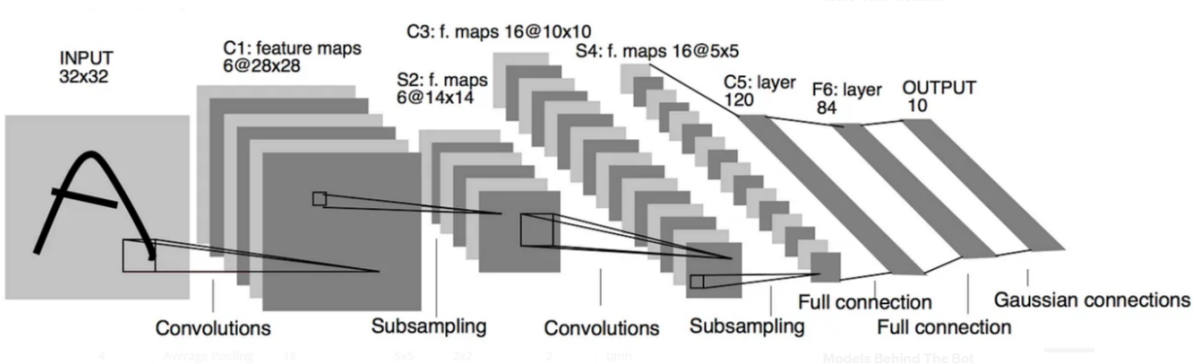
**4.11 Modern deep learning Architectures**

**1. LeNet-5 Architecture**




In 1989, Yann LeCun presented a convolutional neural network named LeNet. In general, LeNet refers to LeNet-5 and is a straightforward convolutional neural network.
LeNet-5 CNN architecture is made up of 7 layers. The layer composition consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers.

**Features of LeNet-5**

- Every convolutional layer includes three parts: convolution, pooling, and nonlinear activation functions
- Using convolution to extract spatial features (Convolution was called receptive fields originally)
- Subsampling average pooling layer
- tanh activation function
- Using MLP as the last classifier
- Sparse connection between layers to reduce the complexity of computation

  **Layer 1-** The first layer is the input layer; It is generally not considered a layer of the network as nothing is learned on that layer. The input layer supports 32x32, and these are the dimensions of the images that will be passed to the next layer.

  Those familiar with the MNIST dataset will know that the images in the MNIST dataset are 28 x 28 in dimensions. In order for the dimension of MNIST images to meet the requirements of the input layer, the 28x28.

  **Layer 2-** Layer C1 is a convolution layer with six 5 × 5 convolution kernels, and the feature allocation size is 28 × 28, whereby input image information can be avoided.

  **Layer 3**- Layer S2 is the undersampling / grouping layer which generates 6 function graphs of length 14x14. Each cell in every function map is attached to 2x2 neighborhoods at the corresponding function map in C1.

  **Layer 4**- C3 convolution layer encompass sixteen 5x5 convolution kernels The input of the primary six function maps C3 is every continuous subset of the 3 function maps in S2, the

access of the following six function maps comes from the access of the 4 continuous subsets and the input for the following 3 function maps is crafted from the 4 discontinuous subsets.Finally, the input for the very last function diagram comes from all the S2 function diagrams.

**Layer 5-** Layer S4 is just like S2 with a length of 2x2 and an output of sixteen 5x5 function graphics.

**Layer 6-** Layer C5 is a convolution layer with one hundred twenty convolution cores of length 5x5. Each cell is attached to the 5x5 neighborhoods along sixteen S4 function charts. Since the function chart length of S4 is likewise 5x5, the output length of C5 is 1 * 1, so S4 and C5 are absolutely linked.
It is referred to as a convolutional layer in preference to a completely linked layer due to the fact if the input of LeNet-5 becomes large and its shape stays unchanged, then its output length is bigger than 1x1, i.e. now no longer a completely linked layer.

**Layer 7-** The F6 layer is connected to C5 and 84 feature charts are generated. In the grayscale images used in the research, the pixel values from 0 to 255 were normalized to values between -0.1 and 1,175 The reason for normalization is to make sure the image stack has a mean of 0 and a standard deviation of 1.

The advantages of this are in the reduction of the training time. In the following example we will normalize the pixel values of the images to take values between 0 and 1.

| Layer | # filters / neurons | Filter size | Stride | Size of feature map | Activation function |
|---|---|---|---|---|---|
| Input | - | - | - | 32 X 32 X 1 | |
| Conv 1 | 6 | 5 * 5 | 1 | 28 X 28 X 6 | tanh |
| Avg. pooling 1 | | 2 * 2 | 2 | 14 X 14 X 6 | |
| Conv 2 | 16 | 5 * 5 | 1 | 10 X 10 X 16 | tanh |
| Avg. pooling 2 | | 2 * 2 | 2 | 5 X 5 X 16 | |
| Conv 3 | 120 | 5 * 5 | 1 | 120 | tanh |
| Fully Connected 1 | - | - | - | 84 | tanh |
| Fully Connected 2 | - | - | - | 10 | Softmax |

1. The first layer is the input layer with feature map size 32X32X1.

2. Then we have the first convolution layer with 6 filters of size 5X5 and stride is 1. The activation function used at his layer is tanh. The output feature map is 28X28X6.

3. Next, we have an average pooling layer with filter size 2X2 and stride 1. The resulting feature map is 14X14X6. Since the pooling layer doesn't affect the number of channels.

4. After this comes the second convolution layer with 16 filters of 5X5 and stride 1. Also, the activation function is tanh. Now the output size is 10X10X16.

5. Again comes the other average pooling layer of 2X2 with stride 2. As a result, the size of the feature map reduced to 5X5X16.

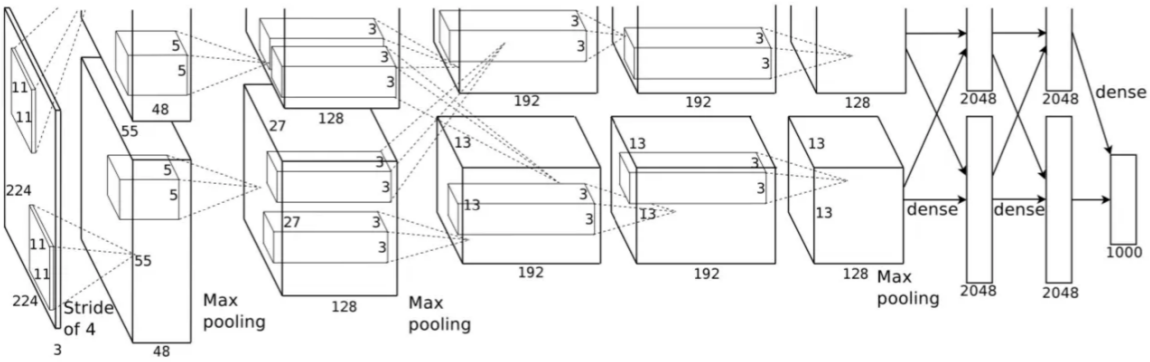
6. The final pooling layer has 120 filters of 5X5 with stride 1 and activation function tanh. Now the output size is 120.

7. The next is a fully connected layer with 84 neurons that result in the output to 84 values and the activation function used here is again tanh.

8. The last layer is the output layer with 10 neurons and Softmax function. The Softmax gives the probability that a data point belongs to a particular class. The highest value is then predicted.

9. This is the entire architecture of the Lenet-5 model. The number of trainable parameters of this architecture is around sixty thousand.

**2. AlexNet:**




The architecture consists of eight layers: five convolutional layers and three fully-connected layers. But this isn't what makes AlexNet special; these are some of the features used that are new approaches to convolutional neural networks:

**ReLU Nonlinearity:** AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function, which was standard at the time. ReLU's advantage is in training time; a CNN using ReLU was able to reach a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh.

**Multiple GPUs:** Back in the day, GPUs were still rolling around with 3 gigabytes of memory (nowadays those kinds of memory would be rookie numbers). This was especially bad because the training set had 1.2 million images. AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time.

**Overlapping Pooling:** CNNs traditionally "pool" outputs of neighboring groups of neurons with no overlapping. However, when the authors introduced overlap, they saw a reduction in error by about 0.5% and found that models with overlapping pooling generally find it harder to overfit.

**The Overfitting Problem**. AlexNet had 60 million parameters, a major issue in terms of overfitting. Two methods were employed to reduce overfitting:

**Data Augmentation:** The authors used label-preserving transformation to make their data more varied. Specifically, they generated image translations and horizontal reflections, which increased the training set by a factor of 2048. They also performed Principle Component Analysis (PCA) on the RGB pixel values to change the intensities of RGB channels, which reduced the top-1 error rate by more than 1%.

**Dropout**: This technique consists of "turning off" neurons with a predetermined probability (e.g. 50%). This means that every iteration uses a different sample of the model's parameters, which forces each neuron to have more robust features that can be used with other random neurons. However, dropout also increases the training time needed for the model's convergence.

| Layer | # filters / neurons | Filter size | Stride | Padding | Size of feature map | Activation function |
|---|---|---|---|---|---|---|
| Input | - | - | - | - | 227 x 227 x 3 | - |
| Conv 1 | 96 | 11 x 11 | 4 | - | 55 x 55 x 96 | ReLU |
| Max Pool 1 | - | 3 x 3 | 2 | - | 27 x 27 x 96 | - |
| Conv 2 | 256 | 5 x 5 | 1 | 2 | 27 x 27 x 256 | ReLU |
| Max Pool 2 | - | 3 x 3 | 2 | - | 13 x 13 x 256 | - |
| Conv 3 | 384 | 3 x 3 | 1 | 1 | 13 x 13 x 384 | ReLU |
| Conv 4 | 384 | 3 x 3 | 1 | 1 | 13 x 13 x 384 | ReLU |
| Conv 5 | 256 | 3 x 3 | 1 | 1 | 13 x 13 x 256 | ReLU |
| Max Pool 3 | - | 3 x 3 | 2 | - | 6 x 6 x 256 | - |
| Dropout 1 | rate = 0.5 | - | - | - | 6 x 6 x 256 | - |

**Convolution and Maxpooling Layers**

1. The input to this model is the images of size 227X227X3.
2. Then we apply the first convolution layer with 96 filters of size 11X11 with stride 4. The activation function used in this layer is relu. The output feature map is 55X55X96.
3. To calculate the output size of a convolution layer,
   output= ((Input-filter size)/ stride)+1
4. Also, the number of filters becomes the channel in the output feature map.
5. Next, we have the first Maxpooling layer, of size 3X3 and stride 2. Then we get the resulting feature map with the size 27X27X96.
6. After this, we apply the second convolution operation. This time the filter size is reduced to 5X5 and we have 256 such filters. The stride is 1 and padding 2. The activation function used is again relu. Now the output size we get is 27X27X256.
7. Again we applied a max-pooling layer of size 3X3 with stride 2. The resulting feature map is of shape 13X13X256.
8. Now we apply the third convolution operation with 384 filters of size 3X3 stride 1 and also padding 1. Again the activation function used is relu. The output feature map is of shape 13X13X384.

9. Then we have the fourth convolution operation with 384 filters of size 3X3. The stride along with the padding is 1. On top of that activation function used is relu. Now the output size remains unchanged i.e 13X13X384.
10. After this, we have the final convolution layer of size 3X3 with 256 such filters. The stride and padding are set to one also the activation function is relu. The resulting feature map is of shape 13X13X256.
11. So if you look at the architecture till now, the number of filters is increasing as we are going deeper. Hence it is extracting more features as we move deeper into the architecture. Also, the filter size is reducing, which means the initial filter was larger and as we go ahead the filter size is decreasing, resulting in a decrease in the feature map shape.
12. Next, we apply the third max-pooling layer of size 3X3 and stride 2. Resulting in the feature map of the shape 6X6X256.

**Fully Connected and Dropout Layers**

| Layer | # filters / neurons | Filter size | Stride | Padding | Size of feature map | Activation function |
|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| - | - | - | - | - | - | - |
| Dropout 1 | rate = 0.5 | - | - | - | 6 x 6 x 256 | - |
| Fully Connected 1 | - | - | - | - | 4096 | ReLU |
| Dropout 2 | rate = 0.5 | - | - | - | 4096 | - |
| Fully Connected 2 | - | - | - | - | 4096 | ReLU |
| Fully Connected 3 | - | - | - | - | 1000 | Softmax |

1. we have our first dropout layer. The drop-out rate is set to be 0.5.

2. Then we have the first fully connected layer with a relu activation function. The size of the output is 4096. Next comes another dropout layer with the dropout rate fixed at 0.5.

3. This followed by a second fully connected layer with 4096 neurons and relu activation.

4. Finally, we have the last fully connected layer or output layer with 1000 neurons as we have 10000 classes in the data set. The activation function used at this layer is Softmax.

5. This is the architecture of the Alexnet model. It has a total of 62.3 million learnable parameters.