**Lenet-5**

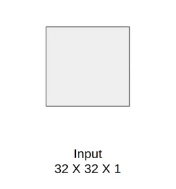
***What is Lenet5?***

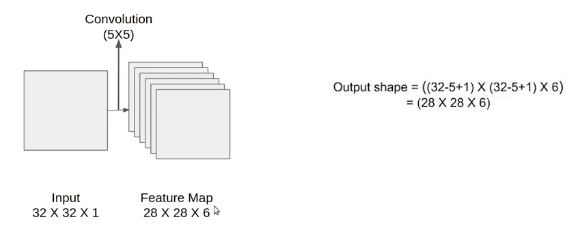
Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters.

The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification.

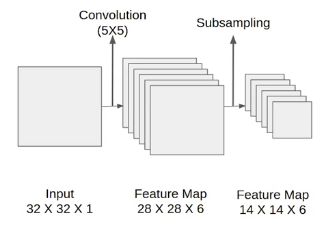
***The Architecture of the Model***

The network has 5 layers with learnable parameters and hence named Lenet-5. It has three sets of convolution layers with a combination of average pooling. After the convolution and average pooling layers, we have two fully connected layers. At last, a Softmax classifier which classifies the images into respective class.

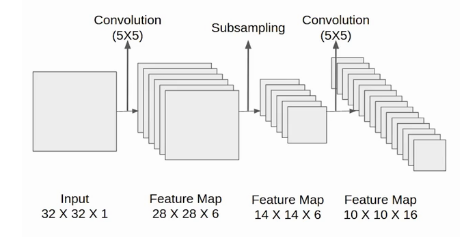


The input to this model is a 32 X 32 grayscale image hence the number of channels is one. 

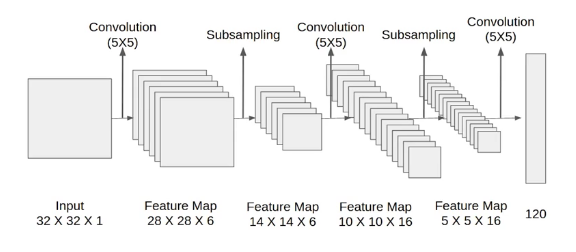
We then apply the first convolution operation with the filter size 5X5 and we have 6 such filters. As a result, we get a feature map of size 28X28X6. Here the number of channels is equal to the number of filters applied.



After the first pooling operation, we apply the average pooling and the size of the feature map is reduced by half. Note that, the number of channels is intact.



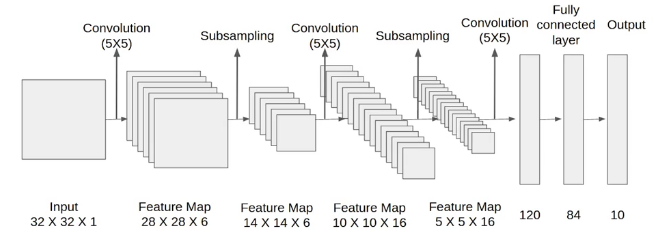
Next, we have a convolution layer with sixteen filters of size 5X5. Again the feature map changed it is 10X10X16. The output size is calculated in a similar manner. After this, we again applied an average pooling or subsampling layer, which again reduce the size of the feature map by half i.e 5X5X16.



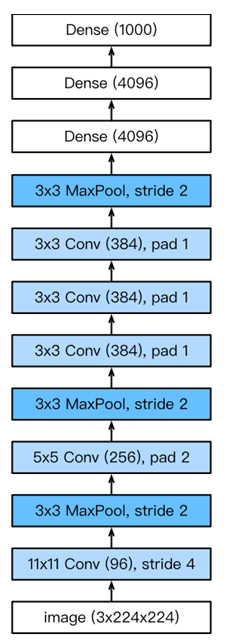
Then we have a final convolution layer of size 5X5 with 120 filters. As shown in the above image. Leaving the feature map size 1X1X120. After which flatten result is 120 values.

After these convolution layers, we have a fully connected layer with eighty-four neurons. At last, we have an output layer with ten neurons since the data have ten classes.

Here is the final architecture of the Lenet-5 model.



**AlexNet**

***What is AlexNet?***

AlexNet is an architectural network consisting of 8 convolution layers, which includes: 5 convolutional layers, 2 fully connected layers and 1 fully connected output layer. Besides, instead of using the Sigmoid activation function, in AlexNet we use the ReLU function.

***Activation function***

In AlexNet, instead of using Sigmoid, we use the ReLU function. For the simple reason that when using the Sigmoid function, the resulting output value ranges from 0 to 1, the gradient of the Sigmoid function in this range is almost zero, if the parameter initialization is not correct, it is very likely not backpropagation can be applied, causing the parameters to not continue to be updated. On the other hand, the ReLu function makes it easier for the model to be trained by various parameter initialization methods, with the gradient over the positive interval always being 1.

***Power regulation and data preprocessing***

To control the complexity of the model, while LeNet uses the Weight Decay technique, AlexNet uses the Dropout technique. Besides, AlexNet will also apply Data augmentation techniques to enrich the data set, such as flipping (rotating, flipping images), clipping (cropping images), changing color (changing images),... help the data set to be larger, make the training process more efficient, and limit the overfitting ability of the model.

***Summary***

AlexNet has a structure quite similar to the traditional LeNet network architecture, but with more convolution layers and more parameters.

Although there are now many other more efficient architectures, AlexNet is the premise for developing today's deep learning network architectures.

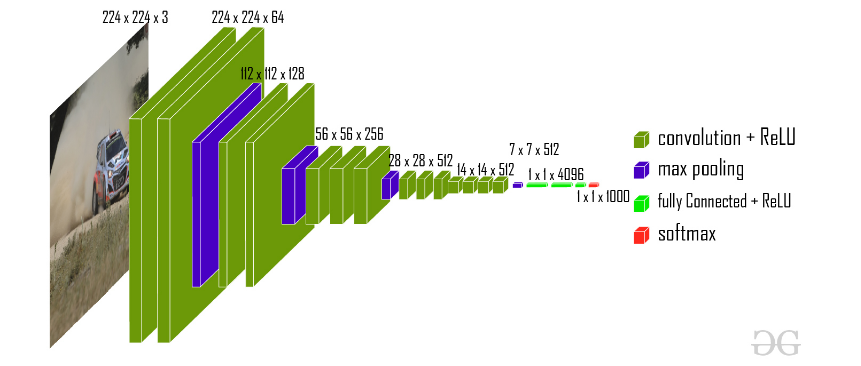
Although it seems that AlexNet's number of steps is not much more than LeNet's, it took a lot of time for scientists to research and produce excellent experimental results from this network architecture.

Dropout, ReLU function and data preprocessing with data agumentation are key to the efficiency of this network architecture.

**VGG16**

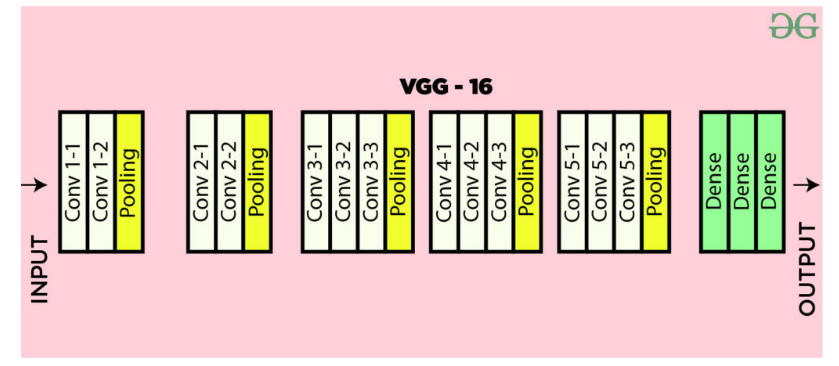
***What Is VGG16?***

VGG16 refers to the VGG model, also called VGGNet. It is a convolution neural network (CNN) model supporting 16 layers. K. Simonyan and A. Zisserman from Oxford University proposed this model and published it in a paper called Very Deep Convolutional Networks for Large-Scale Image Recognition.

VGG16 improves on AlexNet and replaces the large filters with sequences of smaller 3×3 filters. In AlexNet, the kernel size is 11 for the first convolutional layer and 5 for the second layer. The researchers trained the VGG model for several weeks using NVIDIA Titan Black GPUs. 

***VGG16 Architecture***

VGG16, as its name suggests, is a 16-layer deep neural network. VGG16 is thus a relatively extensive network with a total of 138 million parameters—it’s huge even by today’s standards. However, the simplicity of the VGGNet16 architecture is its main attraction.

The VGGNet architecture incorporates the most important convolution neural network features. 

A VGG network consists of small convolution filters. VGG16 has three fully connected layers and 13 convolutional layers.

Here is a quick outline of the VGG architecture:

Input—VGGNet receives a 224×224 image input. In the ImageNet competition, the model’s creators kept the image input size constant by cropping a 224×224 section from the center of each image.

Convolutional layers—the convolutional filters of VGG use the smallest possible receptive field of 3×3. VGG also uses a 1×1 convolution filter as the input’s linear transformation.

ReLu activation—next is the Rectified Linear Unit Activation Function (ReLU) component, AlexNet’s major innovation for reducing training time. ReLU is a linear function that provides a matching output for positive inputs and outputs zero for negative inputs. VGG has a set convolution stride of 1 pixel to preserve the spatial resolution after convolution (the stride value reflects how many pixels the filter “moves” to cover the entire space of the image).

Hidden layers—all the VGG network’s hidden layers use ReLU instead of Local Response Normalization like AlexNet. The latter increases training time and memory consumption with little improvement to overall accuracy.

Pooling layers–A pooling layer follows several convolutional layers—this helps reduce the dimensionality and the number of parameters of the feature maps created by each convolution step. Pooling is crucial given the rapid growth of the number of available filters from 64 to 128, 256, and eventually 512 in the final layers.

Fully connected layers—VGGNet includes three fully connected layers. The first two layers each have 4096 channels, and the third layer has 1000 channels, one for every class.

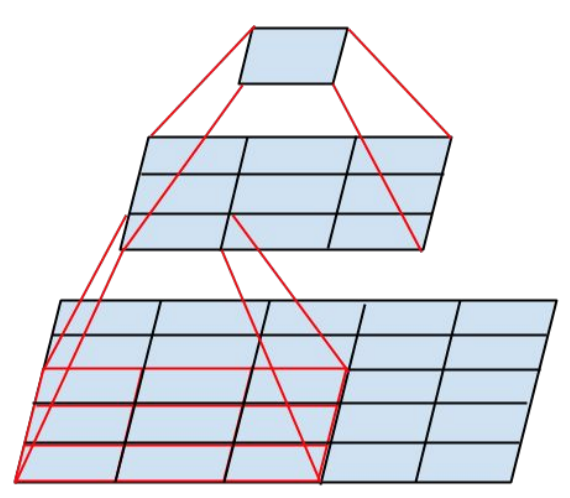
**Inception v3**

Inception v3 mainly focuses on burning less computational power by modifying the previous Inception architectures. This idea was proposed in the paper Rethinking the Inception Architecture for Computer Vision, published in 2015. It was co-authored by Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens.

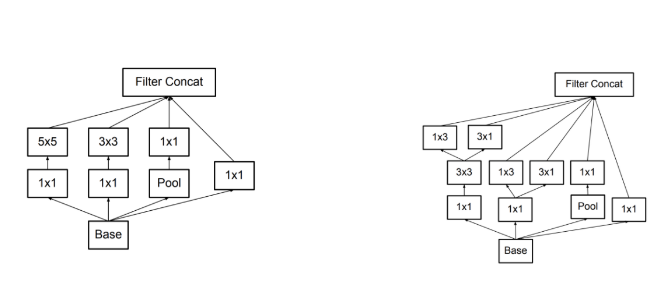
***Inception v3 Architecture***

The architecture of an Inception v3 network is progressively built, step-by-step, as explained below:

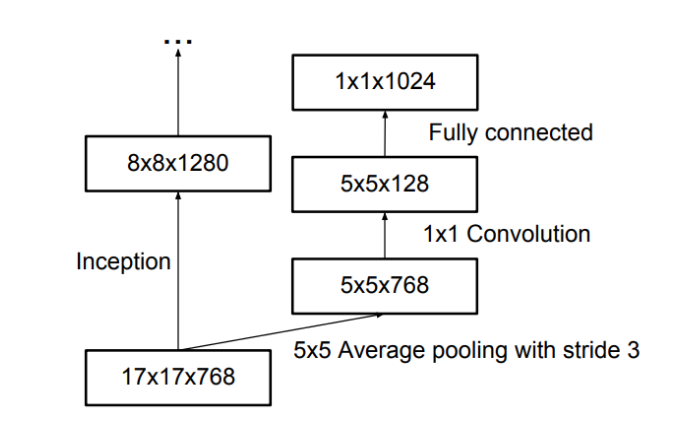
1. Factorized Convolutions: this helps to reduce the computational efficiency as it reduces the number of parameters involved in a network. It also keeps a check on the network efficiency.
2. Smaller convolutions: replacing bigger convolutions with smaller convolutions definitely leads to faster training. Say a 5 × 5 filter has 25 parameters; two 3 × 3 filters replacing a 5 × 5 convolution has only 18 (3\*3 + 3\*3) parameters instead.



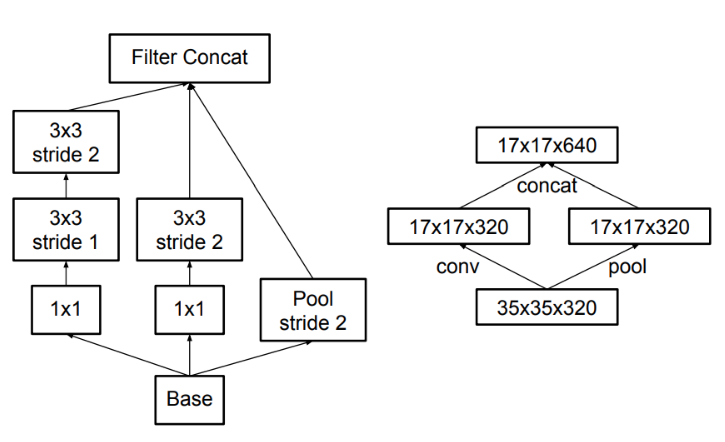
1. Asymmetric convolutions: A 3 × 3 convolution could be replaced by a 1 × 3 convolution followed by a 3 × 1 convolution. If a 3 × 3 convolution is replaced by a 2 × 2 convolution, the number of parameters would be slightly higher than the asymmetric convolution proposed.



1. Auxiliary classifier: an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss. In GoogLeNet auxiliary classifiers were used for a deeper network, whereas in Inception v3 an auxiliary classifier acts as a regularizer.



1. Grid size reduction: Grid size reduction is usually done by pooling operations. However, to combat the bottlenecks of computational cost, a more efficient technique is proposed:

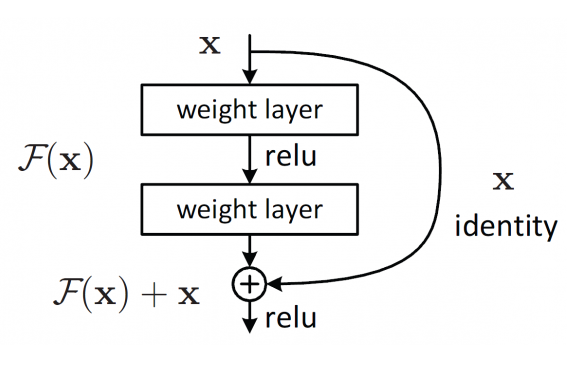


**ResNet**

As deep neural networks are both time-consuming to train and prone to overfitting, a team at Microsoft introduced a residual learning framework to improve the training of networks that are substantially deeper than those used previously. This research was published in the paper titled Deep Residual Learning for Image Recognition in 2015. And so, the famous ResNet (short for "Residual Network") was born.

When training deep networks there comes a point where an increase in depth causes accuracy to saturate, then degrade rapidly. This is called the "degradation problem." This highlights that not all neural network architectures are equally easy to optimize.

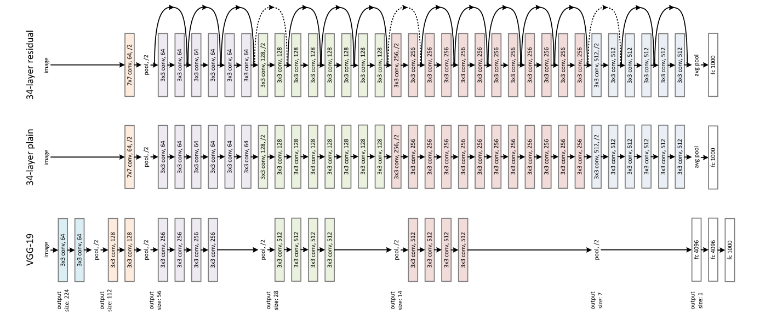
ResNet uses a technique called "residual mapping" to combat this issue. Instead of hoping that every few stacked layers directly fit a desired underlying mapping, the Residual Network explicitly lets these layers fit a residual mapping. Below is the building block of a Residual network.



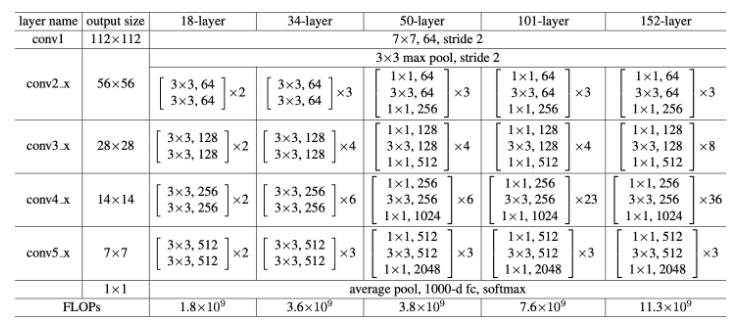
The formulation of F(x)+x can be realized by feedforward neural networks with shortcut connections.

Many problems can be addressed using ResNets. They are easy to optimize and achieve higher accuracy when the depth of the network increases, producing results that are better than previous networks. Like its predecessors which we covered in Part 1, ResNet was first trained and tested on ImageNet's over 1.2 million training images belonging to 1000 different classes.

***ResNet Architecture***

Compared to the conventional neural network architectures, ResNets are relatively easy to understand. Below is the image of a VGG network, a plain 34-layer neural network, and a 34-layer residual neural network. In the plain network, for the same output feature map, the layers have the same number of filters. If the size of output features is halved the number of filters is doubled, making the training process more complex. 

Meanwhile in the Residual Neural Network, as we can see, there are far fewer filters and lower complexity during the training with respect to VGG. A shortcut connection is added that turns the network into its counterpart residual version. This shortcut connection performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no additional parameter. The projection shortcut is mathematically represented as F(x{W}+x), which is used to match dimensions computed by 1×1 convolutions.

Below is the table showing the layers and parameters in the different ResNet Architectures. 

Each ResNet block is either two layers deep (used in small networks like ResNet 18 or 34), or 3 layers deep (ResNet 50, 101, or 152).

**DenseNet**

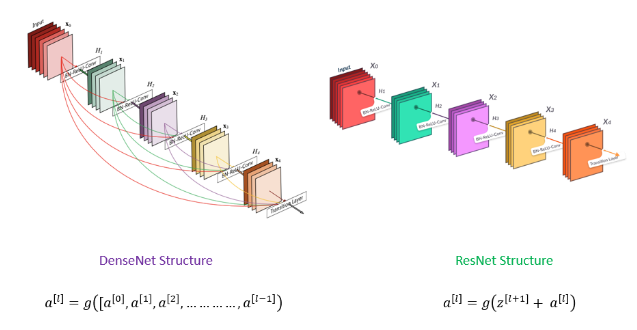
DenseNet is one of the new discoveries in neural networks for visual object recognition. DenseNet is quite similar to ResNet with some fundamental differences. ResNet uses an additive method (+) that merges the previous layer (identity) with the future layer, whereas DenseNet concatenates (.) the output of the previous layer with the future layer.

***DenseNet Architecture***

***DenseNet Structure***

DenseNet falls in the category of classic networks.

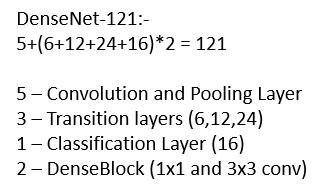
This image shows a 5-layer dense block with a growth rate of k = 4 and the standard ResNet structure.



An output of the previous layer acts as an input of the second layer by using composite function operation. This composite operation consists of the convolution layer, pooling layer, batch normalization, and non-linear activation layer.

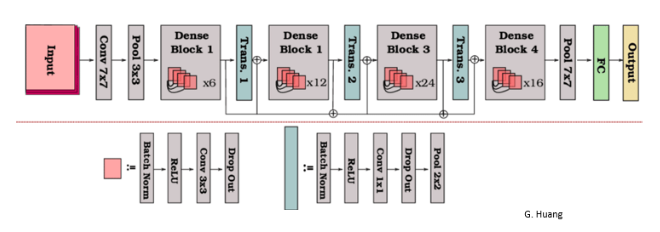
These connections mean that the network has L(L+1)/2 direct connections. L is the number of layers in the architecture.

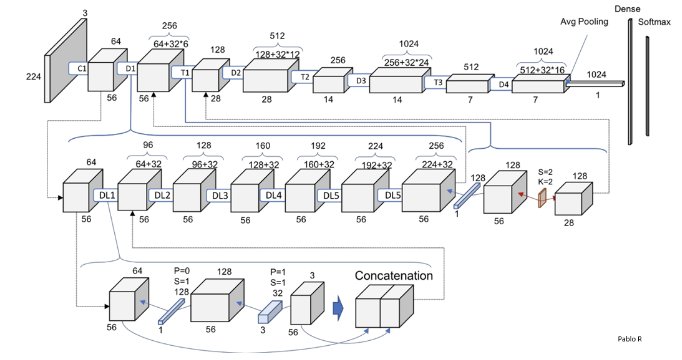
The DenseNet has different versions, like DenseNet-121, DenseNet-160, DenseNet-201, etc. The numbers denote the number of layers in the neural network. The number 121 is computed as follows:

******

***DenseBlocks and Layers***

Be it adding or concatenating, the grouping of layers by the above equation is only possible if feature map dimensions are the same. What if dimensions are different? The DenseNet is divided into DenseBlocks where a number of filters are different, but dimensions within the block are the same. Transition Layer applies batch normalization using downsampling; it's an essential step in CNN.

Let's see what's inside the DenseBlock and transition layer.: 

This is the full architecture in abstract form.: 

The number of filters changes between the DenseBlocks, increasing the dimensions of the channel. The growth rate (k) helps in generalizing the l-th layer. It controls the amount of information to be added to each layer.’

