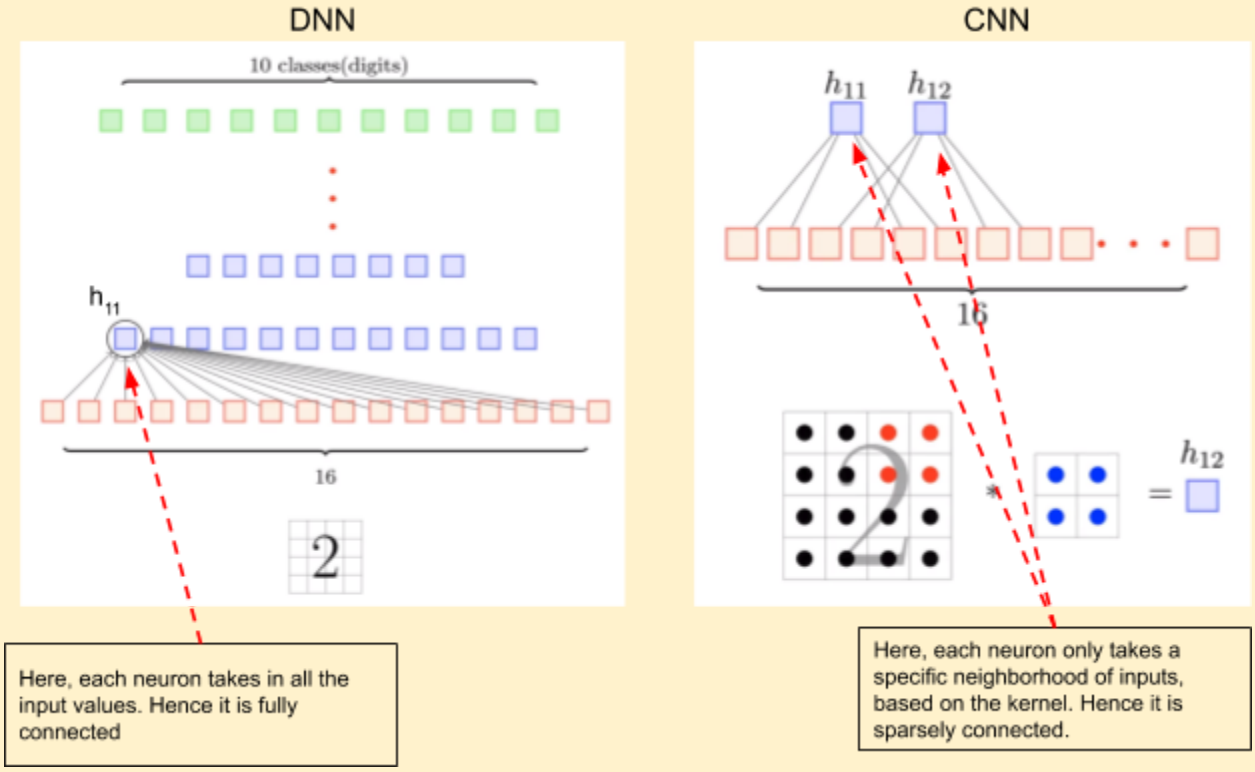
PadhAI: From Convolution Operation to Neural Network

One Fourth Labs

Sparse connectivity and Weight Sharing

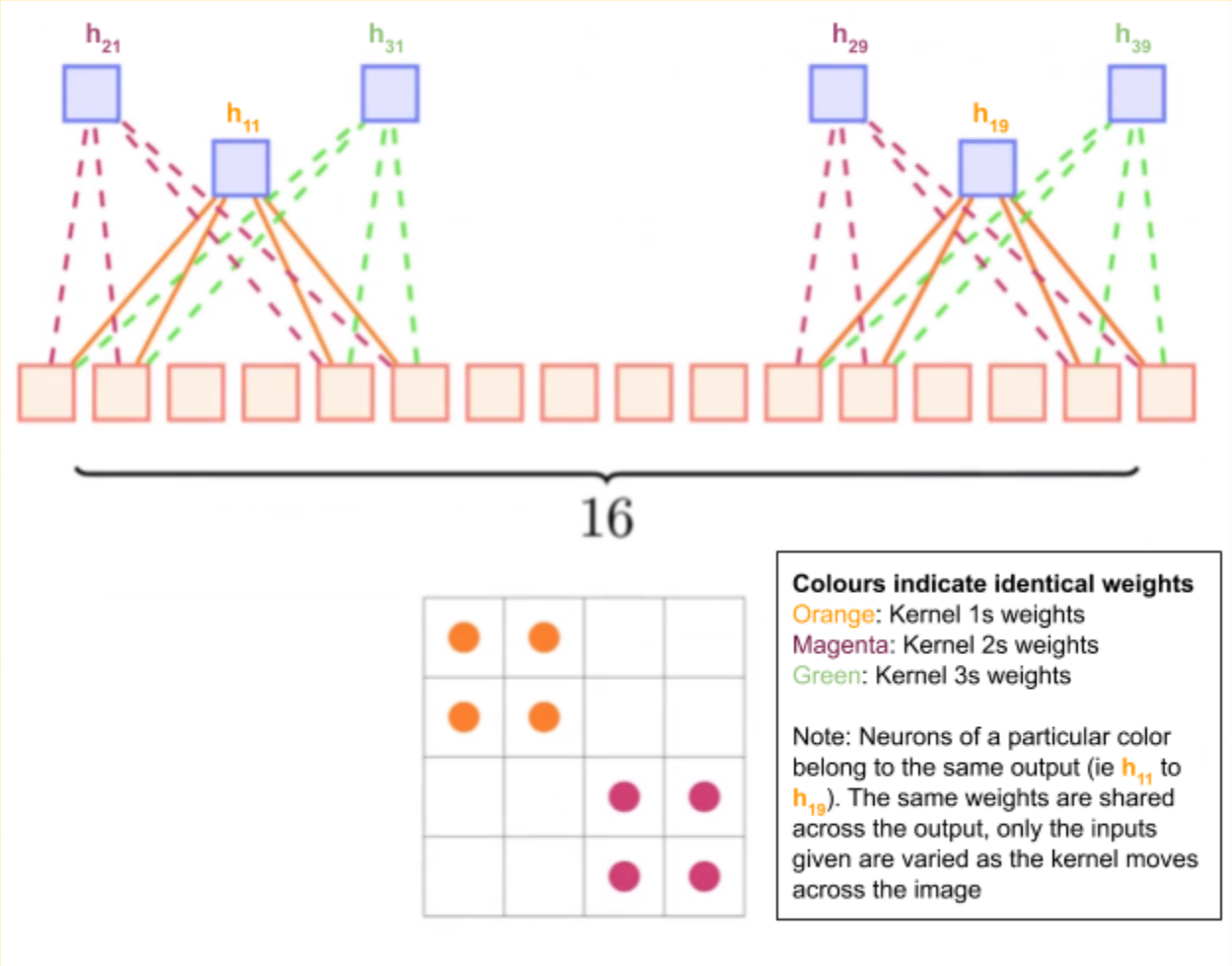
Let's look at the differences between Fully-connected DNNs and CNNs

1. We already have an intuition of how CNNs are different from DNNs, the key point being that in CNNs we take a weighted aggregation of a neighborhood of inputs instead of the all the inputs like in DNNs. Also, it is better to visualise CNNs in terms of matrices than by flat vectors as in the case of FFNs
2. More formally, two of the main aspects of CNNs are **Sparse Connectivity** & **Weight Sharing.**
3. Let us look at the **connectivity difference** between DNNs and CNNs



a. The above diagram is meant to illustrate the difference between DNNs and CNNs with regards to the input-output relationship
b. In CNNs, as the kernel moves over the input image in strides specified by the stride length S, it passes over the highlighted regions as shown in the figure. The highlighted regions vary in size with the Kernel size F, and they correspond with the input regions that are used to calculate the output for that particular neuron.
c. For the CNN, neuron h_{11} corresponds to inputs 1, 2, 5 & 6, while neuron h_{12} corresponds to inputs 3, 4, 7 & 8.
d. In contrast, in the DNN, neuron h_{11} and every other neuron corresponds to all inputs 1 to 16.
e. This is why DNNs are called fully connected and CNNs are said the be sparsely connected

4. Now, let us look at the concept of **weight sharing**, as illustrated in the figure.



a. Here, we can see how weights are shared across an an entire layer.
b. In CNNs, **weights are nothing but the kernel,** and the **kernel remains constant as we pass over the entire image**, creating different output values based on the neighborhood of inputs
c. One complete pass of the kernel over the image constitutes one Convolutional output.
d. As we have seen earlier, it is possible to have multiple convolutional operations by using multiple kernels over the input.
e. For each of these convoluted outputs, the kernel is constant, thereby **the weights get shared for all the neurons in that output area**. Only the inputs vary.
f. By combining all of these convolutional outputs, we get one convolutional layer
g. Instead of treating this Convolutional Layer as a flat vector, we treat it as a volume, whose depth is given by the number of kernels used to process the input.
h. By this practice, we are effectively reducing the number of parameters yet still retaining model complexity, thereby overcoming one of the shortcomings of DNNs.