neural layers of brain.

(2) It is commonly agreed that biological neurons are far more complex than simplified models used in machine learning – and some authors [15] suggest that neurons are even more complex than spiking models.

(3) Even an untrained several-year-old child can solve a MNIST benchmark with a fairly low error. The child does not understand the *meaning* of the digits, but even without supervised training for any digit pair, the child may say whether the digits are the *same*

or *different*.

(4) Humans require a very limited amount of supervised information for successfully learning *hard* object recognition tasks (cars, planes, faces) – contrary to supervised neural networks that typically require many repetitions of million-sized datasets of labeled objects.

From (1) and (2) we may conclude that the most successful object recognition methods do not reproduce the behavior of the visual cortex; they can successfully solve recognition tasks, but they use a different approach. For example, it is *possible* to classify objects with decision forests, but the brain definitely does not use decision trees to perform classification.

Points (3) and (4) suggest that the visual cortex relies heavily on some form of nonsupervised learning, which results in natural clustering of similar objects. Supervised processing is performed at the later stages to create an association between clusters (digits) and their meanings.

We should mention that modern neural networks often use unsupervised training for feature detection; but in all cases, the unsupervised stage is treated merely like the *preprocessing* of data, which assumes that actual work is performed in the supervised mode.

In this report we propose a new neural design that relies on the following assumptions:

- that the visual cortex performs the most important data processing in the unsupervised mode, with the supervised stage being just post-processing applied to the cortex output (what we call "*mostly unsupervised*" neural networks);
- that biological neurons are strong pattern recognizers that can react to multiple objects in their receptive fields (see [15]) and that groups of neurons may act as a composite "strong neuron" with multiple inputs and outputs;
- that a simple and local learning rule exists for such "strong neurons," which naturally results in the clustering of similar images.

In section 2 we will study several classes of strong neurons and the corresponding learning rules. Section 3 reviews the general principles of mostly unsupervised neural networks. In section 4 we will study one particular implementation of this idea and its performance on the MNIST dataset. In section 5 we will discuss nontechnical questions, and in section 6 we will draw our conclusions.

2. Strong neurons

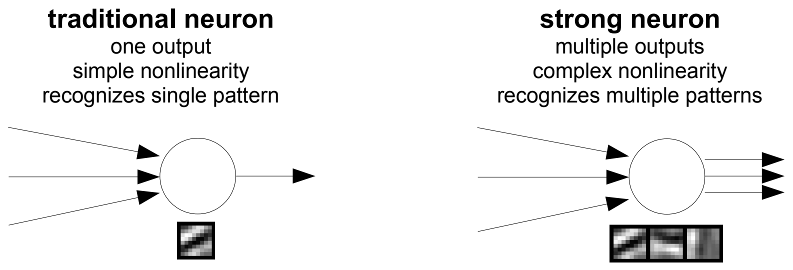


Figure 2.1: on the left we can see traditional neuron which detects just one pattern in its receptive field. Strong neuron (on the right) may remember and detect multiple patterns.

The traditional neuron is a weak processor that can perform only one kind of activity – the detection of just one linearly separable pattern in its receptive field. A strong neuron (see Figure 2.1) has multiple outputs and may perform complex activities, such as (a) working with several nearby receptive fields, (b) processing categorical data, and (c) remembering and detecting multiple patterns.

We propose two kinds of strong neurons: **C-type** and **D-type**, with C-neurons being used on the first layers of the network and D-neurons being used at the later stages of data processing.

C-neurons (see Figure 2.2) are just simple **c**lusterizers (denoted by "C" in their name). They receive analog continuous input (typically a small patch of an image) and classify it according to one of the patterns stored in the neuron's memory. These neurons have two equally important purposes: (a) they are used at the first layer to transform input from analog to categorical representation, which is consumed later by D-type neurons, and (b) they perform initial unsupervised processing of data.

C-neurons usually have a categorical output, with one and always one category being present on its output (**1**-output, as explained later).

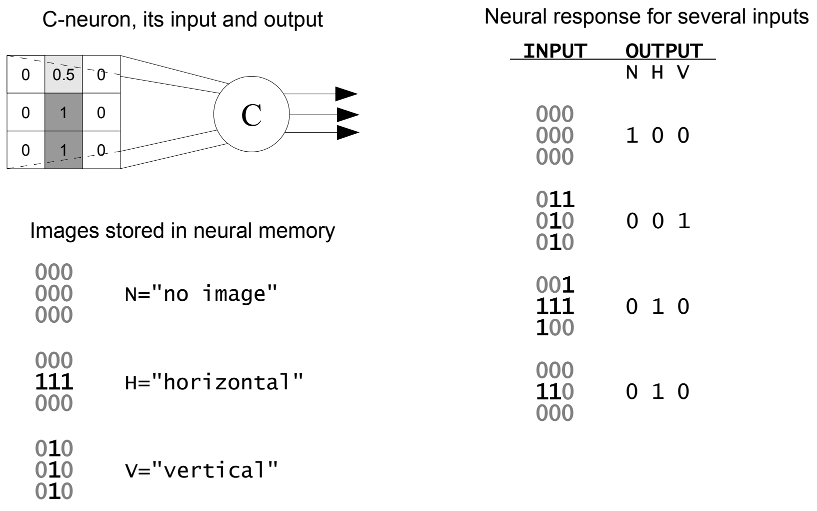


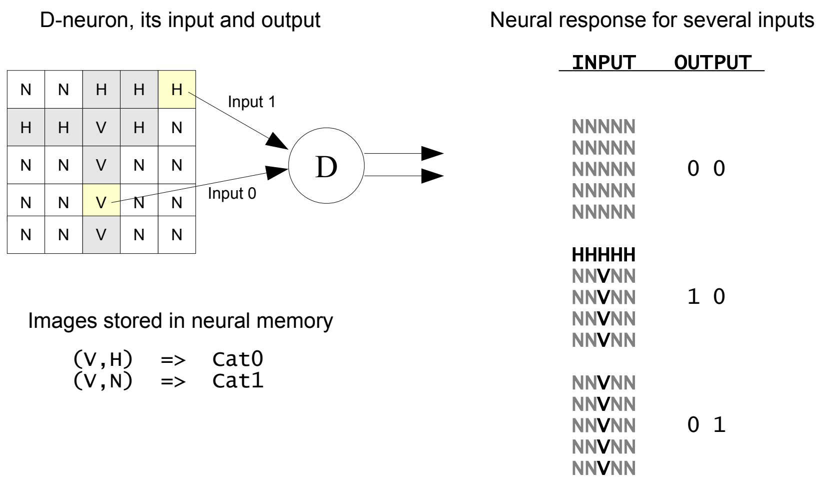
Figure 2.2: on the left we can see strong C-neuron (clusterizer) which classifies its input (3x3 patch) as belonging to one of the clusters stored in neuron's memory. One and only one of outputs is activated. On the right we can see response for several different inputs.

D-neurons are **d**etectors. They have two categorical inputs (or two categorical receptive fields), they have memory, and they detect specific combination(s) of categories in their inputs.

A neuron with two categorical inputs (Figure 2.3) is very simple. Having two categories, *Cat0* and *Cat1*, at input, the neuron tries to find this combination in its memory. If this combination is present (the neuron remembers this pattern), the corresponding output is activated. If the combination is not found in the neuron's memory, no output is active (**01**-output).

It is also possible for a D-neuron to work with two receptive fields (Figure 2.4) – two small grids of neurons with categorical outputs. However, it does not mean that the D-neuron remembers combinations of *image patches*. As before, it stores the combination of just two categorical values, but now it examines all possible input combinations from the first field and input from the second field in its search for these categories.

The idea of receptive fields instead of scalar inputs is to capture stronger correlations – stronger than possible with just two 1x1 inputs.

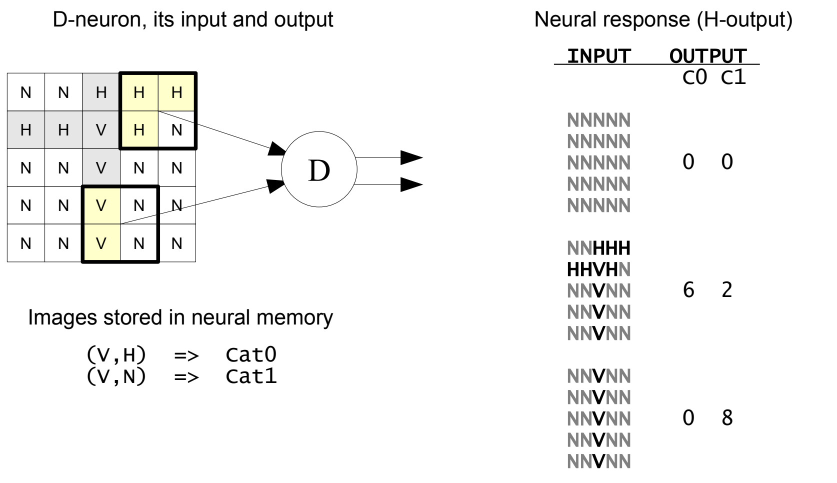


D-neuron, its input and output
N N H H H
H H V H N
N N V N N
N N V N N
N N V N N
Input 1
Input 0
D
Images stored in neural memory
(V,H) => Cat0
(V,N) => Cat1
Neural response for several inputs
INPUT OUTPUT
NNNNN NNNNN NNNNN NNNNN NNNNN 0 0
HHHHH NNVNN NNVNN NNVNN NNVNN 1 0
NNVNN NNVNN NNVNN NNVNN NNVNN 0 1

Figure 2.3: on the left we can see strong D-neuron (detector) which detects specific combination of vertical ("V") and horizontal ("H") lines in its 1x1 receptive fields. At most one of outputs is activated. On the right we can see response for several different inputs.

A D-neuron with receptive fields as inputs may have different types of outputs:

- First, it is possible to have a familiar **01**-output – we activate at most one class. Usually, we select the class that receives the majority of votes from the receptive fields (or no class, if no pattern is found). This input can be seen in Figure 2.3.
- It is also possible to activate several outputs – one per pattern detected. Such an output is called 11-output (two 1's are used in the name because several active outputs are possible).
- Finally, we can simply return a histogram – a so-called H-output – which is shown in Figure 2.4



D-neuron, its input and output
N N H H H
H H V H N
N N V N N
N N V N N
N N V N N
D
Images stored in neural memory
(V,H) => Cat0
(V,N) => Cat1
Neural response (H-output)
INPUT OUTPUT
C0 C1
NNNNN NNNNN NNNNN NNNNN NNNNN 0 0
NNHHH HHVHN NNVNN NNVNN NNVNN 6 2
NNVNN NNVNN NNVNN NNVNN NNVNN 0 8

Figure 2.4: on the left we can see strong D-neuron (detector) with two 2x2 receptive fields. On the right we can see histogram response (H-output) for several different inputs.

As we are now familiarized with strong neurons, we can move to *learning rules*. Unlike their supervised counterparts, strong neurons are trained *locally* – we do not need backpropagation to train them.

> **NOTE:** supervised neural networks (both convolutional and nonconvolutional) use backpropagation, which is a *global* rule. In order to train specific neurons we need information about the neurons from the upper layers. Thus, information moves in two directions – first from the bottom layers to the top, then from the top to the bottom. On the backward pass, we work with an integrated response from the entire network.

C-neurons (clusterizers) are "trained" with K-means. They do not receive any response from upper layers during training. We just sample all possible inputs (small image patches), run K-means, and use centers as patterns stored in the neuron's memory. In fact, we can call it "network construction" or "neural development" instead of training.

Such an approach was known long before this report; however, in supervised networks it was always used just as a "feature detector." The rest of the network was trained with a supervised algorithm. This report discusses another approach – to integrate K-means into a mostly unsupervised network.

D-neurons (detectors) are "trained" with another simple algorithm. Each D-neuron maintains its own contingency table for the combination of its inputs. After sampling enough data, it marks some "interesting" entries in the table as "patterns" and responds only to these patterns.

Several ways exist to recognize a specific combination of categories as "interesting." First, it is possible to respond to the most probable, most frequent combinations of inputs. We called this "P-rating" – probability rating.

Another, more promising approach is estimating probability distributions for inputs #0 and #1, $P_0(category)$ and $P_1(category)$, and searching for such combinations of categories so that $P(cat_0, cat_1) > P_0(cat_0) * P_1(cat_1)$, i.e., to detect *patterns* that arise more frequently than would be possible by chance alone. We call this second approach "Sigma-rating" after the standard deviation (sigma) used to estimate the significance of the difference in probabilities.

Both approaches work but with different performance. During our experiments we found that Sigma-rating is better at capturing the most important informative dependencies. P-rating also allows us to build a working classifier, but it is consistently outperformed by Sigma-rating. The only drawback of Sigma-rating is that it cannot mark *all* combinations of inputs as patterns because it is impossible for all combinations to arise more frequently than expected by chance alone (sum of their probabilities will be larger than 1.0).

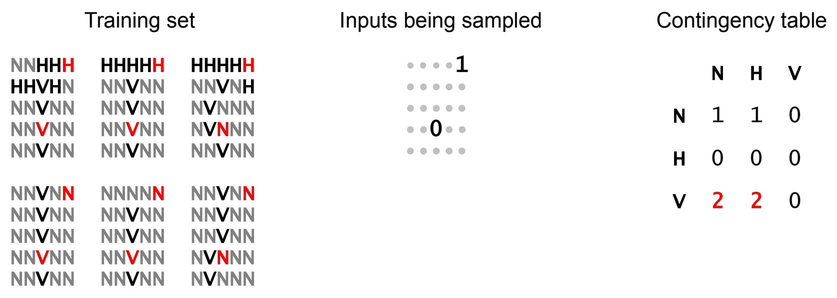


Figure 2.5: how contingency table is built. We sample a pair of inputs from the training set (for the sake of simplicity we do not consider receptive fields) and build contingency table. Two entries with highest probabilities (V,H) and (V,N) are marked as "patterns" (P-rating was used to recognize most important combinations).

It is interesting that an approach similar to Sigma-rating was mentioned several decades before us in [9] (Pearlmutter and Hinton, 1986). The authors used a procedure called "G maximization" to detect patterns in the input area of a *continuous* neuron. This idea has been mentioned several times since then (e.g., in [8], Barlow, 1989) and was even reinvented by some authors, but no one proposed to use it for neurons with categorical outputs.

Such an approach is strikingly different from backpropagation – we do not try to explicitly or implicitly minimize error on the training set! We just perform some local operation in order to detect interesting patterns in a neuron's inputs without ever thinking about the classification stage.

Obviously, such an approach to training won't work in the "general case of nonlinear classification in N-dimensional hyperspace." But, surprisingly, it works well for image classification problems.

Another striking difference from backpropagation networks is that strong neurons deal with categorical nonsmooth inputs and outputs. After years with "conventional" neural networks, we started to think that "*neural*"="*smoothness*." Even non-neural models, like committees of decision trees, introduce some degree of smoothness by averaging results from multiple nonsmooth models.

Again, the success of an intrinsically nonsmooth neural model is a surprise to us.

3. StrongNet

One obvious way to use an idea of strong neurons is to create a backpropagation network of strong highly nonlinear neurons and to train it in a supervised setting. Such an approach is dominant in the machine-learning field – to fit complex, multilayer, highly nonlinear models to the data.

However, the strong neurons proposed by us do not need fitting – they may learn without any supervised feedback. We will see later that they really learn, i.e., perform some kind of information processing that improves the linear separability of different image classes. We will also show that "strong" is not about a higher processing power – it is about being able to use a completely different approach to recognition.

So, our neurons do not need supervised training. But what do they need? It turns out that it is enough to organize them into unsupervised layers with only minor supervised *post-processing*.

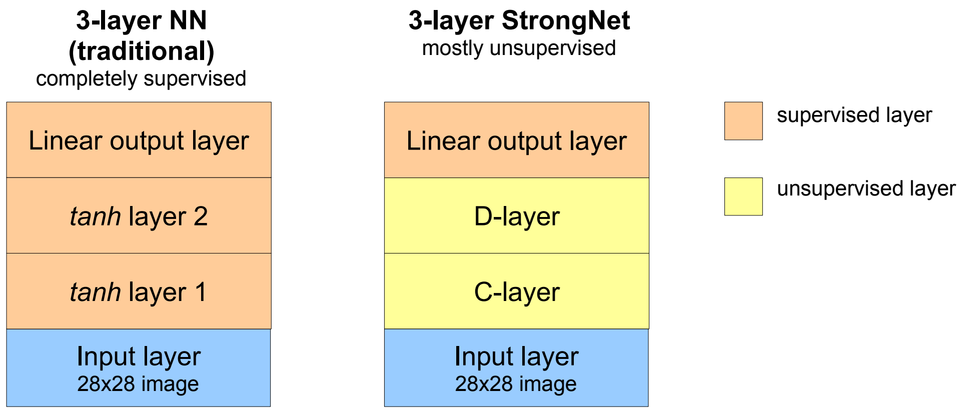


Figure 3.1: completely supervised network (on the left) with all of its coefficients being tuned in the supervised manner. Two layers of the strong neural network (on the right) are constructed in the unsupervised manner, and only last layer is trained.

In the figure above we see one incarnation of StrongNet – a network with two unsupervised inner layers and one supervised output layer. In fact, it is the only design that has been tested by us so far, but it demonstrates the most important properties that should remain in other StrongNet designs:

1. an unsupervised C-layer (clusterizer) is used at the beginning of the network to perform the initial transformation from analog to categorical representation;
2. subsequent unsupervised layers are layers of detector (D-type) neurons;
3. we have only one supervised linear layer (output one).

The following requirements should be met by all unsupervised layers (except for, perhaps, the last unsupervised one):

a) spatial structure of the layer is important – neurons must be organized into a grid;

b) all neurons within a layer must have a *shared vocabulary*, i.e., when two neurons activate the same outputs, this means that they responded to the same inputs. It does not mean that *all* neurons must respond to the same inputs. Different neurons may recognize different patterns in the data, but they must use the same vocabulary when they report their findings.

These requirements allow us to have D-neurons with receptive fields instead of scalar inputs: (a) is essential because the receptive field is a spatial structure, and (b) is important because without this requirement it would be impossible to aggregate responses from different input neurons.

For the last unsupervised layer, these requirements are relaxed because its output is fed directly to a supervised linear layer, which is able to consume any data independently of their structure. So, in the last layer it is possible to have spatially unstructured neurons, neurons with *local vocabularies*, or even with different numbers of outputs.

The linear output layer is the only layer where *training* is performed. The layer can be trained with an iterative algorithm (GD, SGD, LBFGS) or with a direct method (dense, direct, linear, least squares solver). In our experiments we preferred the direct solver, but any method will work. All other layers are *constructed* around information flowing from input to output.

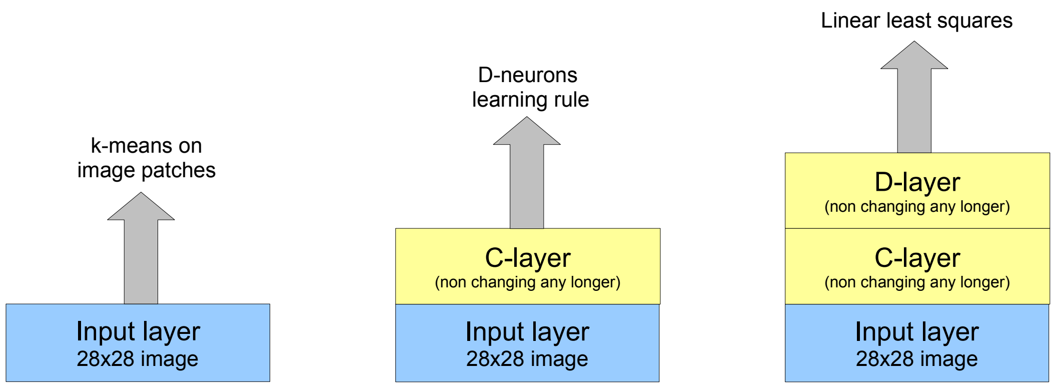


Figure 3.2: development of StrongNet: input layer, addition of C-layer, addition of D-layer, then supervised learning (linear least squares) on outputs of last unsupervised layer.

We believe that this unidirectional flow of information is one of the greatest benefits of StrongNet. Learning rules for C- and D-layers are simple and easily parallelizable – and are not prone to the problem of bad local extrema.

One interesting question that may arise is how does it differ from yet another feature detector? Is StrongNet really something new – or just a combination of a nonlinear feature detector with a linear neural network?

There is no universally agreed definition of a feature detector, but most sources agree that:

- a feature detector usually has low complexity;
- feature detection is low-level *preprocessing* of an image;
- actual processing of data takes place at later supervised stages of the algorithm;
- generally, you start with a "good enough" feature detector. If the results are not acceptable, you build a deeper neural model around the same detector, i.e., the detector is considered fixed, and you tweak a nonlinear model built on top of it.

However, with StrongNet we have:

- a highly nonlinear unsupervised stage
- a fixed-complexity supervised layer
- if you are not happy with the results, you improve the *unsupervised stage*

Our opinion is that the results on the MNIST set (see the next section) suggest that StrongNet is more than just a feature detector.

4. StrongNet on MNIST

In order to test StrongNet, we used a well-known MNIST benchmark.

MNIST is a collection of 70,000 labeled images (28x28 pixels, grayscale), separated into training (60,000) and test (10,000) sets. Each image is a handwritten digit (0 to 9), so we have 10 classes. The MNIST homepage (*http://yann.lecun.com/exdb/mnist/*) includes both the dataset itself and an extensive list of results achieved with different methods.

Results achieved with two/three-layer neural networks range from a 4.7% classification error ([4], 1998) to a 0.7% classification error ([1], 2003). The best results are achieved with deep models: convolutional networks, 0.35% ([5], 2011), and committees of convolutional networks, 0.23% ([2], 2012).

Because we tested a three-layer StrongNet, we compared its performance with that of two/three-layer models (which is in the range 4.7%–0.7%).

We used StrongNet with the following properties:

- C-layer with F_1 outputs, "trained" with K-means clustering on 4x4 input patches. C-neurons form a 25x25 grid (25=28–4+1) and share common vocabulary. In some models 4x4 images patches were whitened before applying K-means (in this case whitening transformation was an integral part of the network);
- D-layer with F_2 outputs and random spatial structure. We randomly selected pairs of points on the grid formed by the previous layer, such that the distance between these points was in [4,6] range. For each such pair we formed two 3x3 receptive fields centered around chosen grid points and trained the D-neuron with these receptive fields as inputs. Such neurons were trained to recognize up to F_2 patterns in their inputs. Layer formation stopped after the total number of outputs exceeded M;
- linear layer with M inputs and 10 outputs; this layer was "trained" with a direct linear least squares solver (mean squared error was minimized);
- Sigma-rating to detect patterns in receptive fields.

The network was "trained" using only a training set (60,000 images) and tested using a completely separate test set (10,000 images). We tested many combinations of feature counts F_1 / F_2, different linear layer sizes, and different kinds of outputs for the D-layer. The tables below summarize the most interesting results:

Parameters	Error (test set)
F1=F2=5, M=1500, H-output	2.7%
F1=F2=5, M=3000, H-output	2.0%
F1=F2=5, M=5000, H-output	1.9%
F1=5, F2=10, M=5000, H-output	1.8%
F1=10, F2=10, M=5000, H-output	1.6%
F1=10, F2=10, M=8000, H-output	1.4%
F1=12, F2=10, M=8000, H-output	1.3%
F1=12, F2=12, M=10000 Affine distortions Whitening on 4x4 image patches H-output	**1.1%**

Table 4.1: MNIST test set error for 3-layer StrongNets with different parameters.

As we can see, the most successful combination is:

- $F_1 = 12$ – each neuron of the C-layer remembers and detects 12 patterns
- $F_2 = 12$ – each neuron of the D-layer remembers and detects 12 patterns
- $M = 8000$ – the overall count of outputs in all neurons of D-layer is 8000, which is the input size for the linear supervised layer
- affine distortions are applied to the training set in order to increase its size from 60.000 to 120.000 elements

The test set error was 1.1%, which beats 11 out of 14 shallow neural networks cited on the MNIST homepage. We note that these shallow neural networks are completely supervised architectures, i.e., architectures where *all* coefficients are tuned in the supervised mode in order to reduce the training set error. StrongNet, however, has only one layer of supervised coefficients – and still it competes with the traditional approach.

Another point to note is that the idea of using K-means (C-layer of StrongNet) in pattern recognition is not new. We can cite [6] and [7] as examples of using K-means for feature extraction. However, most authors prefer to use a very large number of features/clusters even for small image patches. For example, [7] uses 500 clusters to extract features from photos of street numbers. We managed to get good results with just 12 features, thanks to the layer of strong D-neurons.

It is interesting to study how unsupervised processing changes data properties. The most important property is linear separability – our ability to separate classes in a dataset with just a linear model. The smaller the linear separability error is, the better suited is the dataset for classification. Another metric to study is the number of dimensions necessary to represent the dataset.

Stage	Data dimensionality	Linear separability error (test set)
Input layer (28x28 image)	784	14.1%
After C-layer	9408	3.6%
After D-layer	10000	1.1%

Table 4.2: changes in data properties after different stages of processing.

As we can see, unsupervised processing greatly improves the linear separability of the data. The C-layer alone improves the error from 14.1% to 3.6% (4X improvement). The second layer of StrongNet gives an additional three times the improvement (the error is reduced to 1.1%).

These data confirm that, in StrongNet, most of the work is performed by strong unsupervised neurons and that the linear supervised stage is merely post-processing applied to network outputs.

5. Discussion and questions

In the previous sections we studied the idea of strong neurons, learning rules for them, and "mostly unsupervised" neural design. In this section we want to move away from the details and instead discuss the big picture. This discussion may be a bit speculative, but we feel that some questions should be discussed, even if only conjecturally with no scientifically proven knowledge.

On unsupervised learning.

First, for us it is a mystery that the separability error is systematically reduced by a completely unsupervised procedure that is not explicitly aimed at improving linear separability.

For example, it is not surprising that deep neural networks are good at dealing with MNIST.

If you (a) fit flexible model to the data and (b) take precautions against overfitting, then you may reasonably expect that the model will perform well on the test set. However, from this point of view, the success of StrongNet is *unexpected*.

It is possible to explain the success of StrongNet by saying that "the C-layer extracts features from the data" and "the D-layer detects patterns in the features." But we are not satisfied with such an explanation. For us, a really good explanation must also answer the following questions:

- Why is StrongNet better than the linear classifier? For a conventional neural network, the answer is simple. You can model a linear classifier with a network, thus a network is, at least, not worse than the linear classifier. However, it is not so obvious with StrongNet.
- Is it possible to reach a state-of-the-art error on MNIST (0.25%) with StrongNet or are there intrinsic limits on its performance? For example, a large enough neural network is guaranteed to solve any problem given enough data and computing power [10] (the question is how to train a network with *limited* data and computing power). In contrast, a linear classifier has intrinsically limited discriminating power. But what about StrongNet?
- Will such an approach work on speech recognition problems? If no, why? If yes, why?

- On what classes of problems could we expect success with unsupervised approaches? What do all these problems have in common? "Cluster assumption" is the answer to this question, but is it possible to give another response?

We ask these questions because we feel that unsupervised learning i*n general* (not just StrongNet) is not yet well understood.

In recent years computer vision experts achieved success with unsupervised feature extraction (and with unsupervised pretraining, which is another interesting topic for discussion). But in most cases, unsupervised learning is used as a trick that "just makes things better."

Today we have an extensive theory of supervised learning, which: (a) explains *what* we do when we fit a model to data, (b) includes an analysis of asymptotic cases for various kinds of models, and (c) studies overfitting and explains *why* regularization works. VC theory [11] is a good example of theoretical advances in this direction.

For unsupervised learning we have no such extensive theory, mostly just a set of general methods and problem-specific tricks. We feel that unsupervised learning is too good at solving important problems and it is not fully explained within the current theoretical framework.

On simple solutions.

Another question we want to discuss is the idea of "simple approaches to important problems."

We know that the idea of supervised learning may be expressed simply as a "fit flexible model to data." We know that supervised learning may be implemented with neural networks or decision forests – a simple solution (*let's forget about implementation details, ok?*). We know that the idea of unsupervised learning may be expressed as "detecting structure in the data" – again, a simple definition. The strong neurons proposed by us are really simple data processors – strong, but simple – and a very simple learning rule for D-neurons can give impressive results.

*So, what if **all** high-level activities of the human brain are guided by simple design principles?*

Our conjecture seems to be plausible from the biological viewpoint. The developing brain has no supervisor that shapes it as new neurons are formed. Its growth is guided by internal mechanisms, and the fact that these mechanisms give stable results suggests that they must be really simple – because complex is brittle and less likely to result from natural evolution.

Vision must be simple. Hearing must be simple. Walking must be simple.

INTELLIGENCE MAY BE SIMPLE TOO.

Back in 1989, H.B. Barlow [8] formulatedvery simple principles of human perception – "decorrelate correlated, detect unexpectedly frequent". We believe that it is possible to make an equally simple formulation for the high-level cognitive functions of the human brain... although we do not propose one in this paper.

6. Conclusions

In this report we presented StrongNet – a novel neural design that features two innovations:

(a) highly nonlinear/nonsmooth strong neurons with multiple outputs, which may recognize multiple patterns in their receptive fields, and

(b) backpropagation-free architecture, where all layers except for the last are constructed in an unsupervised mode.

StrongNet is "trained" by the addition of self-organizing unsupervised layers. Only the last layer (linear) is trained in the conventional sense – by fitting a model to desired outputs (which can be done with a direct linear least squares solver). Such an architecture was modeled after the visual cortex, which, we believe, extensively uses unsupervised learning for image recognition. Because network "training" is noniterative, it has a constant running time, is not prone to problems of bad local extrema, and is easy to parallelize.

It is possible to use features extracted by unsupervised layers of StrongNet as input for some advanced supervised method (say, SVM or a convolutional network). However, we are interested in performing the most important parts of data processing in an unsupervised mode, that's why we used a linear output layer.

We tested one specific three-layer StrongNet on a MNIST benchmark and demonstrated that it successfully competes with traditional two/three-layer designs (test error was 1.1%, outperforms 11 out of 14 shallow networks cited on the MNIST homepage).

This result demonstrates the feasibility of our new approach and suggests a need to investigate the performance of deeper StrongNets.

References

[1] Simard et al., ICDAR 2003

[2] Dan Cireşan, Ueli Meier, Juergen Schmidhuber (2012). "Multi-column Deep Neural Networks for Image Classification".

[3] Kai Labusch, Erhardt Barth, Thomas Martinetz (2008). "Simple Method for High-Performance Digit Recognition Based on Sparse Coding".

[4] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner (1998). "Gradient-Based Learning Applied to Document Recognition".

[5] Dan C. Cires¸an, Ueli Meier, Jonathan Masci, Luca M. Gambardella, Jurgen Schmidhuber (2011). "Flexible, High Performance Convolutional Neural Networks for Image Classification"

[6] Alexandre Vilcek. (2013) "Scalable Deep Learning for Image Classification with K-Means and SVM".

[7] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng. (2011) "Reading Digits in Natural Images with Unsupervised Feature Learning". *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

[8] Barlow, H. B. (1989) "Unsupervised learning". Neural Comp. 1, 295-311.

[9] Pearlmutter, B.A. and Hinton, G.E. (1986) "G-maximization: An unsupervised learning procedure for discovering regularities".

[10] Kurt Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks", Neural Networks, 4(2), 251–257

[11] Vapnik, Vladimir N (2000). "The Nature of Statistical Learning Theory". Information Science and Statistics. Springer-Verlag.

[12] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner (1998). "Gradient-Based Learning Applied to Document Recognition", Proceedings of the IEEE, 86(11):2278-2324.

[13] Ruslan Salakhutdinov and Geoffrey Hinton, "Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure", AI-Stats 2007.

[14] Stuart, Greg; Nelson Spruston; Bert Sakmann; Michael Häusser (1997). "Action potential initiation and backpropagation in neurons of the mammalian CNS"

[15] Pattern-wave model of brain function. Mechanisms of information processing. A.Redozubov (2014).