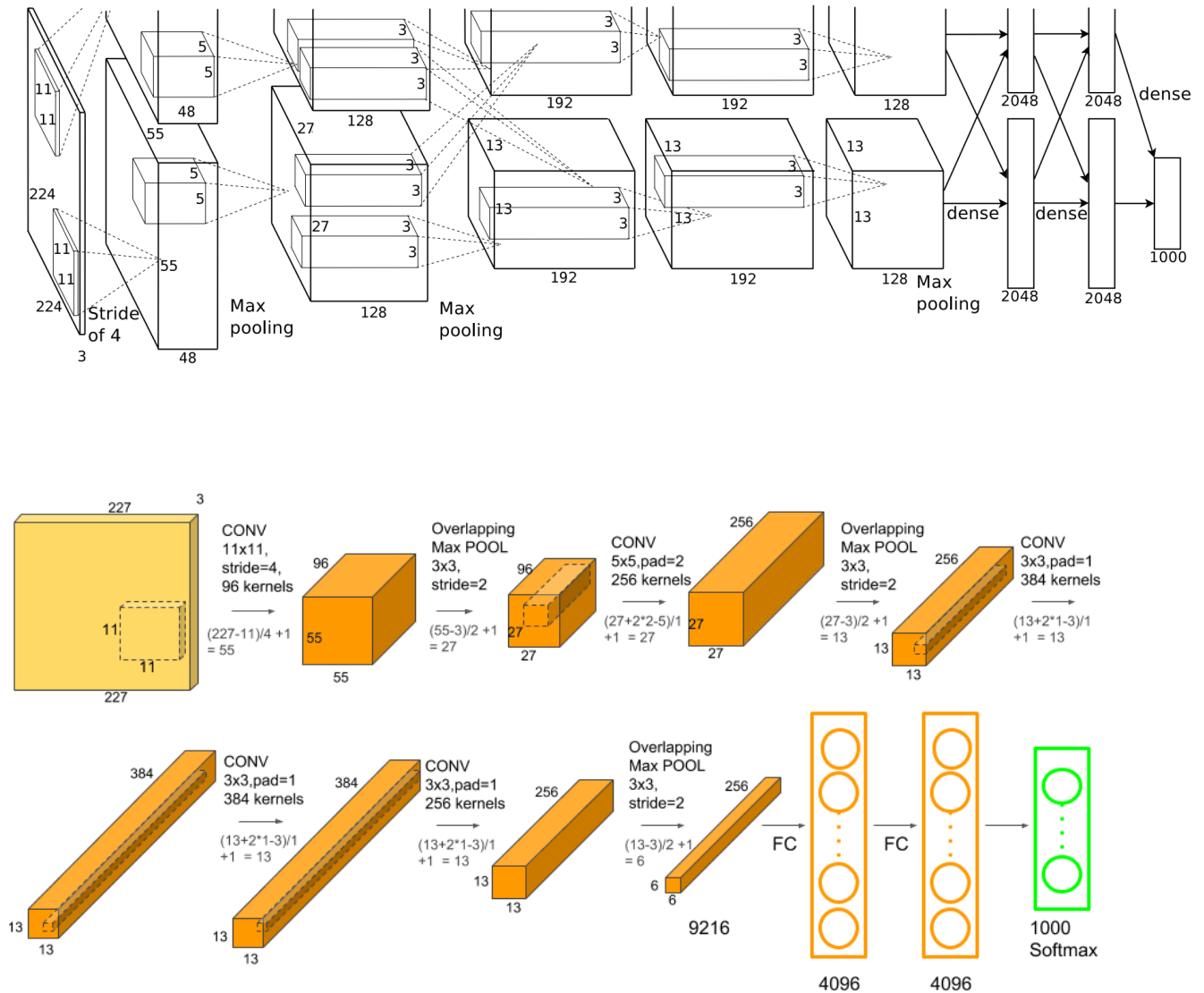


# Questions to Explore

Explore the architectures of AlexNet, VGGNet.

## AlexNet



AlexNet is a pioneering convolutional neural network (CNN) architecture that won the ImageNet Large Scale Visual Recognition Challenge in 2012. It consists of eight layers, including five convolutional layers and three fully connected layers. Here's a breakdown of its key components:

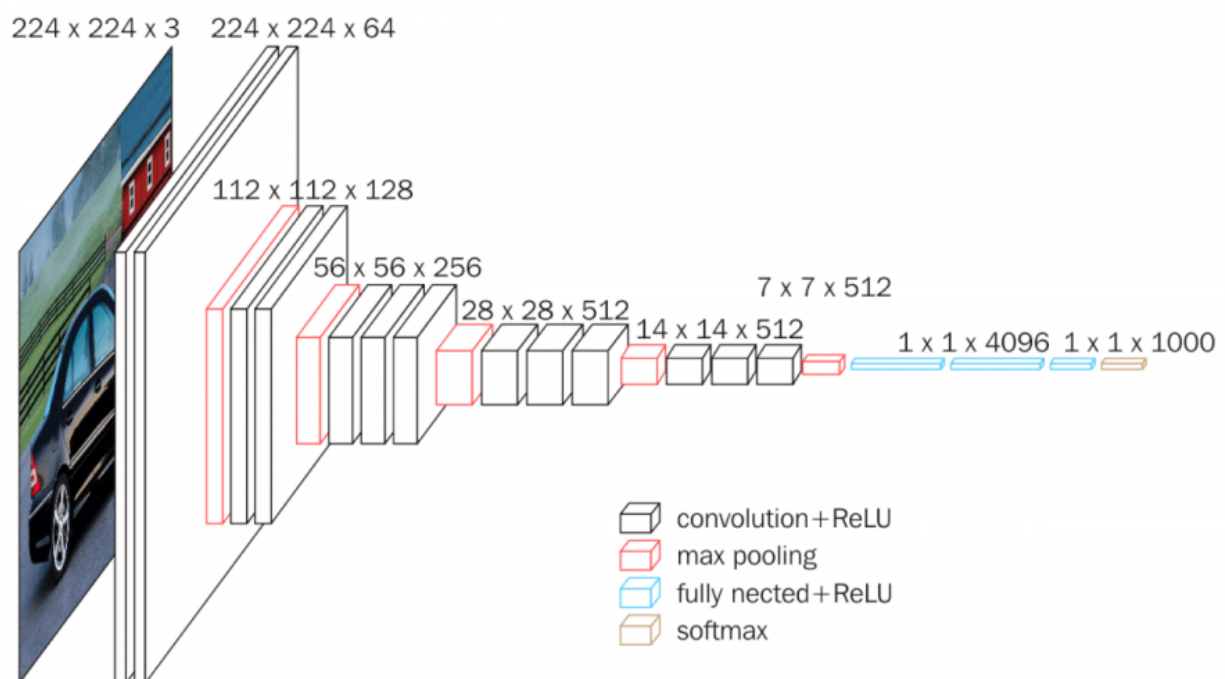
**Convolutional Layer:** For main feature extraction.

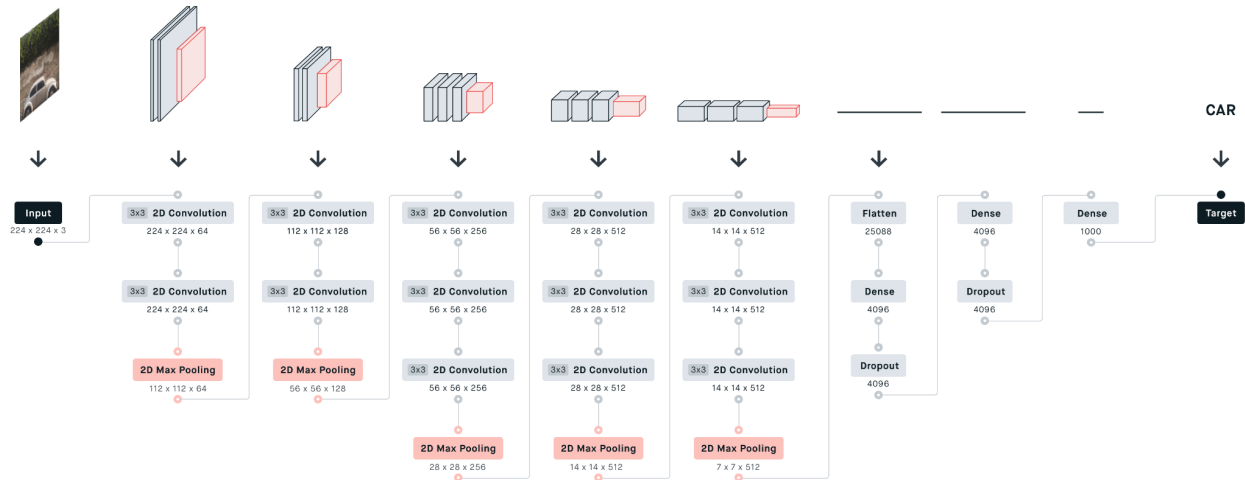
**ReLU Activation:** AlexNet popularized the use of the Rectified Linear Unit (ReLU) activation function.

**Max Pooling:** Max pooling layers follow some of the convolutional layers. They downsample the feature maps, reducing their spatial dimensions while retaining the most important features.

**Fully Connected Layers:** The final layers of AlexNet are fully connected layers that aggregate the extracted features and produce the classification output.

## VGGNet





VGGNet:

VGGNet (Visual Geometry Group) is a simple and deep architecture. It consists of 16 or 19 layers and follows a uniform structure throughout the network.

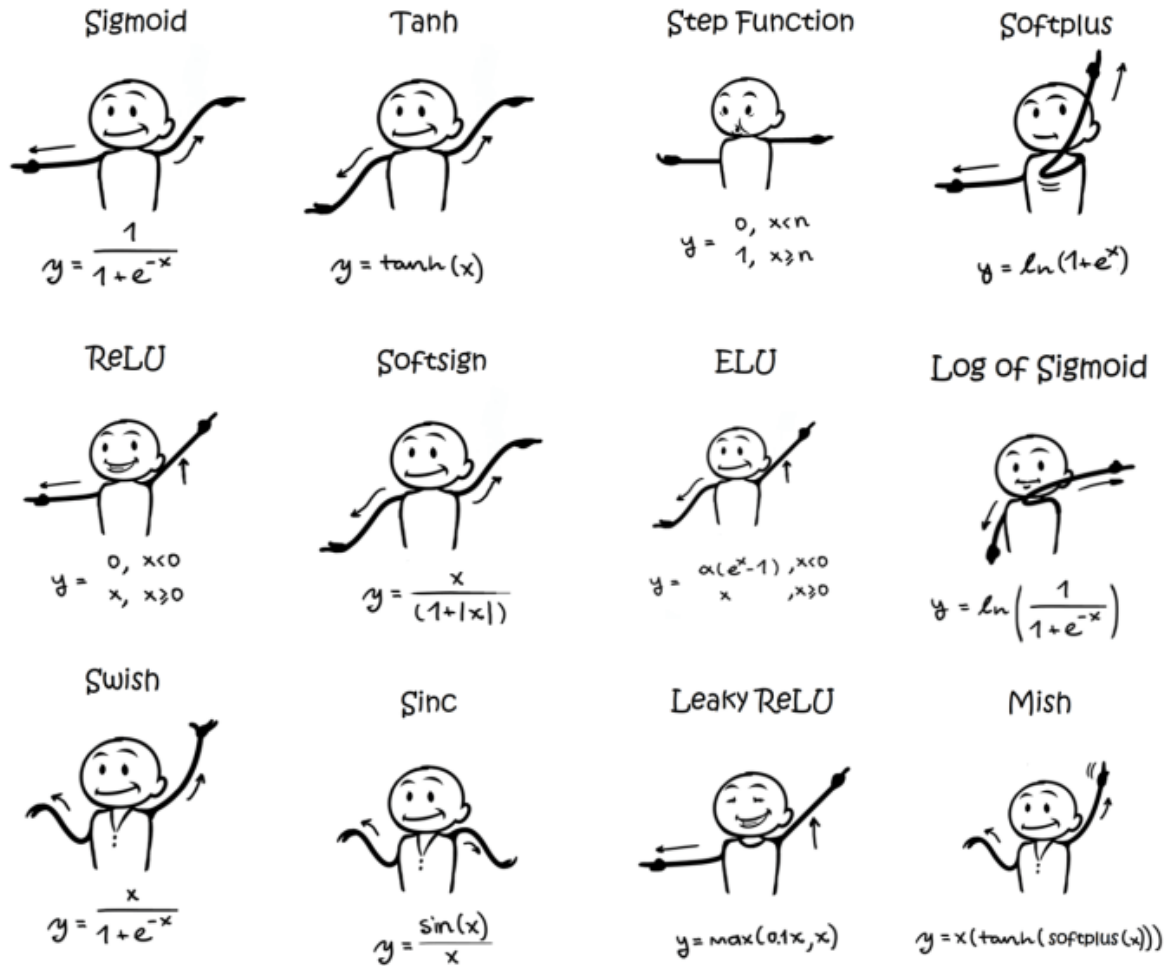
### Convolutional Layers

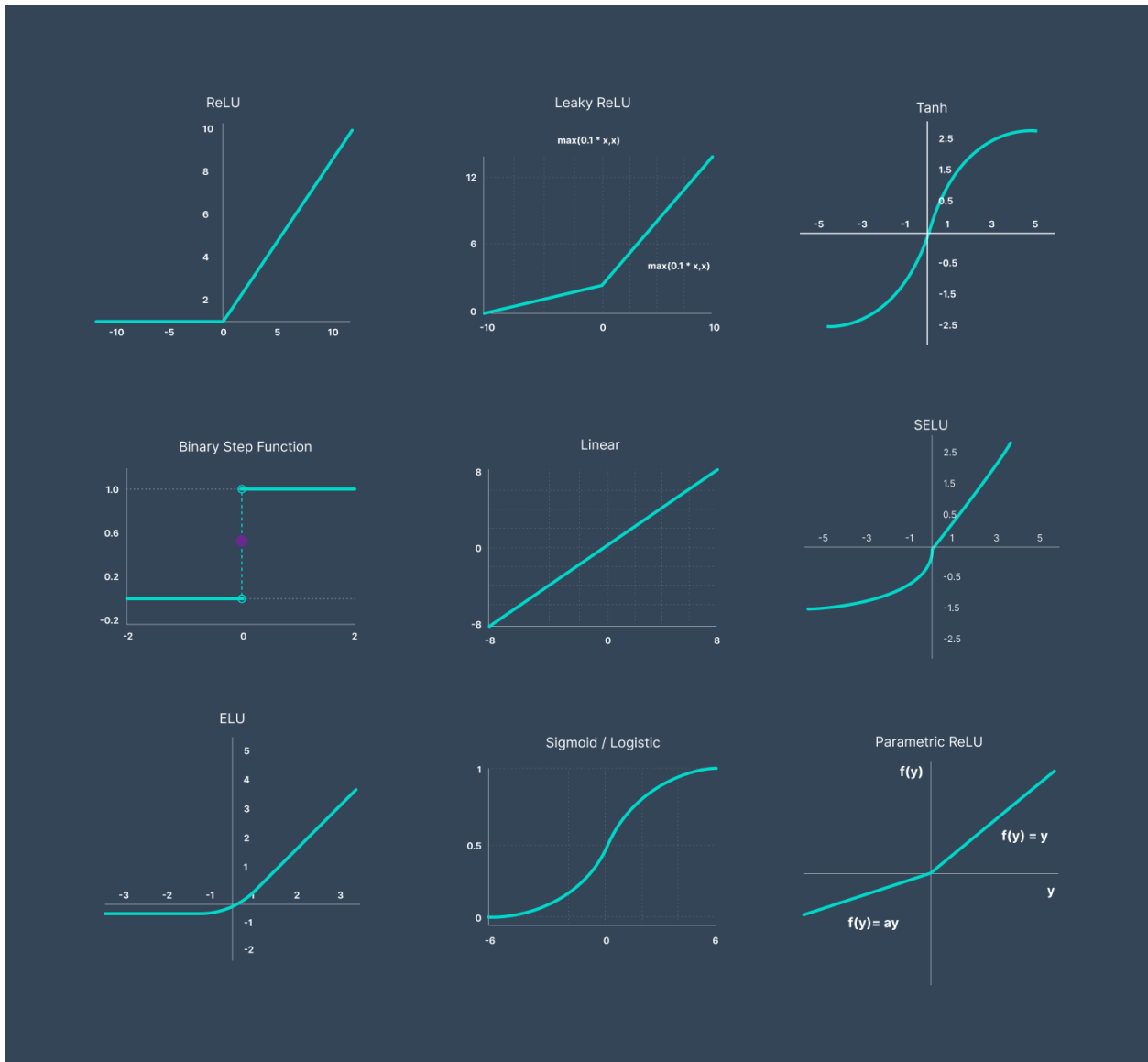
#### Max Pooling

**Deep Structure:** VGGNet is deeper than AlexNet, with up to 19 layers.

**Transfer Learning:** VGGNet has been widely adopted for transfer learning due to its pre-trained models.

Explore different activation functions, their pros and cons.


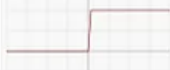











Activation functions define the output by helping neural networks to use important information while suppressing irrelevant data points. The purpose of an activation function is to add non-linearity to the neural network.

An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.

Binary, Linear, Non-linear are the 3 types of Neural Networks Activation Functions.

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

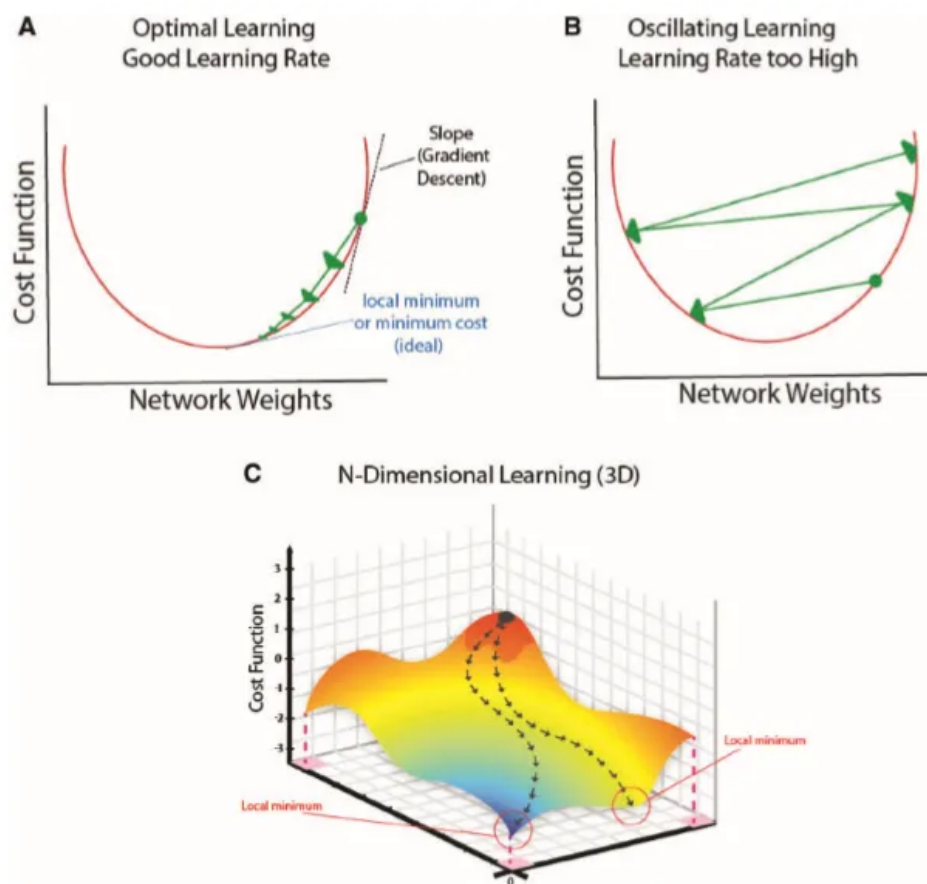
S No.	Activation function	Pros	Cons
1	<u>Sigmoid</u>	<p>-&gt; Gives you a smooth gradient while converging.</p> <p>-&gt; One of the best Normalized functions.</p>	<p>-&gt; Gives you a smooth gradient while converging.</p> <p>-&gt; One of the best Normalised functions.</p>

		-> Gives a clear prediction(classification) with 1 & 0.	-> Gives a clear prediction(classification) with 1 & 0.
<u>2</u>	<u>Tanh</u>	-> Zero-centric function unlike Sigmoid.  -> It is a smooth gradient converging function.	-> Prone to Vanishing Gradient function.  -> Computationally expensive function(exponential in nature).
<u>3</u>	<u>ReLU</u>	-> Can deal with Vanishing Gradient problem.  -> Computationally inexpensive function(linear in nature).	-> Not a zero-centric function.  -> Gives zero value as inactive in the negative axis.
<u>4</u>	<u>Leaky RELU</u>	It is the same as of RELU function except it gives some partial value(0.01 instead zero as of RELU) in the negative axis.	It is the same as of RELU function except it gives some partial value(0.01 instead zero as of RELU) in the negative axis.
<u>5</u>	<u>Softmax</u>	Normally used as the output in multi-class classification problems to find out different probabilities	

		for different classes(Unlike Sigmoid which is preferred for a binary-class classification).	

Inverse Multiquadratic, Multiquadratic, Gaussian, Maxout, Cosine, Probit, Logit, Smooth rectifier, Modified Relu(s), LeCun's Tanh, Hard Tanh, Absolute, Piecewise are few more.

Explain in detail vanishing and exploding gradients.





For a given problem and data we train a neural network model to find the solution. Once this is done, we obtain a loss as the difference between predicted and true values. We plot this loss function. For better performance we try to decrease the loss by reaching a local or ideally a global minimum in this plotted loss function.

When networks are deep the gradient calculated has less impact on the layers near the input layer during back propagation decreasing the amount of learning by the model especially in the hidden layers near the input layer. This problem is known as vanishing gradients. An illustrative image can be seen above (top part left side).

When the learning rate is large, during back propagation, the updates done to the aggregated functions of the previous layers are affected. When optimizing it jumps around the loss function and most probably never converges at a local minimum, decreasing the amount of learning by the model. This problem is known as exploding gradients. An illustrative image can be seen above (top part right side).

Best way to solve these is to choose an optimum learning rate.

## Why are neural networks with skip connections known as residual networks?

**Skip connections** are a type of shortcut that connects the output of one layer to the input of another layer that is not adjacent to it.

For example, in a CNN with four layers, A, B, C, and D, a skip connection could connect layer A to layer C, or layer B to layer D, or both.

While training deep neural nets, the performance of the model drops down with the increase in depth of the architecture. This is known as the degradation problem.

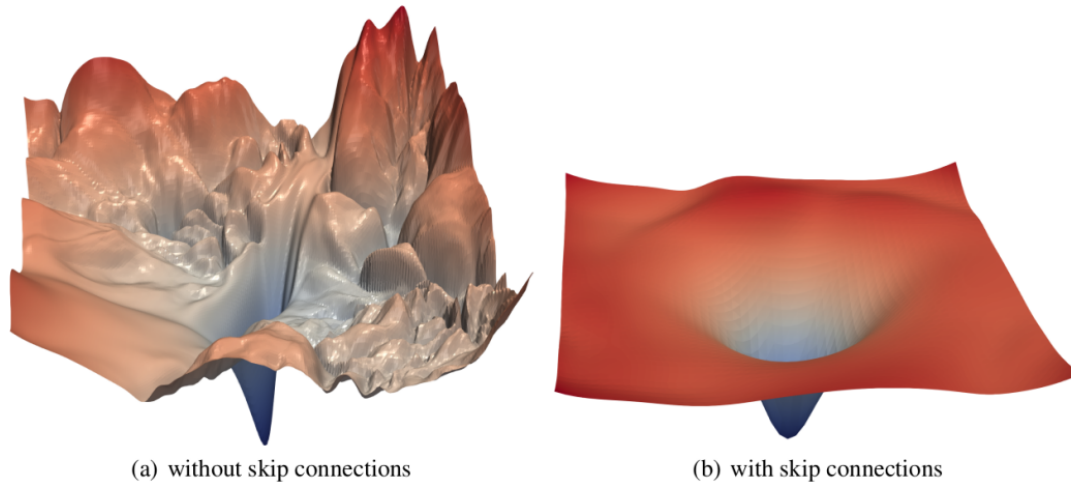
Possible reason can be vanishing gradient and/or exploding gradient problems. However, the authors of ResNet (He et al.) argued that the use of Batch Normalization and proper initialization of weights through normalization ensures that the gradients have healthy norms.

One of the primary reasons is due to random initialization of weights with a mean around zero, L1, and L2 regularization. As a result, the weights in the model would always be around zero and thus the deeper layers can't learn identity mappings as well.

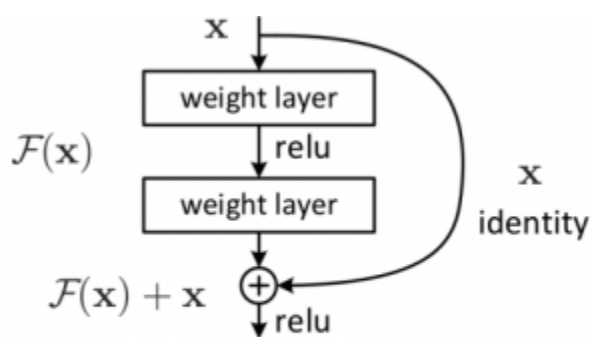
Here comes the concept of skip connections which would enable us to train very deep neural networks. Let's learn this awesome concept now.

Skip Connections were introduced to solve different problems in different architectures. In the case of ResNets, skip connections solved the degradation problem.

Work done by Li et al. which enables us to visualize the complex loss surfaces. The results from the networks with skip connections are even more surprising! Take a look at them.



Residual Networks were proposed by He et al. in 2015 to solve the image classification problem. In ResNets, the information from the initial layers is passed to deeper layers by matrix addition. This operation doesn't have any additional parameters as the output from the previous layer is added to the layer ahead. A single residual block with skip connection looks like this:



# Understand the Resnet and inception architectures.

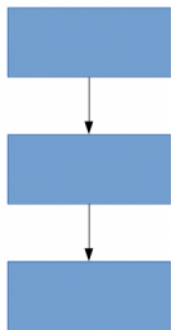
**Residual Network:** In order to solve the problem of the vanishing/exploding gradient, this architecture introduced the concept called Residual Blocks. In this network, we use a technique called skip connections. The skip connection connects activations of a layer to further layers by skipping some layers in between. This forms a residual block. Resnets are made by stacking these residual blocks together.

## Inception:

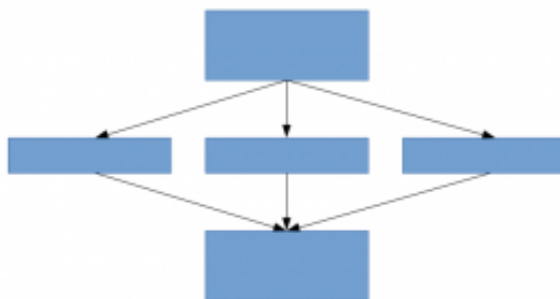
Bigger the model, the more prone it is to overfitting. This is particularly noticeable when the training data is small.

Increasing the number of parameters means you need to increase your existing computational resources.

A solution for this, is to move on to sparsely connected network architectures which will replace fully connected network architectures, especially inside convolutional layers.

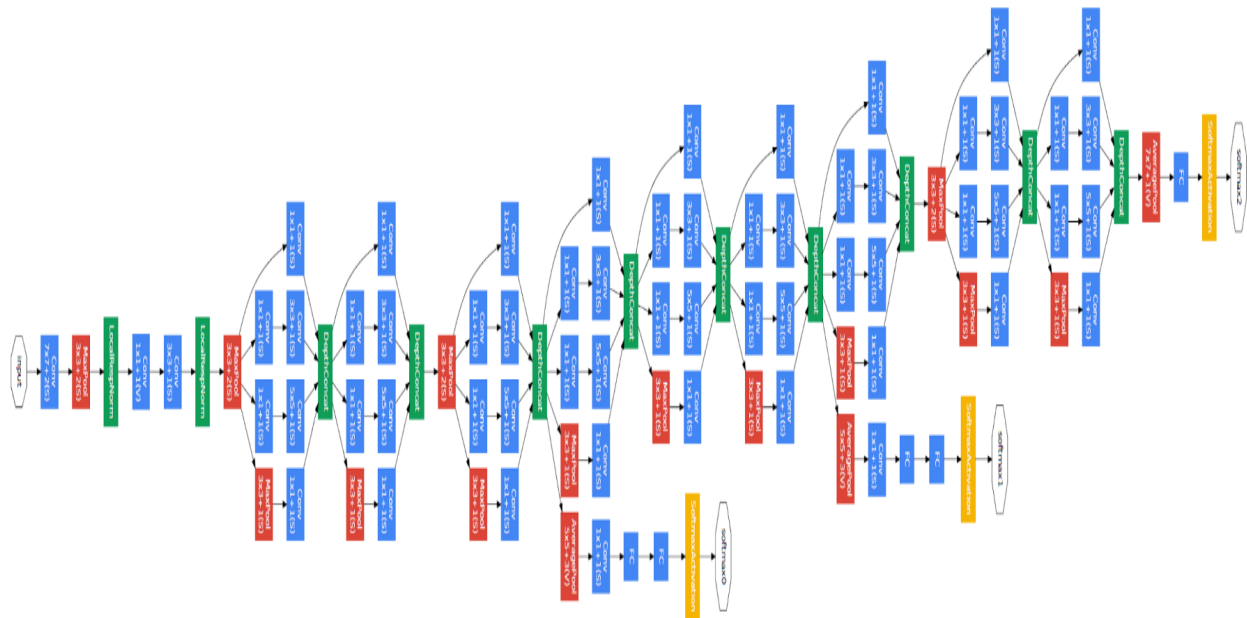


Densely connected architecture



Sparsely connected architecture

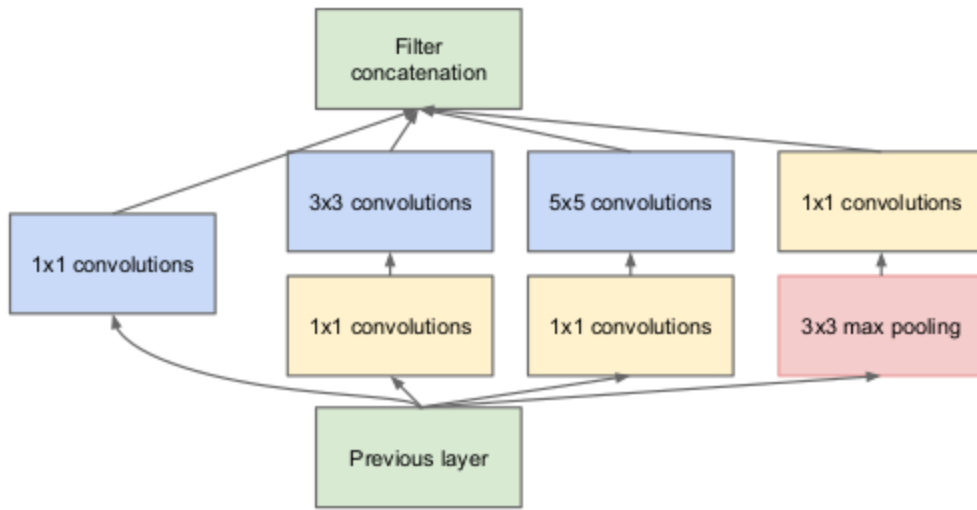
A new idea of creating deep architectures. This approach lets you maintain the “computational budget”, while increasing the depth and width of the network.



“(Inception Layer) is a combination of all those layers (namely,  $1 \times 1$  Convolutional layer,  $3 \times 3$  Convolutional layer,  $5 \times 5$  Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage.”

Along with the above-mentioned layers, there are two major add-ons in the original inception layer:

- $1 \times 1$  Convolutional layer before applying another layer, which is mainly used for dimensionality reduction
- A parallel Max Pooling layer, which provides another option to the inception layer



Inception Layer

To understand the importance of the inception layer's structure, the author calls on the Hebbian principle from human learning. This says that “neurons that fire together, wire together”. The author suggests that when creating a subsequent layer in a deep learning model, one should pay attention to the learnings of the previous layer.