## **Our First Convolutional Neural Network (CNN)**

How to use a convolutional neural network for image classification?

1. The following diagram illustrates the configuration and working of a Convolutional Neural Network. It follows the **LeNet** architecture, created by Yann LeCun
2. Let us sequentially break down the various layers in this CNN
3. **Input**:
   1. The image takes 32x32 pixel inputs.
   2. There is no depth component because the images are in black & white.
4. **Convolution Layer 1**:
   1. Here, the filter size F = 5, and the central cell is the pixel of interest
   2. Stride length S = 1
   3. We use a total of 6 filters, i.e. K = 6
   4. No padding is used, i.e. P = 0
   5. Each of the filters generate 28x28 output (calculated using WO, HO formula).
   6. Our hidden representation at this layer is **a1 = 28x28x6** (DO = K).
   7. Non-linearity like tanh or ReLU(preferred for CNN) is applied to **a1** making it **h1**
   8. If we were to proceed as a Fully Connected Network, we would have an extremely large number of parameters (32x32 x 28x28x6 = 4,816,896 parameters).
   9. However in this sparsely connected network, each of the 6 filters is of size 5x5x1. So the number of parameters would be much more manageable (6 x 5x5x1 = 150 parameters).
   10. This is significantly smaller than in a fully connected network, thereby reducing the chance of overfitting.
   11. Here, the values F, S, K, P etc are all counted as hyperparameters.
5. **Max Pooling Layer 1**:
   1. The hyperparameters are as follows
   2. Filter size F = 2
   3. Stride length S = 2
   4. No. of filters K = 6
   5. Padding P = 0
   6. Here, from a 2x2 filter, we select only 1 value. Therefore for a stride of 2, the output dimensions are half of the input(h1) dimensions, i.e 14x14
   7. We apply the max pooling independently to all 6 of the h1 layers, giving us **h2 = 14x14x6**
   8. No parameters for this layer as we are simply choosing the largest value in the filter and not applying any weights to it.
6. **Convolutional Layer 2**:
   1. The hyperparameters are as follows
   2. Filter size F = 5
   3. Stride length S = 1
   4. No. of filters K = 16
   5. Padding P = 0
   6. Thus, the filter dimensions are 5x5x6
   7. Here, 16 filters are applied to the input h2, thereby giving us an output depth of DO = 16
   8. Calculating WO and HO using the formula, we get 10x10
   9. Our hidden representation at this layer is **a3 = 10x10x16**
   10. Non-linearity like tanh or ReLU(preferred for CNN) is applied to **a3** making it **h3**
   11. The number of parameters for the filters(16 x 5x5x6) is 2400 parameters
   12. This is much smaller than what we would have had in a fully connected network
7. **Max Pooling Layer 2**:
   1. The hyperparameters are as follows
   2. Filter size F = 2
   3. Stride length S = 1
   4. No. of filters K = 16
   5. Padding P = 0
   6. Here, from a 2x2 filter, we select only 1 value. Therefore for a stride of 2, the output dimensions are half of the input(h3) dimensions, i.e 14x14
   7. We apply the max pooling independently to all 16 of the h1 layers, giving us **h4 = 5x5x16**
   8. No params for this layer as we are simply choosing the largest value in the filter.
8. **Fully connected layer 1**:
   1. Number of neurons: 120
   2. Input is **h4** flattened, i.e. 5x5x16 = 400
   3. No. of parameters in **h5** = 120x400 + 120-bias = 48120 parameters
9. **Fully connected layer 2**:
   1. Number of neurons: 84
   2. Input is number of neurons in **h5** = 120
   3. No. of parameters in **h6** = 84x120 + 84-bias = 10164 parameters
10. **Output layer**:
    1. Number of neurons: 10
    2. Input is number of neurons in **h6** = 84
    3. No. of parameters in **y** = 10x84 + 10-bias = 850 parameters
11. Overall, this combination of Convolutional and fully-connected layers is much more efficient than an entirely fully connected network. It has a significantly lower number of parameters but still is able to estimate functions of very high complexity.