# TRIED TPB-03 Report

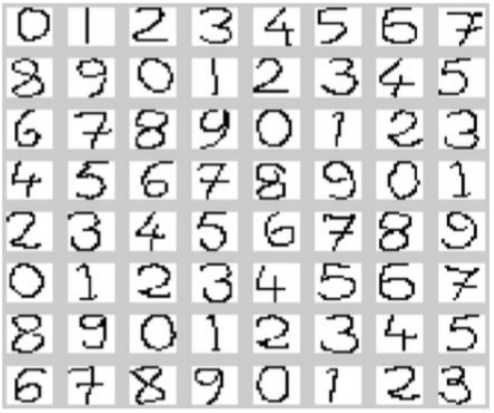
Carl Robinson - 27th Nov 2017

## Part 3: Network with Masks and Shared Weights

## 0) Objective, Data and Method

### Objective

To implement a convolutional neural network with masks and shared weights, and investigate the effect of changing mask size and transfer function on performance. In particular we will examine the hidden layer representation.



### Dataset

Database: 480 individual handwritten digits, in black & white images of 16*16 pixels, represented by a vector of 256 pixels coded ±1.
Labels: 480 vectors of length 10, coded as +1 in the index of the correct digit, and -1 everywhere else.
Training set = 300 examples, Validation set = 100 examples, Test set = 80 examples.

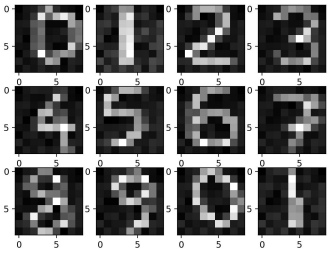
### Method

The neural network was trained 5 times for each combination of transfer function (linear, tanh) and mask size (3, 5, 8, 10, 12). For each combination, the lowest classification error of the 5 runs was recorded, and the internal representation of the hidden layer visualised. An analysis of the results followed.

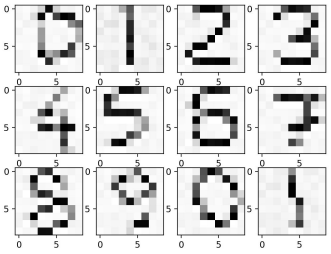
## Internal representations

The following images are the internal representations of the hidden layer, for networks using linear and hyperbolic tangent transfer functions, for masks of various sizes. The pixel values correspond to the linear combination of the shared weights and the unique set of pixels contained by the mask for that neuron.

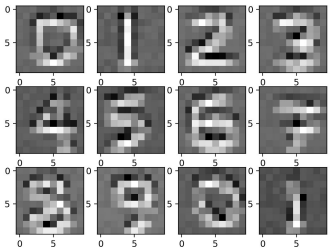
**Linear / 3*3 mask**



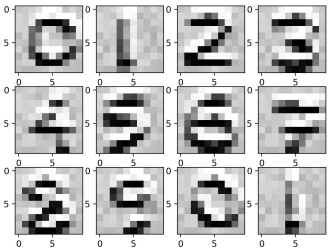
**Tanh / 3*3 mask**



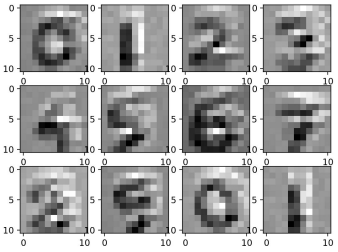
**Linear / 5*5 mask**



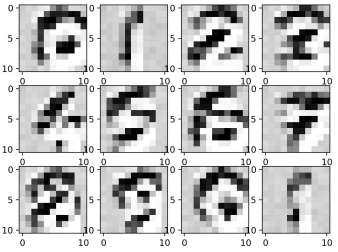
**Tanh / 5*5 mask**



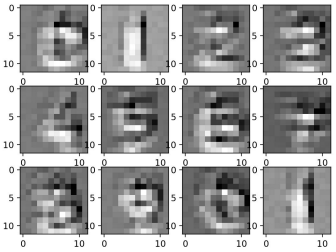
**Linear / 8*8 mask**



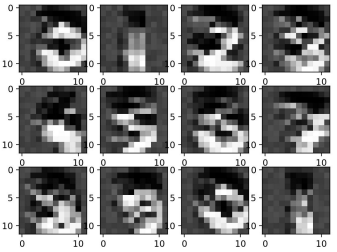
**Tanh / 8*8 mask**



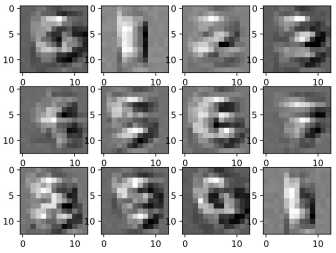
**Linear / 10*10 mask**



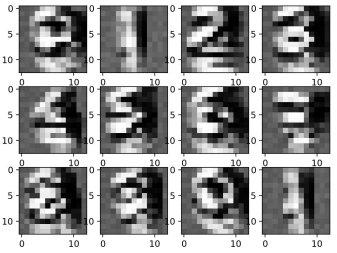
**Tanh / 10*10 mask**



**Linear / 12*12 mask**



**Tanh / 12*12 mask**



- We notice that the blurring and deformation of the visible digits in the internal representations increases as the size of the mask increases.
- The blurring and deformation is worse for the Tanh transfer function than for the linear function for an equivalent mask size. This is because the linear function preserves the differences between pixel values, which the Tanh function squeezes the pixel values at both ends of the scale together.

## Validation Set Classification Performance

**Minimum Classification Error Rate (x 100%) on Validation Set**

| | Mask size (pixels) | | | | |
|---|---|---|---|---|---|
| | **3*3** | **5*5** | **8*8** | **10*10** | **12*12** |
| **Linear** | 0.05 | 0.03 | 0.02 | 0.02 | 0.02 |
| **Tanh** | 0.05 | 0.01 | 0.02 | 0.01 | 0.00 |

- The Tanh transfer function produces a lower error than the linear transfer function for all mask sizes.
- The larger the mask, the lower the classification error.

**Normalised Quadratic Error at Minimum Classification Error Rate on Validation Set**

| | Mask size (pixels) | | | | |
|---|---|---|---|---|---|
| | **3*3** | **5*5** | **8*8** | **10*10** | **12*12** |
| **Linear** | 0.050 | 0.044 | 0.058 | 0.032 | 0.046 |
| **Tanh** | 0.068 | 0.045 | 0.034 | 0.082 | 0.049 |

- There is no discernable correlation in normalised quadratic error between either mask size or transfer function choice.

**Iterations at Global Minimum Classification Error on Validation Set / Average Minimum Iterations (over 5 runs)**

| | Mask size (pixels) | | | | |
|---|---|---|---|---|---|
| | **3*3** | **5*5** | **8*8** | **10*10** | **12*12** |
| **Linear** | 20 / 26 | 30 / 34 | 10 / 18 | 30 / 28 | 10 / 16 |
| **Tanh** | 20 / 18 | 10 / 16 | 40 / 22 | 10 / 30 | 20 / 22 |

- There is no discernable correlation between iterations and either mask size or transfer function choice.
- While this shows that using a larger mask does not increase the number of iterations, each iteration will take longer to compute due to the increased number of weights, and therefore calculations, involved. Therefore iterations are not a reliable measure of calculation time or complexity, and should be regarded with caution.

## Conclusions

### Mask size

- From the table of results, we see that the larger the mask, the lower the classification error.
- From the images, we see that the internal representation is more 'blurred / smooth' for larger masks than smaller masks. This is because as the mask size increases, each neuron in the hidden layer takes a linear combination of a larger range of source image pixels, allowing it to have a strong activation for a wider range of handwritten digits that traverse the mask area. The better results for larger masks could be because of this; they create a hidden layer that that generalises better, as it can handle more variations in handwriting style (translation, rotation and deformation of the digit).
- Larger mask sizes require more weights (parameters) than smaller masks, so are more computationally expensive. There was a very noticeable difference in the computation time on my personal laptop between mask size 3*3 and 12*12. However, as the set of mask weights are shared across all hidden layer neurons, there are far fewer weights to update during back-propagation. This makes the operation much less computationally expensive than if the network was fully connected or if each hidden layer neuron used its own set of weights.
- The 12*12 mask is therefore the preferred mask.

### Transfer function

Tanh is more suitable for classification of nonlinearly separable data such as images of handwritten digits, as it satisfies all of the following characteristics (whereas a linear transfer function satisfies very few):

- **Nonlinear** – a two-layer neural network using a nonlinear activation function can be proven to be a universal function approximator. A linear activation function does not satisfy this property. For binary activation functions (e.g. heaviside step) or linear activation functions, many neurons must be used if trying to solve problems any more complicated than a linear separation of categories. Classifying handwritten digits is one such problem. Only nonlinear activation functions allow such networks to compute nontrivial problems using only a small number of nodes.
- **Continuously differentiable** – desirable for enabling gradient-based optimization methods. A linear activation function has a constant gradient, so is not helpful in this regard either.
- **Limited range** – When the range of the activation function is finite, gradient-based training methods tend to be more stable. A linear activation function has an infinite range.
- **Approximates identity near the origin** - allows the neural network to learn efficiently when the weights are initialized with small random values. Both the linear and tanh activation functions satisfy this property.

From the table of results, we see that the Tanh transfer function does indeed produce a lower classification error rate than the linear transfer function for all mask sizes. Tanh is therefore the preferred activation/transfer function.

A 12*12 mask and Tanh function correctly classified all examples, confirming that this architecture is the most effective.

### Masks vs Coding

In part 2 we found that the lowest quadratic error (0.127) and therefore lowest classification error (0%) was achieved when using a combination of left_profile and right_profile coding. This is the same classification error rate achieved with a 12*12 mask and a Tanh transfer function.

When coding the images with left_profile and right_profile, the input layer has 16+16=32 neurons. The lowest classification error was achieved with m=12. The input layer was fully connected to this hidden layer, resulting in 32*12=384 connections (weights). In contrast, the mask method used a map of 8*8=64 hidden layer neurons, but the weights are shared amongst all hidden layer neurons, so the total number of weights is only 12*12=144. The mask method therefore uses fewer weights between the input and hidden layer.

However, between the hidden layer and the 10-neuron output layer, the coding architecture uses only 12*10=120 weights, while the mask architecture uses 64*10=640 weights.

**Number of Weights used in each Architecture / Coding combination**

| | Weights between input layer and hidden layer | Weights between hidden layer and output layer | Total Weights in Network |
|---|---|---|---|
| Fully connected, coding profile_left, profile_right | 384 | 120 | **504** |
| 12*12 mask with shared weights, no coding | 144 | 640 | **784** |

We can conclude that reducing the dimensionality of the input with a coding scheme reduces the total number of weights in the network more than a mask with shared weights. The two architectures produced the same classification error (0%) but the coding only required approximately two-thirds of the calculations to achieve the same result.

On the other hand, feature engineering is a time consuming task in itself, so the time saved in calculation may well be spent on this instead. Despite this trade-off, the logical next step would be to combine a coded input with a mask and shared weights, to measure the performance gain offered by a combination of both techniques.