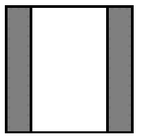
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

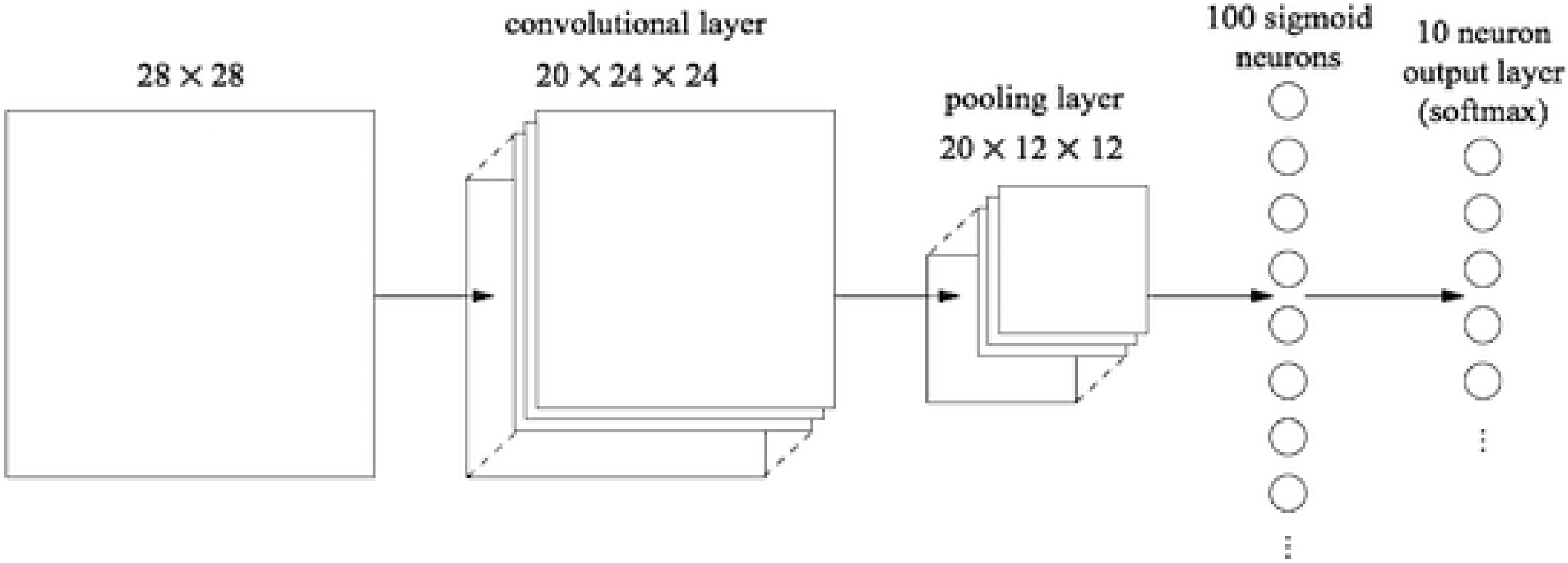
1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Conv Nets

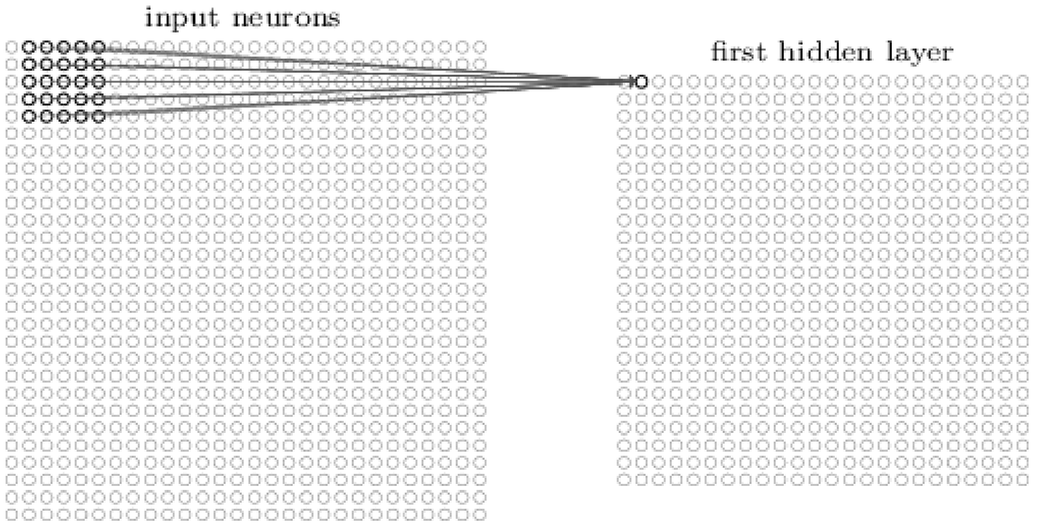

28 × 28
convolutional layer
20 × 24 × 24
pooling layer
20 × 12 × 12
100 sigmoid neurons
10 neuron output layer (softmax)

A Convolutional Neural Network (CNN) is a type of deep learning algorithm designed to analyze visual data, such as images or videos. It has proven to be highly effective in tasks like image classification, object detection, and image recognition.

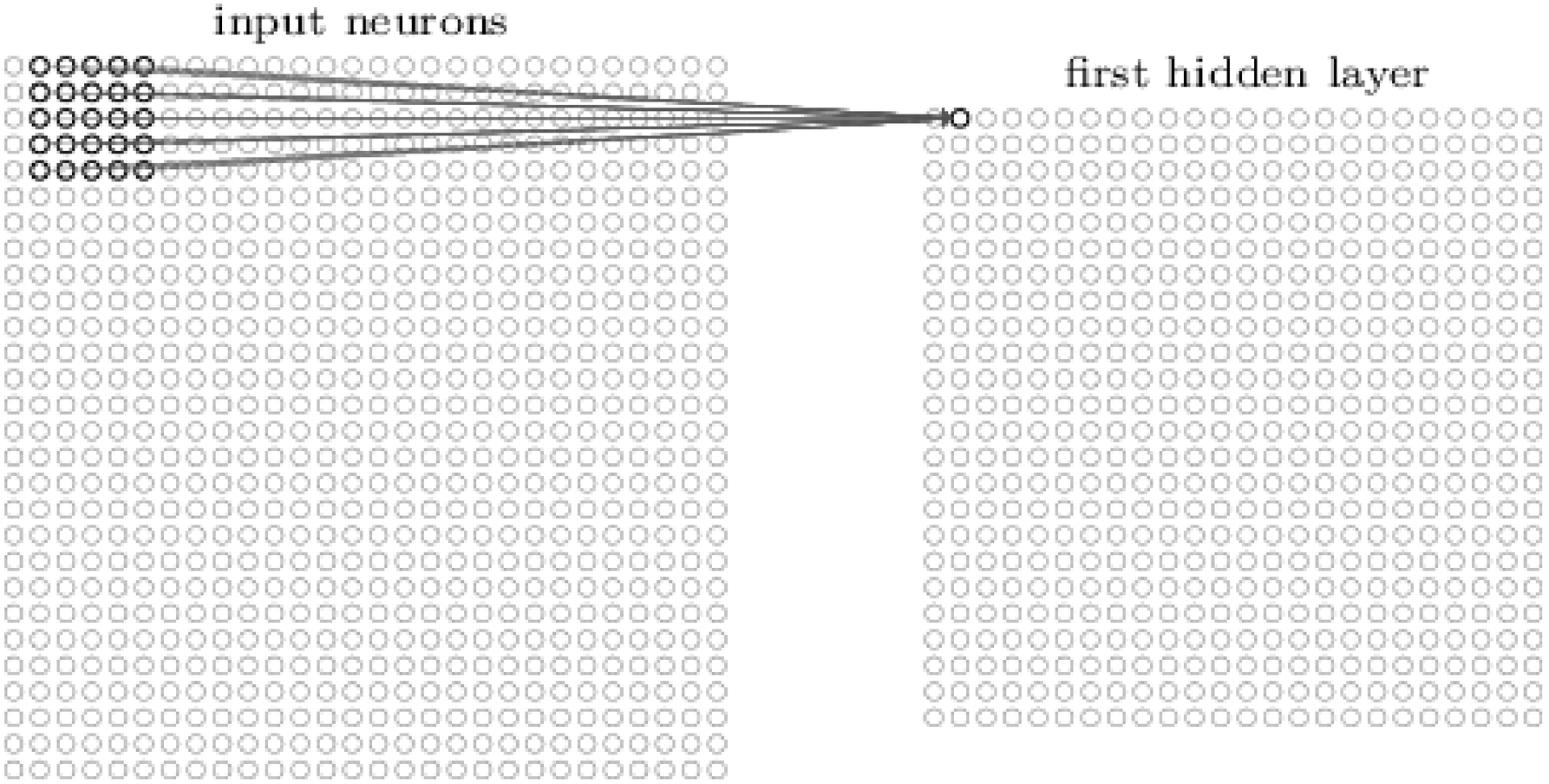
Convolutional neural networks use three basic ideas: local receptive fields, shared weights, and pooling. Let's look at each of these ideas in turn.

Local receptive fields: In the fully-connected layers shown earlier, the inputs were depicted as a vertical line of neurons. In a convolutional net, the inputs as a mxn matrix of neurons, where m,n are the size of the images.

Again in the FC layers, each hidden neuron was connected to each and every neuron in the previous layer. But in Convnet, we won't connect every input pixel to every hidden neuron. Instead, we only make connections in small, localized regions of the input image. To be more precise, each neuron in the first hidden layer will be connected to a small region of the input neurons, say, for example, a 5×5 region, corresponding to 25 input pixels. That region in the input image is called the local receptive field for the hidden neuron.



Then we slide the local receptive field over by one pixel to the right (i.e., by one neuron), to connect to a second hidden neuron:

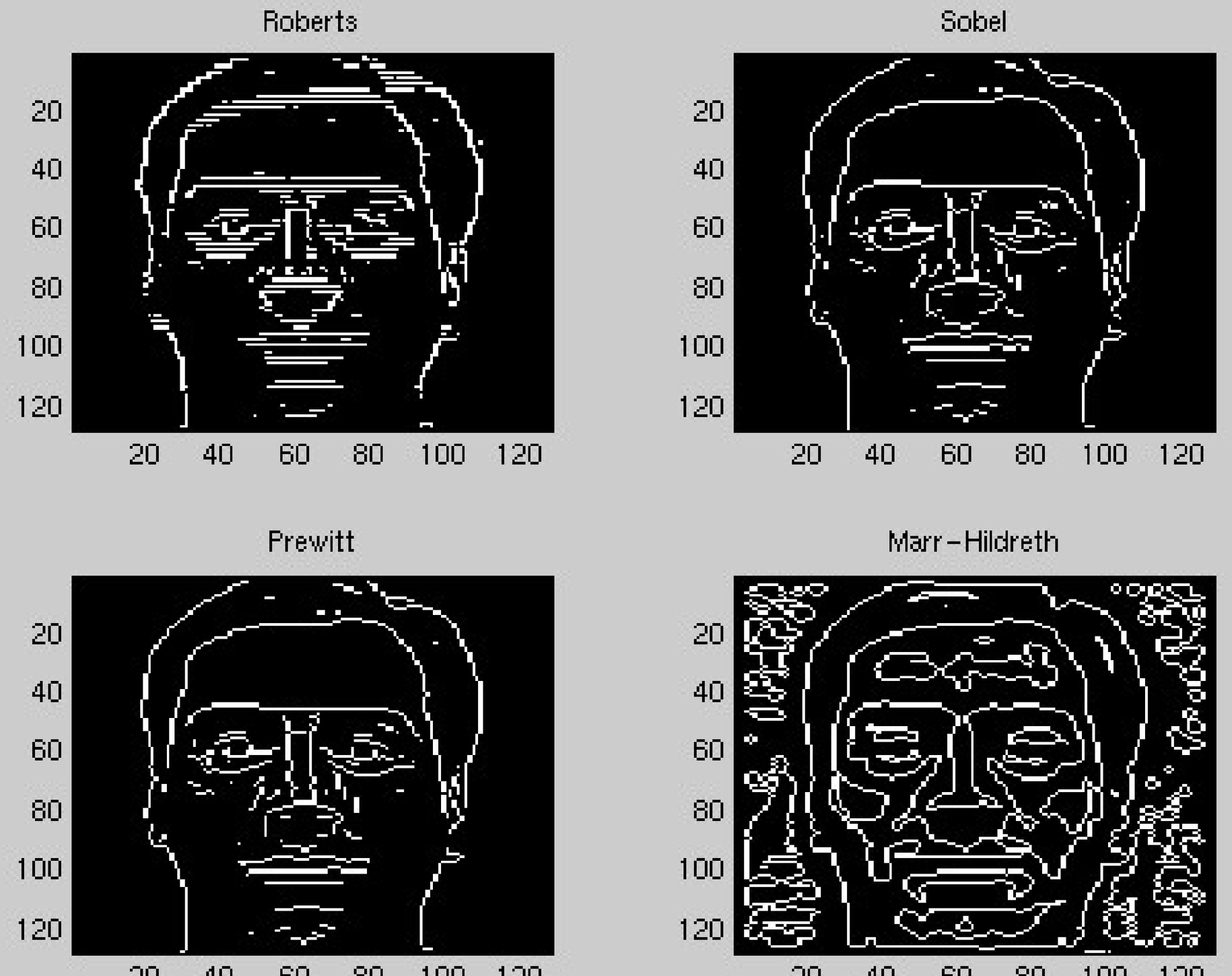


And so on, building up the first hidden layer. Note that if we have a 28×28 input image, and 5×5 local receptive fields, then there will be 24×24 neurons in the hidden layer.

Shared weights and biases: Each hidden neuron has a bias and 5×5 weights connected to its local receptive field. Moreover, we're going to use the same weights and bias for each of the 24×24 hidden neurons.

This means that all the neurons in the first hidden layer detect exactly the same feature, just at different locations in the input image. To see why this makes sense, suppose the weights and bias are such that the hidden neuron can pick out, say, a vertical edge in a particular local receptive field. That ability is also likely to be useful at other places in the image. And so it is useful to apply the same feature detector everywhere in the image. For this reason, we sometimes call the map from the input layer to the hidden layer a feature map. We call the weights defining the feature map the shared weights. And we call the bias defining the feature map in this way the shared bias. The shared weights and bias are often said to define a kernel or filter.

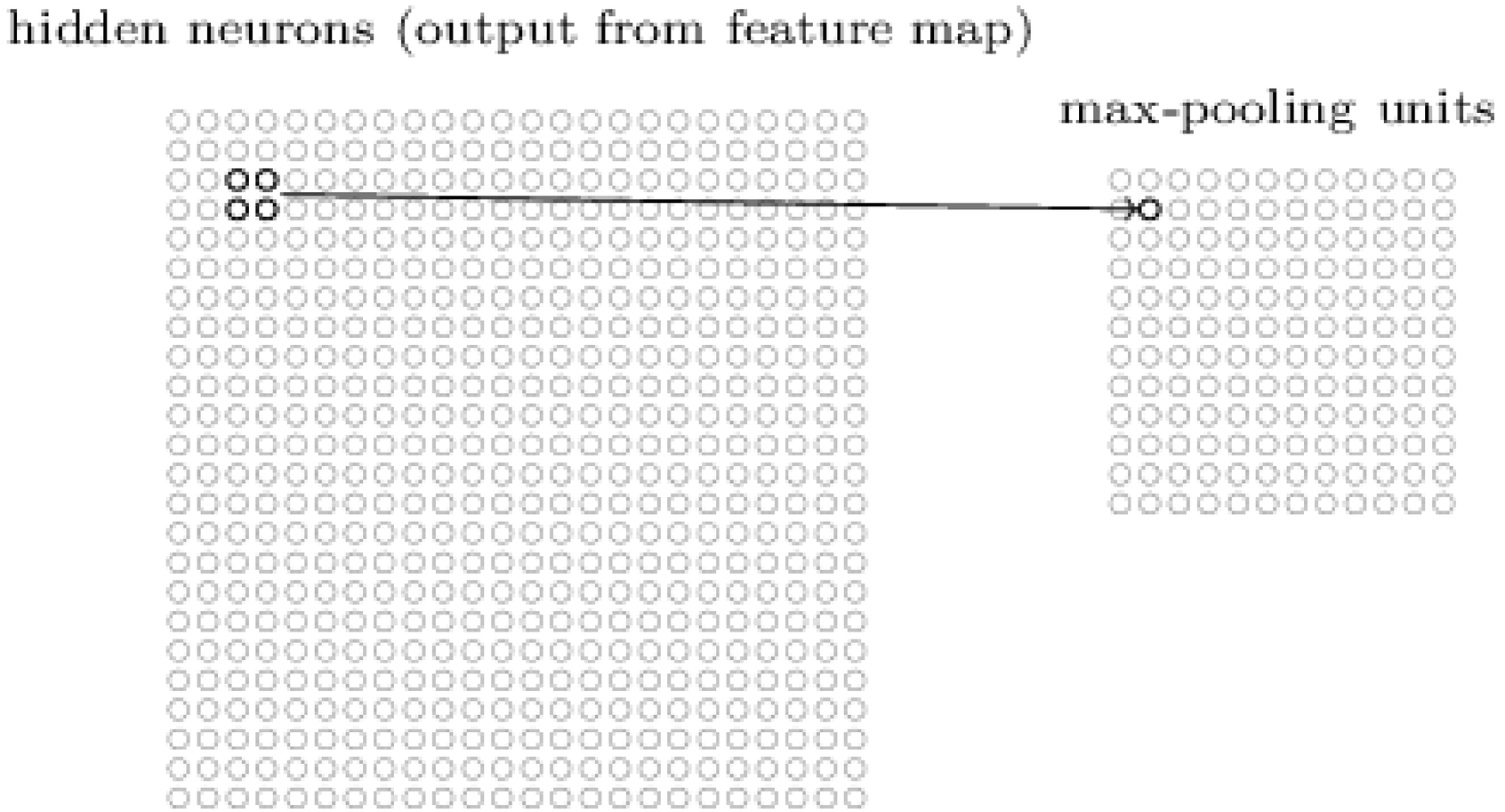
A big advantage of sharing weights and biases is that it greatly reduces the number of parameters involved in a convolutional network. For each feature map we need 25=5×5 shared weights, plus a single shared bias. So each feature map requires 26 parameters. If we have 20 feature maps that's a total of 20×26=520 parameters defining the convolutional layer. By comparison, suppose we had a fully connected first layer, with 784=28×28 input neurons, and a relatively modest 30 hidden neurons, as we used in many of the examples earlier in the book. That's a total of 784×30 weights, plus an extra 30 biases, for a total of 23,550 parameters. In other words, the fully-connected layer would have more than 40 times as many parameters as the convolutional layer.


Roberts
Sobel
Prewitt
Marr – Hildreth
20
40
60
80
100
120

Pooling layers: In addition to the convolutional layers just described, convolutional neural networks also contain pooling layers. Pooling layers are usually used immediately after convolutional layers. What the pooling layers do is simplify the information in the output from the convolutional layer.

In detail, a pooling layer takes each feature map, output from the convolutional layer and prepares a condensed feature map. For instance, each unit in the pooling layer may summarize a region of (say) 2×2 neurons in the previous layer. As a concrete example, one common procedure for pooling is known as max-pooling. In max-pooling, a pooling unit simply outputs the maximum activation in the 2×2 input region, as illustrated in the following diagram:

Max-pooling isn't the only technique used for pooling. Another common approach is known as L2 pooling. Here, instead of taking the maximum activation of a 2×22×2 region of neurons, we take the square root of the sum of the squares of the activations in the 2×22×2 region.



The handwritten digit classification task :

- The network begins with 28×28 input neurons, which are used to encode the pixel intensities for the MNIST image.
- This is then followed by a convolutional layer using a 5×5 local receptive field and 3 feature maps. The result is a layer of 3×24×24 hidden feature neurons.
- The next step is a max-pooling layer, applied to 2×2 regions, across each of the 3 feature maps. The result is a layer of 3×12×12 hidden feature neurons.

- The final layer of connections in the network is a fully-connected layer. That is, this layer connects every neuron from the max-pooled layer to every one of the 10 output neurons.

https://colab.research.google.com/drive/1Uu6rwo_f0PhKMIYODmMp2v--TGf94lFp?usp=sharing