Content

Content

Improving the way neural networks learn

- A better choice of cost function — the cross-entropy
- Regularization methods — L1, L2, dropout, Jackknife resampling
- A better initialisation of the weights
- Good choice of hyper-parameters

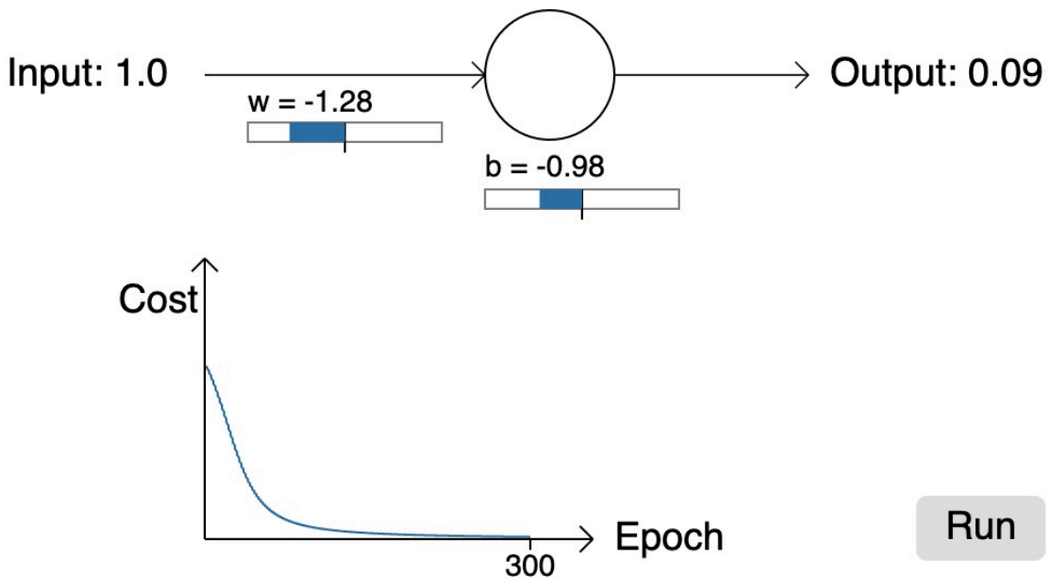
We often learn fastest when we're badly wrong about something. But our artificial neuron has a lot of difficulty learning when it's badly wrong - far more difficulty than when it's just a little wrong.

Starting $w = 0.6, \quad b = 0.9$

input=1, expected output 0, initial output 0.82

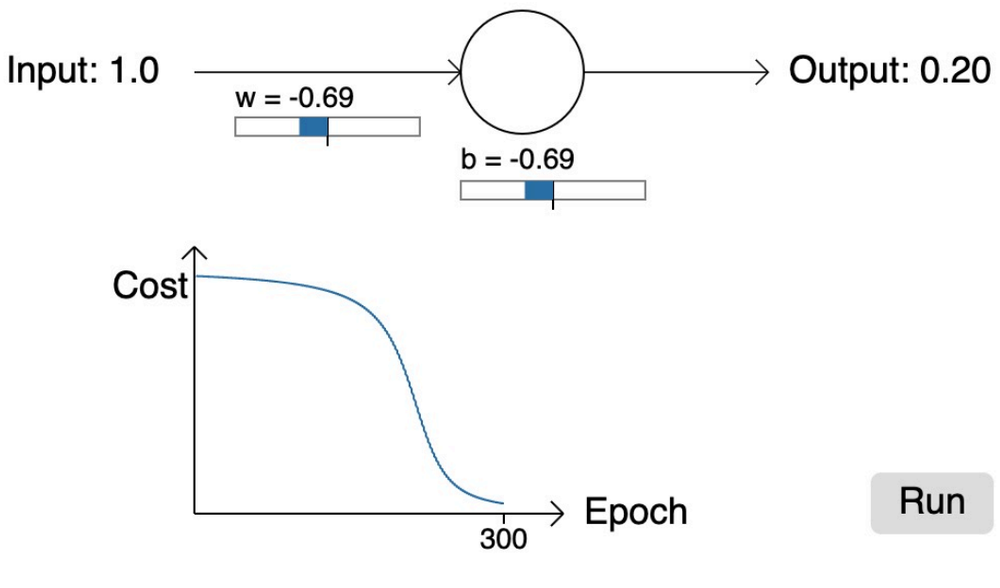
quadratic cost

$$J = \frac{1}{2}(y - a)^2$$



Starting $w = 2, \quad b = 2$

input=1, expected output 0, initial output 0.98

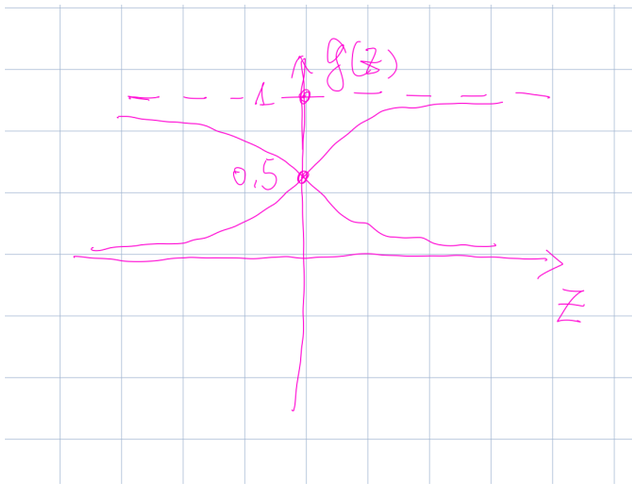


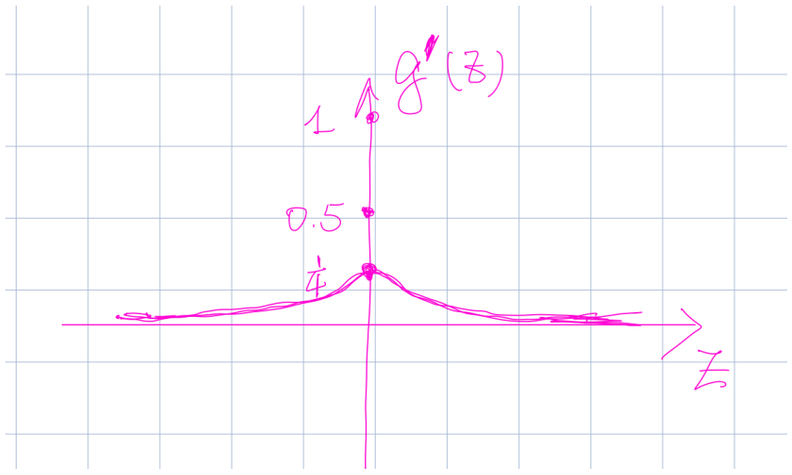
Learning slow down at the saturation

$$\frac{\partial J}{\partial w} = (a - y)g'(z)x = ag'(z)$$

$$\frac{\partial J}{\partial b} = (a - y)g'(z) = ag'(z)$$

when $x = 1$ and $y = 0$





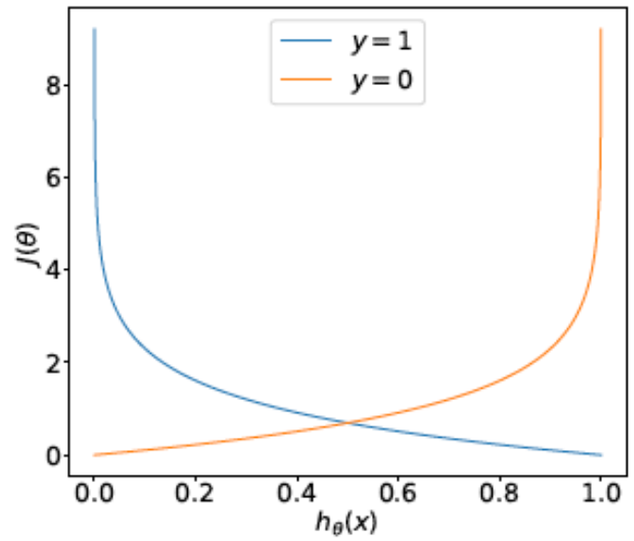
$$\frac{\partial}{\partial \Theta^l_{ij}} J(\Theta) = a^l_j \delta^{l+1}_i \qquad \delta^3 = (\Theta^3)^T \delta^4 .* g'(z^3)$$

Learning slowdown when output neuron saturates

Cross-entropy cost function

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \ln(a_i^L) + (1 - y_i) \ln(1 - a_i^L)]$$

- the cross-entropy is non-negative
- if the neuron's actual output is close to the desired output for all training inputs the cross-entropy will be close to zero
- avoid the learning slow down



$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \ln(a_i^L) + (1-y_i)\ln(1-a_i^L)]$$

$$\frac{\partial J}{\partial \theta_i} = \frac{1}{m}\sum_{i=1}^{m} x_i(g(z_i) - y_i)$$ Error in output control the learning

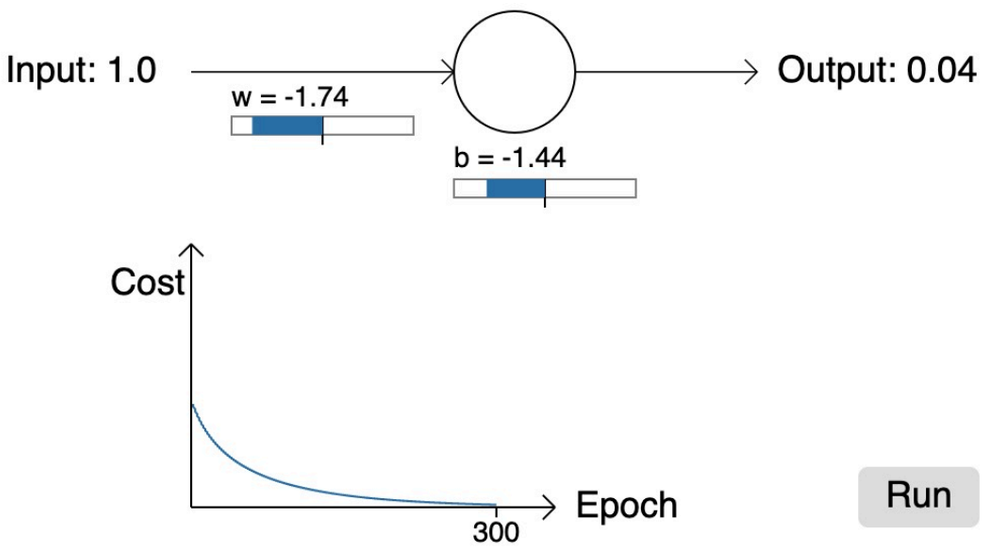
$$\frac{\partial J}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(g(z_i) - y_i)$$

(remember gradient in logistic regression ?)

Starting $w = 0.6, \quad b = 0.9$

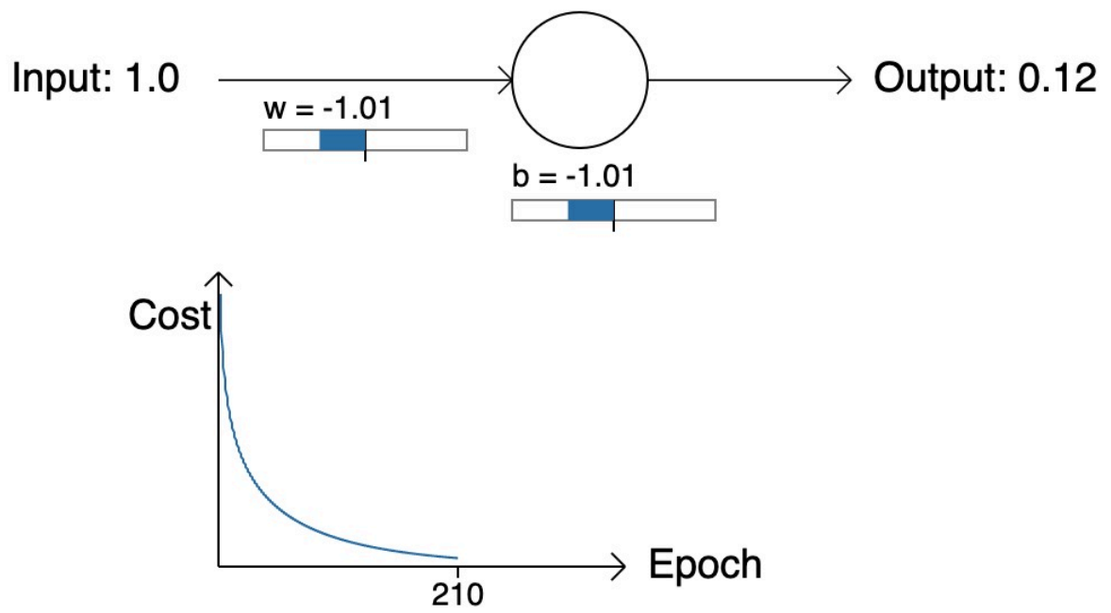
cross-entropy cost

input=1, expected output 0, initial output 0.82



Starting $w = 2, \quad b = 2$

input=1, expected output 0, initial output 0.98

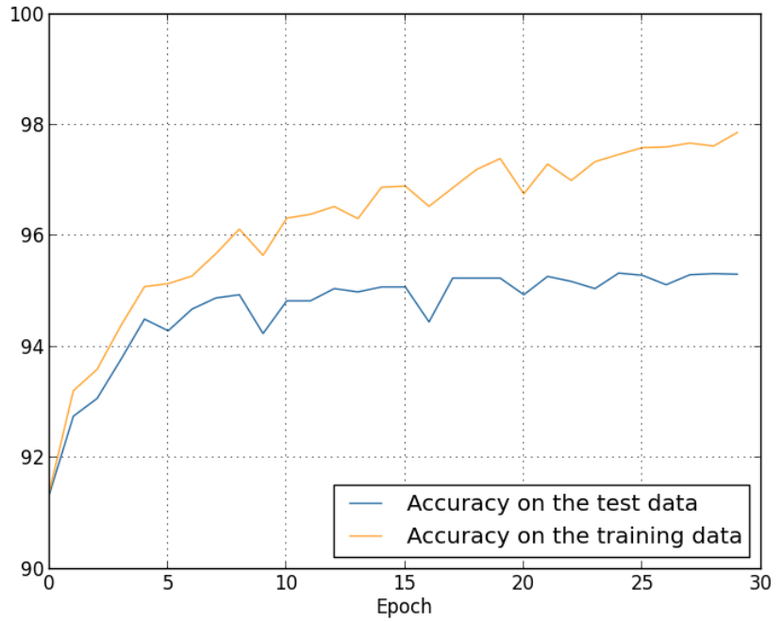


- when we use the quadratic cost learning is *slower* when the neuron is wrong than it is later on, as the neuron gets closer to the correct output

- while with the cross-entropy learning is faster when the neuron is wrong.

- from an information perspective, the cross-entropy measures how "surprised" we are, on average, when we learn the true value for y. We get low surprise if the output is what we expect, and high surprise if the output is unexpected

50000 training
10000 test
30 hidden neutrons, mini_batch_size=10 $\eta = 0.5$



Regularization

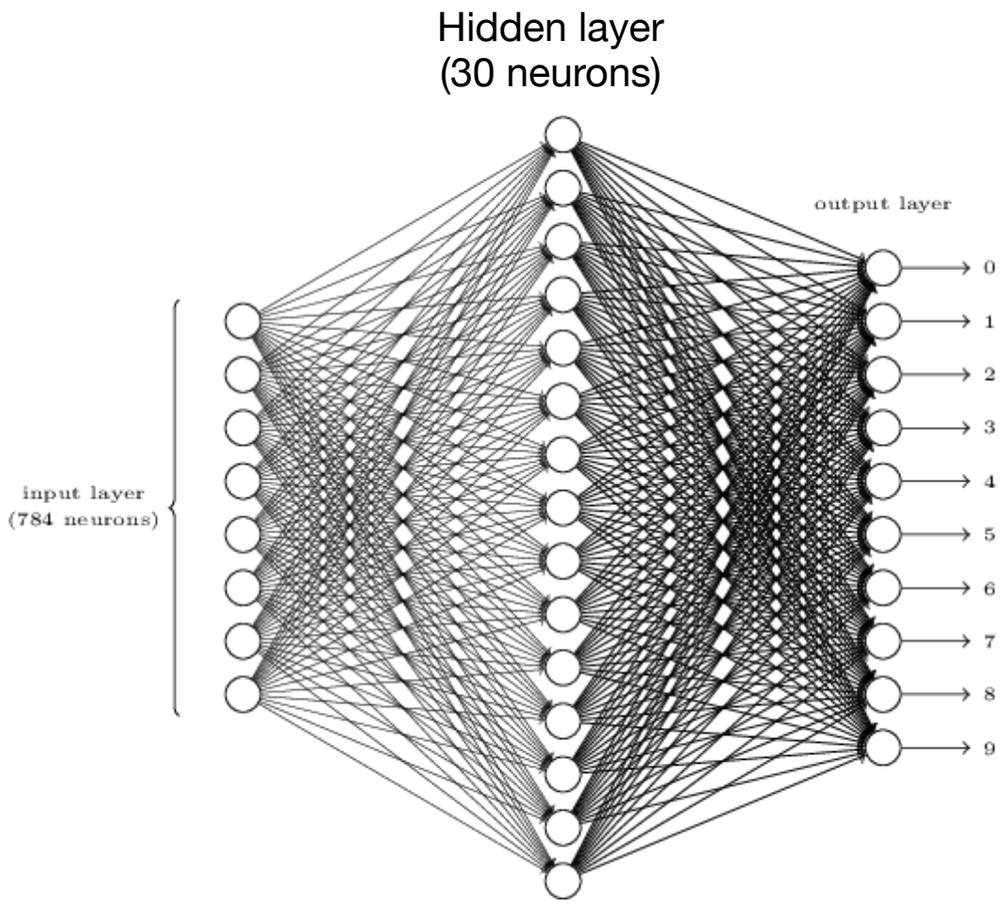
$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \ln(a_i^L) + (1-y_i)\ln(1-a_i^L)] + \frac{\lambda}{2m}\sum_{\Theta}\Theta^2$$

$$\Theta = (1-\frac{\eta\lambda}{m})\Theta - \frac{\eta}{m}\sum_{i}\frac{\partial J}{\partial \Theta}$$

http://neuralnetworksanddeeplearning.com/chap3.html

Hyperparameters: learning rate, mini-batch size

Requires thorough optimisation process

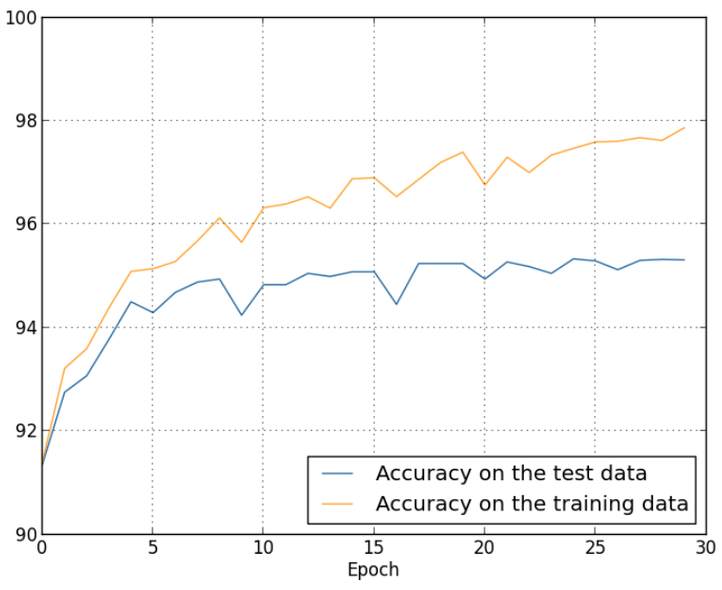


785x30 + 10x31 = 23,860

Fermi: "I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk.".

Training_data	validation_data	test_data
$\{\Theta_{ij}^{(l)}\}$	Hyperparameters	
50,000 images 30 epochs, mini-batch size 10	10,000 images	10,000 images

the validation data as a type of training data that helps us learn good hyper-parameters.



Without regularisation

With regularisation

50000 training
10000 test
30 hidden neutrons, mini_batch_size=10

$\eta = 0.5, \quad \lambda = 5$

- choice a better activation — softmax

Softmax activation — partition function $a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}$

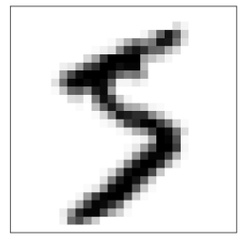
- Regularization methods — L1, L2, dropout,

$$J(\Theta) = J(\Theta) + \frac{\lambda}{m} \sum_{\Theta} |\Theta|$$

$$J(\Theta) = J(\Theta) + \frac{\lambda}{m} \sum_{\Theta} \Theta^2$$

- Jackknife resampling

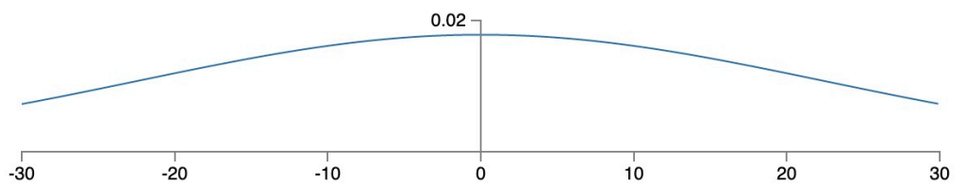
rotate/distort

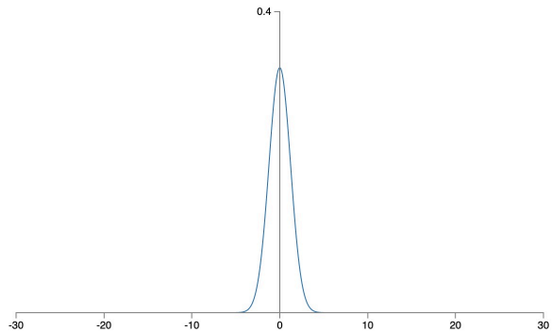


- A better initialisation of the weights — do not saturate the neutrons

$$z = \sum_i^{n_{in}} \Theta_i x_i + b \quad \Theta \in N(0,1) \quad z \gg 1 \quad \textbf{or} \quad z \ll -1$$

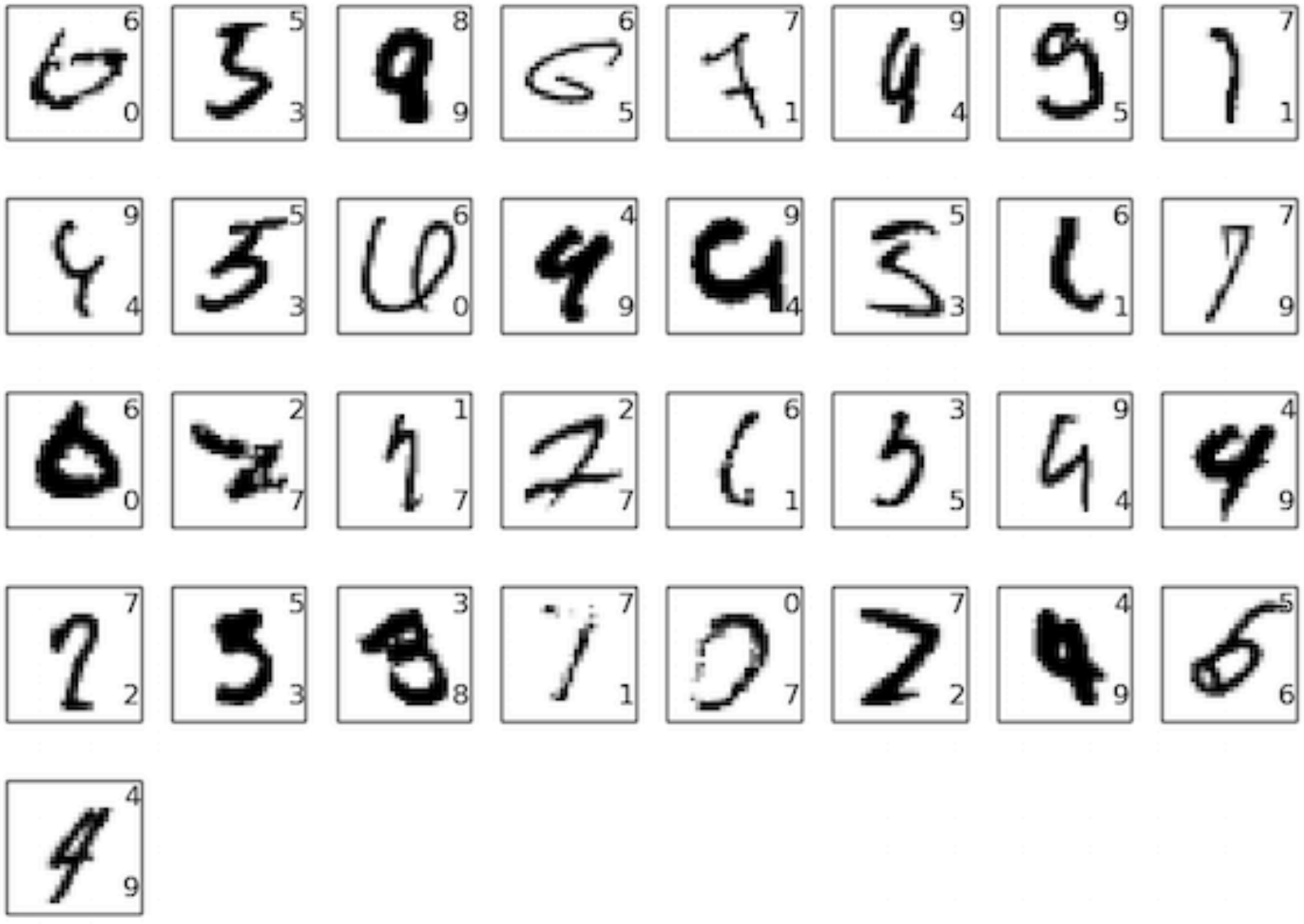
$$\Theta \in N(0, 1/\sqrt{n_{in}})$$





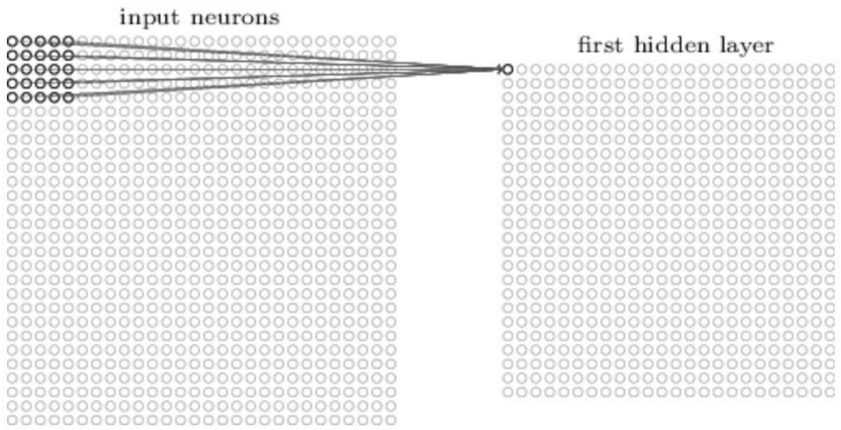
Deep learning - Convolutional neural network (CNN)

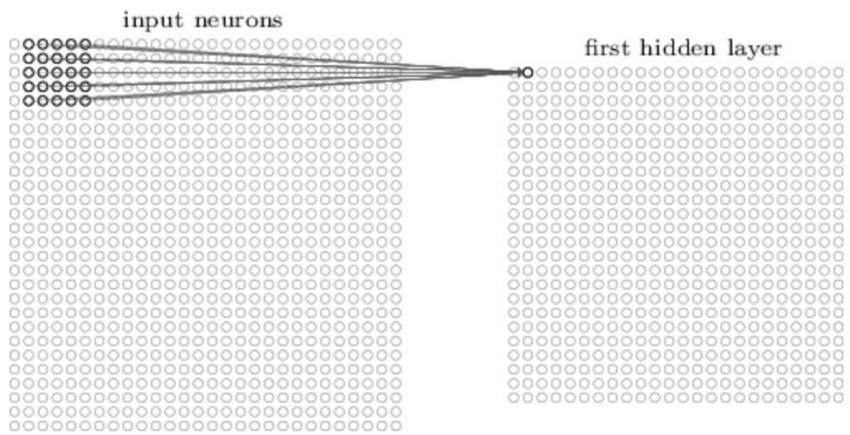
With CNN, 10,000 MNIST test images, one can classify 9,967 correctly. Accuracy 99.67% !!
Here's a peek at the 33 images which are misclassified. These are tough even for humans.



Deep learning - Convolutional neural network (CNN)

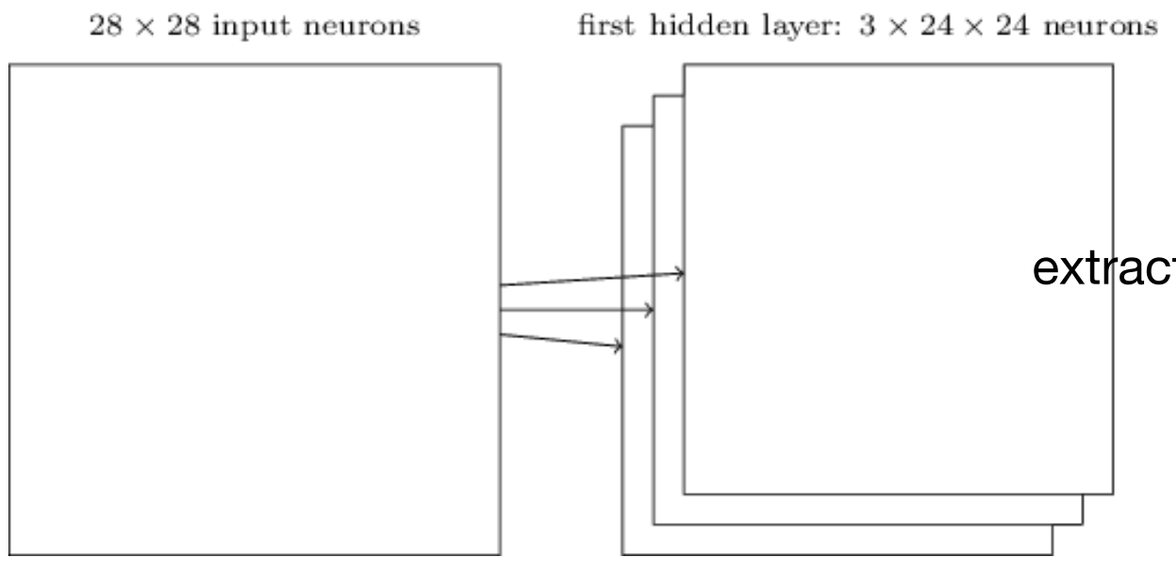
5×5 **local receptive fields** the same weights and bias $g(b + \sum_{l=0}^{4} \sum_{m=0}^{4} \Theta_{l,m} a_{j+l,k+m})$ $j = 1,2,\cdots,24$ $k = 1,2,\cdots,24$





Convolutional layer

More than one **feature map**, more than one set of shared weights+bias, more than one **filer**



extract translational invariant features

LeNet-5, 6 feature maps, 5x5 local receptive field

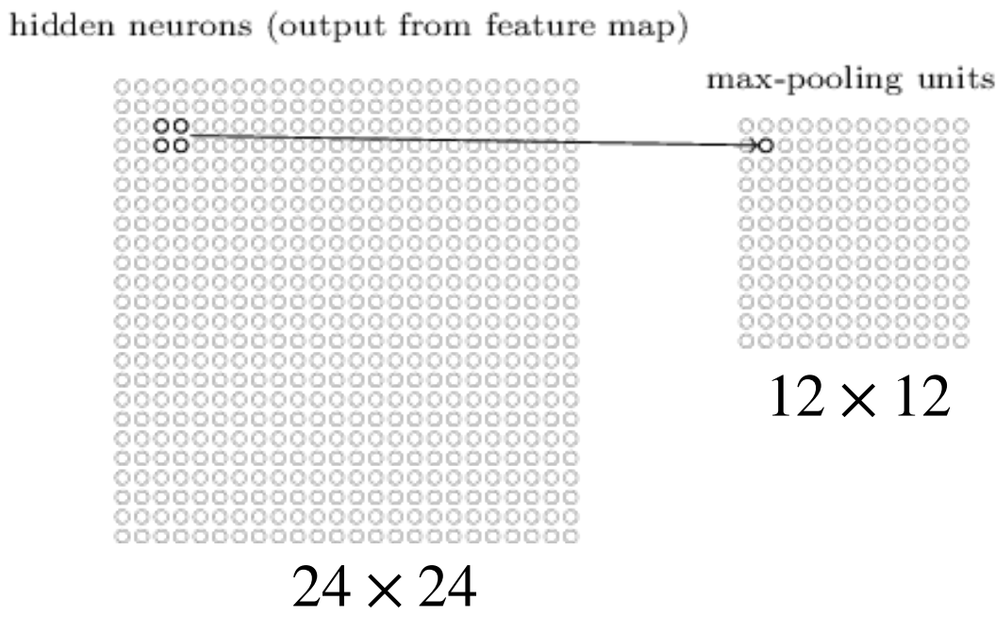
Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition"

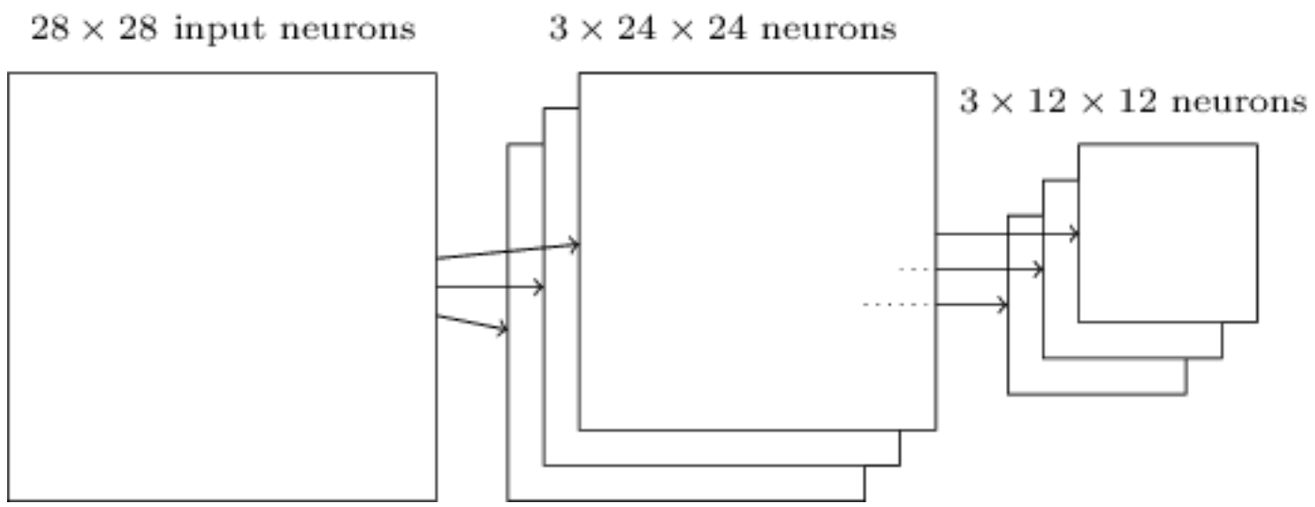
Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324 (1998)

Reduced number of parameters:
Each feature map 5x5 (weights) + 1 (bias)=26 parameter
20 feature maps 20x26 = 520 parameters in total

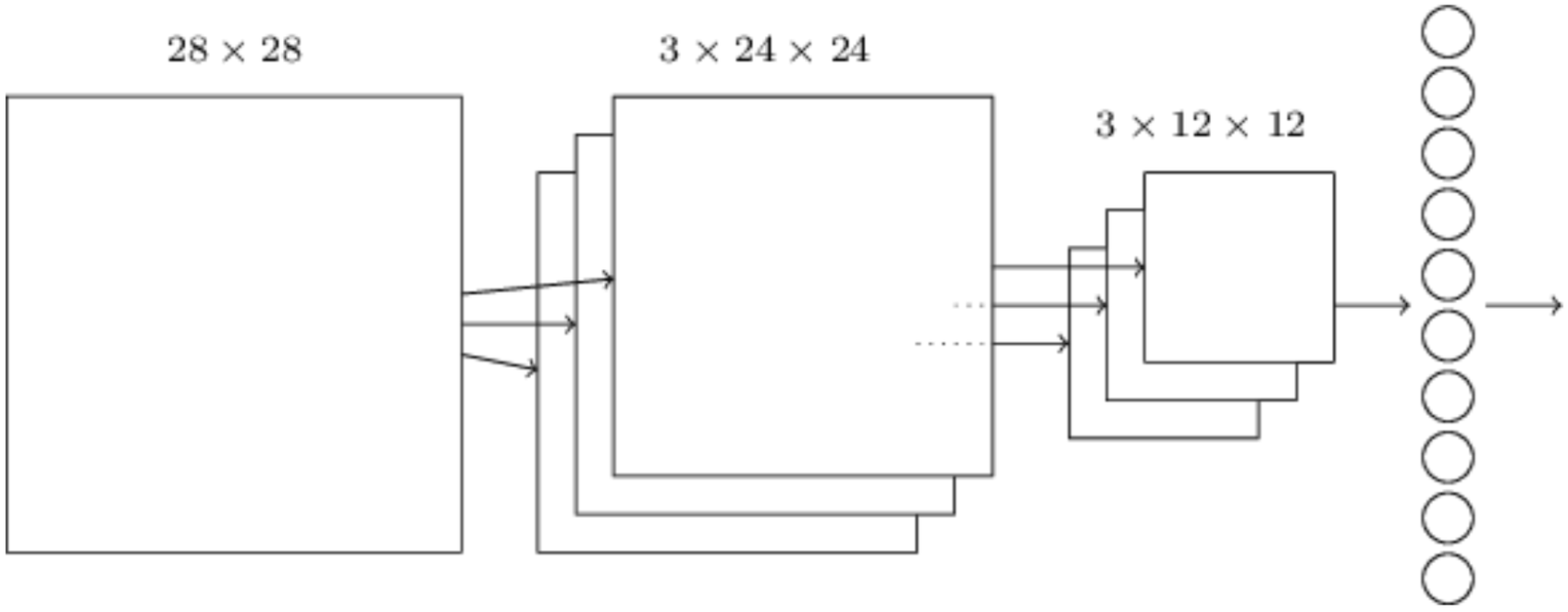
785x30 = 23550 fully-connected layer

Pooling layers max-pooling, take the maximum activation in the 2x2 input region





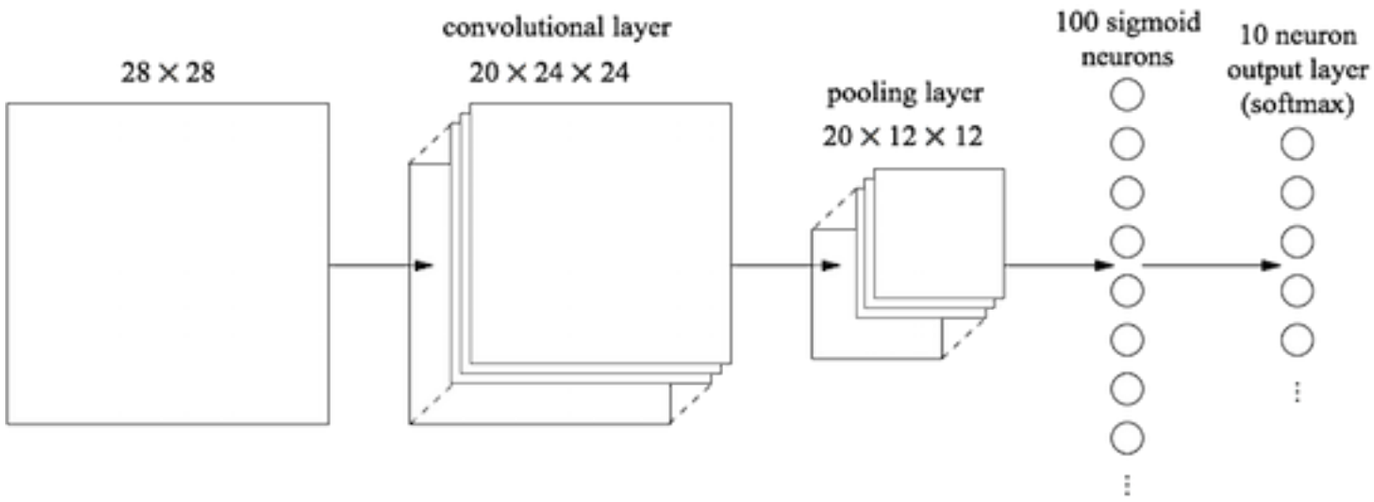
Input **feature map** **pooling** **fully-connected output**



3x26=78 10 x (3x12x12+1)=4330

Backpropagation for pooling and convolution layers

This architecture



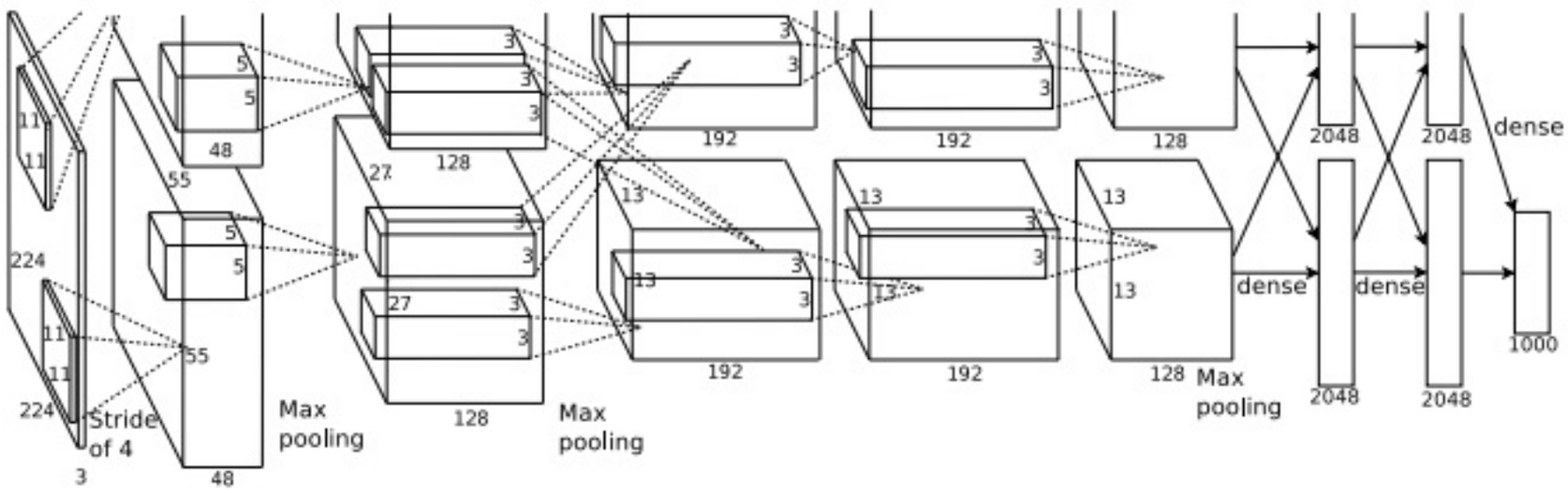
gives 98.78 % accuracy.

http://neuralnetworksanddeeplearning.com/chap6.html

- Expanding the training data
- Inserting extra fully-connected and extra convolution layers
- Using an ensemble of networks
- Using the right cost function to avoid learning slowdown
- Using good weight initialisation (also avoid the learning slowdonw)

ImageNet

GPU (an NVIDIA GeForce GTX 580) didn't have enough on-chip memory to store their entire network

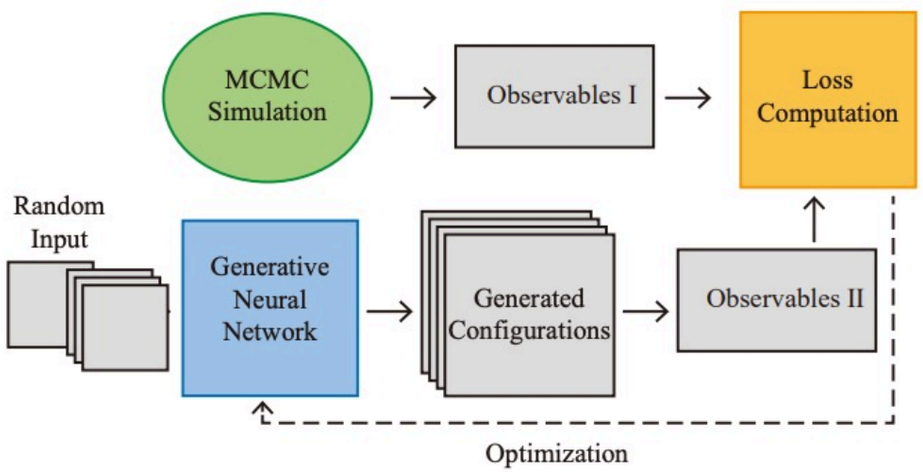


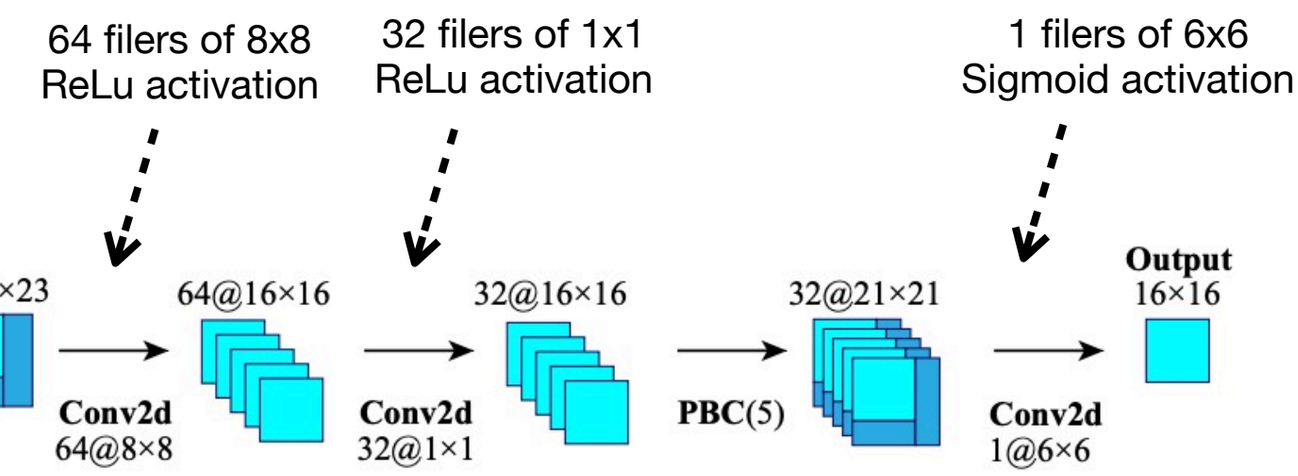
ImageNet classification with deep convolutional neural networks, Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton (2012).

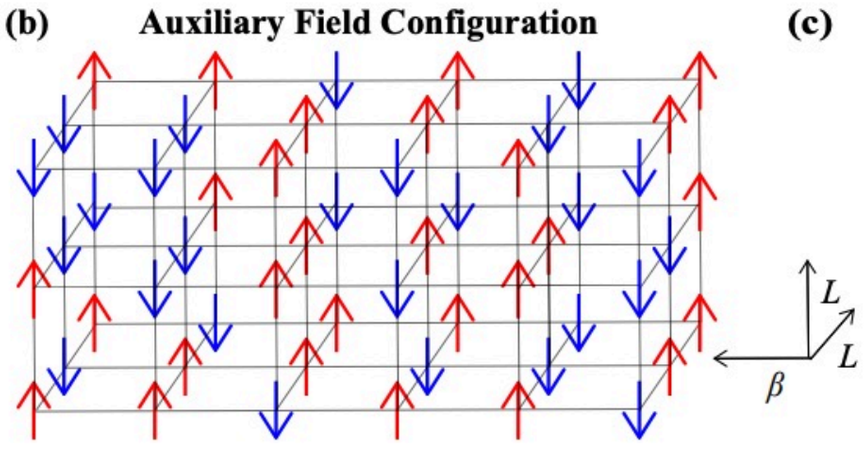
Chin. Phys. Lett. **39**, 050701 (2022)

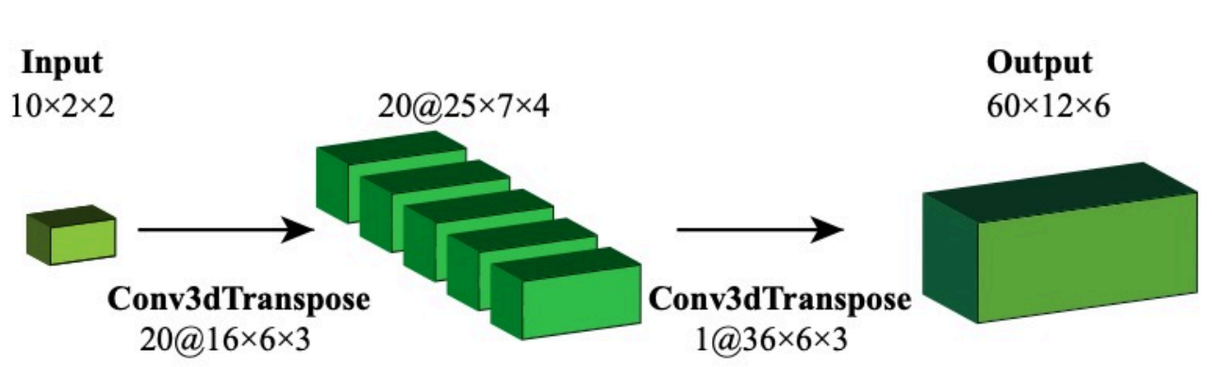
Network-Initialized Monte Carlo Based on Generative Neural Networks

Hongyu Lu, Chuhao Li, Bin-Bin Chen, …, ZYM





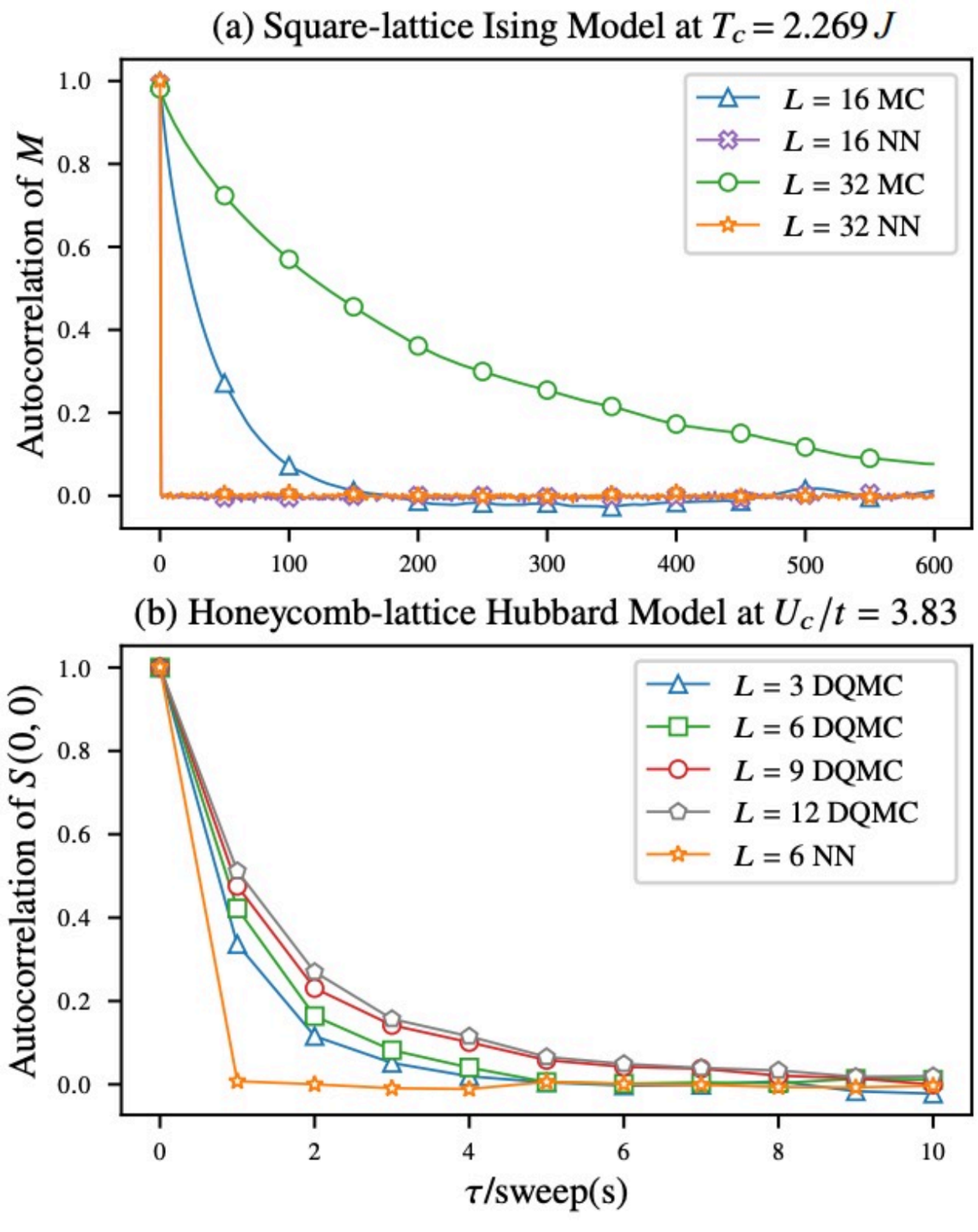


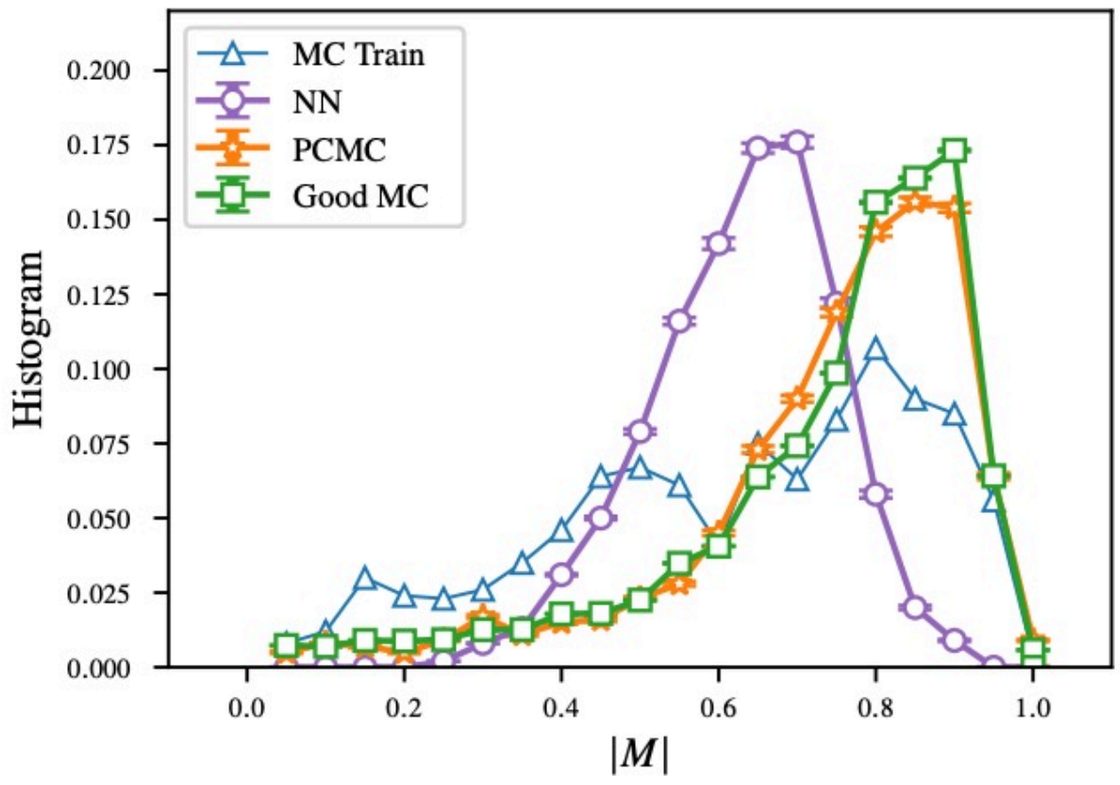


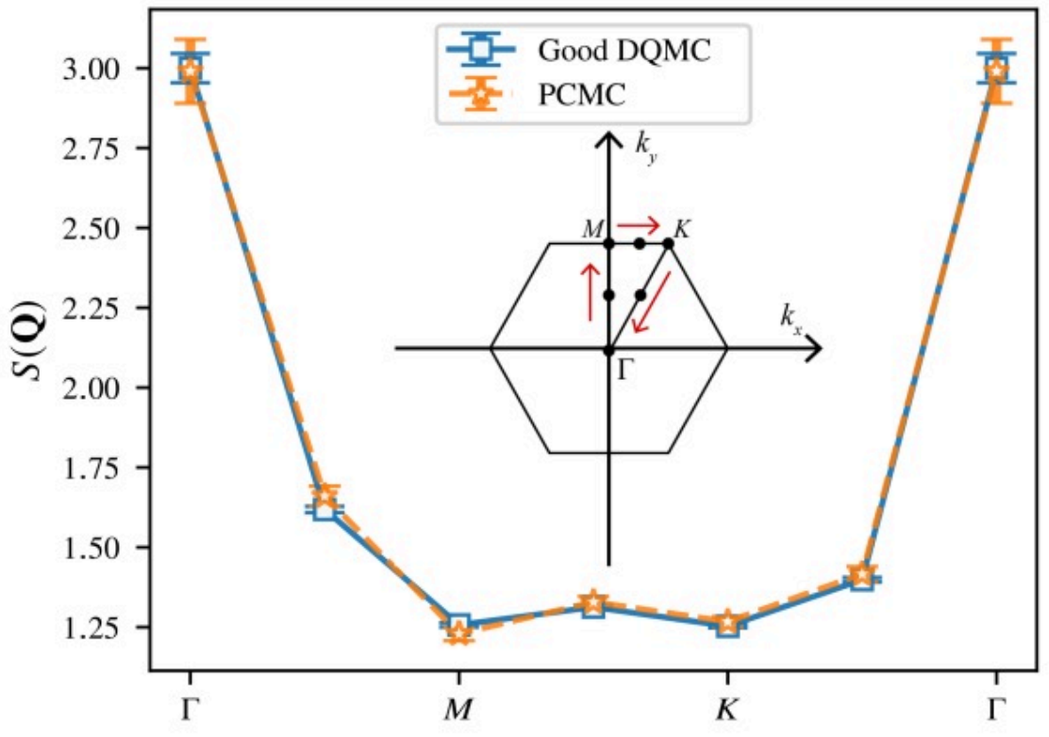
In order to optimize such neural networks, for both classical and quantum models, we prepare 1000 sets of observables measured from MC simulations and take 1000 input random configurations which will be processed into generated configurations. The comparison in loss function is randomly distributed without any grouping. For the choice of observables in the defined loss functions, we simply pick some from the ones we usually focus on. In the Ising case, we take the batch size to be 5 and epoch number to be up to 150 as the computation is quite easy. While in the Hubbard case, we run at most 15 epochs with a batch size of 3. We optimize the network parameters using Adam[63] with conventional learning rate 10^{-3}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

$$Loss(\boldsymbol{G}_l; M_l, E_l) = w_1 \sum_l [|M(\boldsymbol{G}_l)| - |M_l|]^2 + w_2 \sum_l [M^2(\boldsymbol{G}_l) - M_l^2]^2 + w_3 \sum_l [E(\boldsymbol{G}_l) - E_l]^2 \quad (2)$$

where $\boldsymbol{G}_l$ is the l-th generated configuration, M_l and E_l refer to the magnetization $\frac{1}{N}|\sum_j \sigma_j|$ and the energy density $\frac{1}{N}\langle H \rangle$




MC Train
NN
PCMC
Good MC
Histogram
|M|
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
0.0
0.2
0.4
0.6
0.8
1.0


Good DQMC
PCMC
S(Q)
Γ
M
K
Γ
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
k_y
k_x
M
K
Γ

Recurrent neural networks (RNNs)
Generative models

The ability to learn hierarchies of concepts,
building up multiple layers of abstraction,
seems to be fundamental to making sense of the world.

"basic research is what I'm doing when I don't know what I'm doing"

As the old joke goes, if you ask a scientist how far away some discovery is and they say "10 years" (or more), what they mean is "I've got no idea". AI, like controlled fusion and a few other technologies, has been 10 years away for 60 plus years.