**Brief Review of CNN Models:**

***Inception, ResNet, DenseNet, and VGG***

**Inception Model:**

**Introduction:**

In 2014, Google (together with other academic institutions) published a paper describing a novel deep learning convolutional neural network architecture, which at the time was the largest and most efficient deep neural network architecture available.

An Inception Network was used as the unique design, and a GoogLeNet version won the ImageNet Large-scale Visual Recognition Challenge 2014 classification computer vision competition with the best performance (ILVRC14).

Since 2014, both we and the deep learning field have come a long way. Several deep learning architectures will approach or exceed human-level classification and object detection performance by 2020.

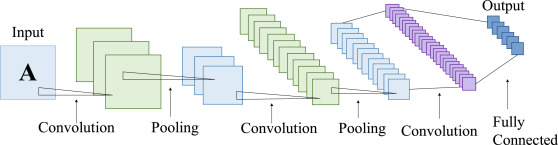
On the other hand, convolutional neural networks owe their innovations and advancements to their forefathers.

**Architecture of Inception Model:**

An inception network is a deep neural network with Inception modules, which are repeating architectural designs.

The following are the essential principles to remember when developing an Inception model:

1. High performance necessitates the use of large deep neural networks. To be considered huge, a neural network must include multiple more layers and units inside these levels.
2. Convolutional neural networks benefit from extracting features at various scales. The biological human visual brain recognises patterns at different scales, which are then combined to create larger object perceptions. As a result, multi-scale networks are able to learn more.
3. Consider the Hebbian Principle, which claims that neurons that fire at the same time will connect.



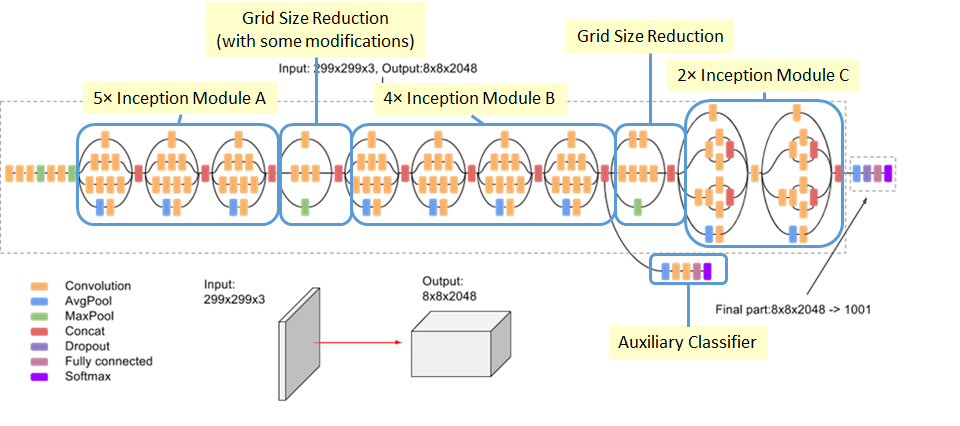
In practical cases, the above-mentioned guidelines have several technological flaws. In large networks, overfitting is prevalent, and chaining numerous convolutional methods together increases the network's computing cost. To make the Inception network and module a reality, the researchers designed intuitive convnet topologies.

In this 2013 report, Lin et al. proposed 1x1 convolution, popularly known as "Network in Network." A 1x1 convolution uses the element-wise product of all pixel values in an image. The image (input data) is convoluted with the conv 1x1 filter, resulting in an output with the dimensions 1 x 1 x n (where 'n' is the number of filters). A 1x1 filter does not learn spatial patterns inside the image, but it does learn patterns across the depth of the image (cross channel). As a result, 1x1 convolution filters not only decrease method dimension but also allow the network to learn more.

The 1x1 convolution reduces the number of input channels, which results in fewer channels in the output. This section is the Inception network's bottleneck layer (shown in a diagram further down below). Pooling layers down-sample images as they go through the network (lower the height and width). 1x1 convolution has the added benefit of reducing the image's height, width, and number of channels.

To improve the performance of a neural network, increase the number of layers (depth) and units/neuron within the layers (width), which ineffectively develops a more comprehensive network.

Structure of a basic ResNet model is given below:



The Inception network has the luxury of using different filter sizes within its convolutional layers. The 1x1 convolution, one of the convolution filter sizes used by Inception, has already been addressed. The others are 3x3 and 5x5.

Within a convnet, different conv filter sizes learn spatial patterns and recognise properties at different scales.

Prior to the advent of the Inception network, researchers had to choose filter sizes to use in deep convolutional neural networks to obtain optimal performance.

By utilising different filter sizes of 1x1, 3x3, and 5x5, Inception eliminates the need for such decisions. 1x1 learns patterns across the depth of the input, whereas 3x3 and 5x5 learn spatial patterns across the three dimensions of the input (height, width, and depth). The representational capacity rises when all of the patterns acquired from the various filter sizes are combined.

To give a single Inception module output, the Inception module contains a concatenation layer that merges all of the conv filters' outputs and feature maps into a single object.

**Benefits of the Inception Module:**

1. Convolutional neural networks provide a high level of performance gain.
2. Efficient computing resource utilisation for an Inception network's high-performance output with minimum increase in compute load
3. Extraction of features from input data at various scales using different convolutional filter sizes
4. Cross-channel patterns are learned by 1x1 conv filters, allowing the network to extract more information.

**ResNet**

**Introduction:**

ResNet, or Residual Networks, is a well-known neural network that is used to perform a variety of computer vision tasks. This model was the ImageNet challenge winner in 2015. ResNet was a game-changer because it allowed us to train 150-layer deep neural networks. Prior to ResNet, training very deep neural networks was challenging due to the problem of vanishing gradients.

AlexNet, the ImageNet 2012 winner and the model that seems to have spurred interest in deep learning, has just eight convolutional layers, compared to 19 for the VGG network, 22 for Inception or GoogleNet, and 152 for ResNet 152.

**Architecture of ResNet:**

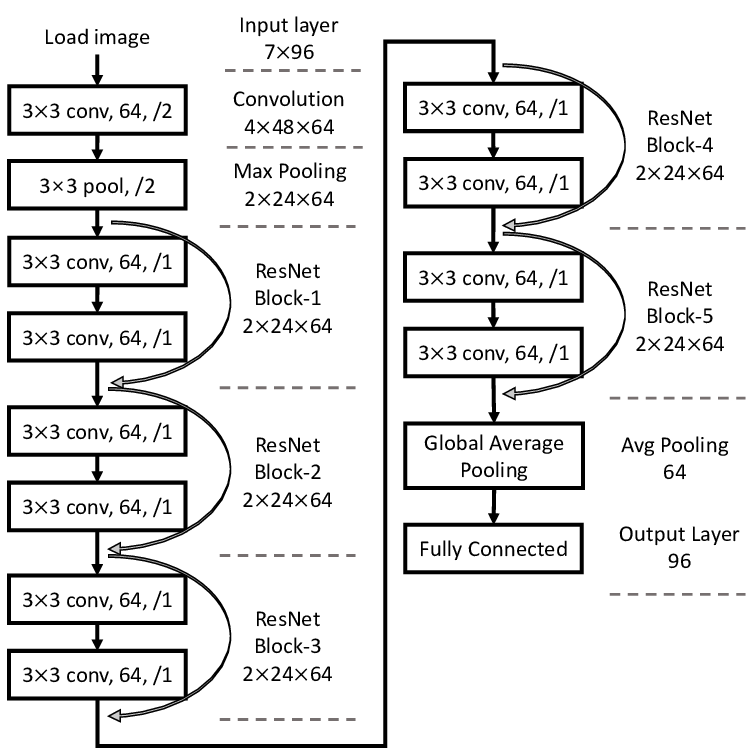
ResNet is a Convolutional Neural Network (CNN) architecture that tackles the "vanishing gradient" problem, enabling deeper networks to outperform shallower networks. A decreasing gradient happens during backpropagation. When there are too many layers in the neural network training process, the gradient becomes very small until it disappears, and optimization is halted. The problem is resolved by ResNet adopting "identity shortcut connections."

It has two stages of implementation:

ResNet generates and bypasses a large number of levels that aren't used initially, recycling activation functions from preceding layer.

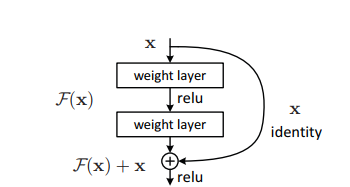
A second stage involves retraining the network and extending the "residual" convolutional layers. This enables the investigation of more parts of the feature space that a shallow convolutional network architecture would have skipped.

Structure of a basic ResNet model is given below:



ResNet was the first to introduce the concept of skip connection. The graphic below depicts the skip connection. Convolution layers are piled one on top of the other in the figure on the left. Convolution layers are still stacked on the right, but the original input is now added to the convolution block's output. This is known as "skipping connection."

These Residual blocks were introduced to help with the issue of training very deep networks, and the ResNet model is made up of them.



The introduction of these Residual blocks alleviated the challenge of training very deep networks, and the ResNet model is built up of these blocks. The first thing we note in the above diagram is that there is a direct connection that skips several of the model's levels. The heart of residual blocks is a connection known as the ‘skip connection.' Because of the skip connection, the output is not same. Without the skip connection, input 'X is multiplied by the layer's weights, then a bias term is added. Then comes the activation function, f() and we get the output as H(x).

**H(x)=f(wx + b) or H(x)=f(x)**

Thanks to the installation of a new skip connection method, the output is now H(x), where

**H(x)=f(x)+x**

However, when using a convolutional layer or pooling layers, the input dimension may differ from the output dimension. As a result, these two ways can be used to solve the problem:

* To enhance its dimensions, zero is padded with the skip connection.
* To match the dimensions, 11 convolutional layers are added to the input.

In this situation, the result is:

**H(x)=f(x)+w1.x**

In this case, an additional parameter w1 is added, but in the first technique, no additional parameter is added.

ResNet's skip connections strategy tackles the problem of disappearing gradient in deep CNNs by enabling the gradient to flow along an additional shortcut channel. In addition, if any layer degrades architecture performance, regularisation will skip it.

The architecture is inspired on VGG-19 and has a 34-layer plain network to which shortcut and skip connections are added. These residual blocks or skip connections change the design into a residual network.

**Advantages of ResNet:**

1. Networks with a large number of layers (even thousands) can be easily taught without increasing the percentage of training errors.
2. Using identity mapping, ResNets can help solve the vanishing gradient problem.

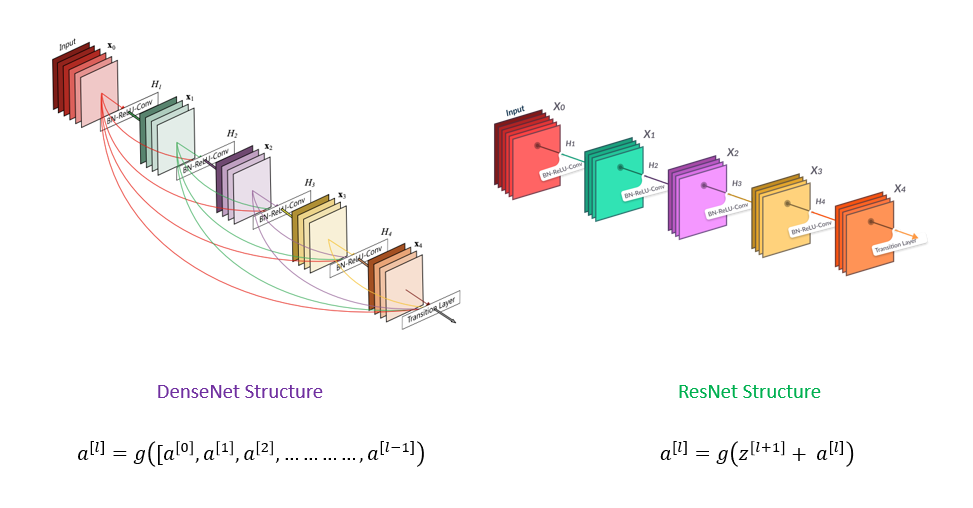
**DenseNet:**

**Introduction:**

The Dense Convolutional Network, or simply DenseNet, is an innovative neural network discovery for visual object detection. DenseNet is quite similar to ResNet, although there are several key distinctions. DenseNet concatenates (.) the output of the previous layer with the output of the future layer, whereas ResNet utilises an additive approach (+) that combines the previous layer (identity) with the future layer. DenseNet was created primarily to address the vanishing gradient's effect on high-level neural networks' accuracy. Simply said, the information evaporates before reaching its target due to the longer travel between the input and output layers. It's a convolutional neural network that use Dense Blocks to create dense connections between layers, with all layers (with matching feature-map sizes) connected directly to one another. To retain the feed-forward nature, each layer gets extra inputs from all preceding levels and passes on its own feature-maps to all subsequent layers.

**Architecture:**

Dense Convolutional Network (DenseNet) is a feed-forward network that connects each layer to every other layer. Our network has L(L+1)/2 direct connections, whereas standard convolutional networks with L layers have L connections - one between each layer and its succeeding layer. All previous layers' feature maps are utilised as inputs into each layer, and its own feature maps are used as inputs into all subsequent layers. DenseNets have a number of compelling advantages, including the elimination of the vanishing-gradient problem, improved feature propagation, feature reuse, and a significant reduction in the number of parameters.



The standard ResNet structure and a 5-layer dense block with a growth rate of k = 4 are shown in this figure. Using the composite function operation, the previous layer's output becomes the second layer's input. Convolution, pooling, batch normalisation, and non-linear activation layers are all part of this composite operation.

The network has L(L+1)/2 direct connections because of these links. The architecture has L layers.

DenseNet comes in a variety of flavours, including DenseNet-121, DenseNet-160, and DenseNet-201. The numbers represent the neural network's layer count.

DenseNet (Dense Convolutional Network) is a network design that focuses on making deep learning networks grow deeper while also making them more efficient to train by employing shorter connections between layers. DenseNet is a convolutional neural network in which each layer is connected to all other layers deeper in the network, so the first layer is connected to the 2nd, 3rd, 4th, and so on. This is done to allow maximal information flow between network tiers. Each layer takes inputs from all previous levels and passes on its own feature maps to all subsequent layers to maintain the feed-forward character. Unlike ResNets, it concatenates features instead of summarising them. The 'ith' layer, then, contains I inputs and is made up of feature maps from all the convolutional blocks before it. All of the following 'I-i' layers receive its own feature maps. In contrast to standard deep learning designs, this introduces '(I(I+1))/2' connections to the network. As a result, it has fewer parameters than standard convolutional neural networks because no meaningless feature maps need to be learned.

Apart from the basic convolutional and pooling layers, DenseNet has two key components. Dense Blocks and Transition layers are what they're called.

DenseNet begins with a basic layer of convolution and pooling. Then there's a dense block followed by a transition layer, followed by another dense block followed by a transition layer, followed by another dense block followed by a transition layer, and lastly a dense block followed by a classification layer.

The first convolution block contains 64 7x7 filters with a stride of 2. A MaxPooling layer with 3x3 max pooling and a stride of 2 follows.

Each convolutional block passes through the following phases after receiving the input: BatchNormalization, ReLU activation, and lastly the actual Conv2D layer. Every dense block has two convolutions, with kernel sizes of 1x1 and 3x3. This is repeated six times in dense block 1, twelve times in dense block 2, twenty-four times in dense block 3, and sixteen times in dense block four.

Each of the 1x1 convolutions in dense block has four times the number of filters. As a result, we use four filters, but three of them are only present once. In addition, the input and output tensors must be concatenated.

**Advantages of DenseNet:**

They solve the vanishing-gradient problem, improve feature propagation, promote feature reuse, and cut the number of parameters in half. In comparison to alternative systems, the DenseNet architecture has numerous notable advantages. The authors first claim that their architecture outperforms the other competing architectures in ImageNet. My research in Near-Identical Images confirmed that the DenseNet design indeed provide the optimum image representation. Second, the authors claim that their increased parameter efficiency makes it easier to train the network. This is true when compared to other network topologies of equal size. I believe that the training time is competitive with that of certain lower-layer networks. The improved performance is well worth the additional training time.

**VGG16:**

**Introduction:**

**In their publication "Very Deep Convolutional Networks for Large-Scale Image Recognition," K. Simonyan and A. Zisserman from the University of Oxford proposed the VGG16 convolutional neural network model. In ImageNet, a dataset of over 14 million images belonging to 1000 classes, the model obtains 92.7 percent top-5 test accuracy. It was a well-known model that was submitted to the ILSVRC-2014. It outperforms AlexNet by sequentially replacing big kernel-size filters (11 and 5 in the first and second convolutional layers, respectively) with numerous 33 kernel-size filters. VGG16 had been training for weeks on NVIDIA Titan Black GPUs.**

VGG is a multilayer deep Convolutional Neural Network (CNN) architecture that stands for Visual Geometry Group. VGG-16 and VGG-19 feature 16 and 19 convolutional layers, respectively, and the term "deep" refers to the number of layers.

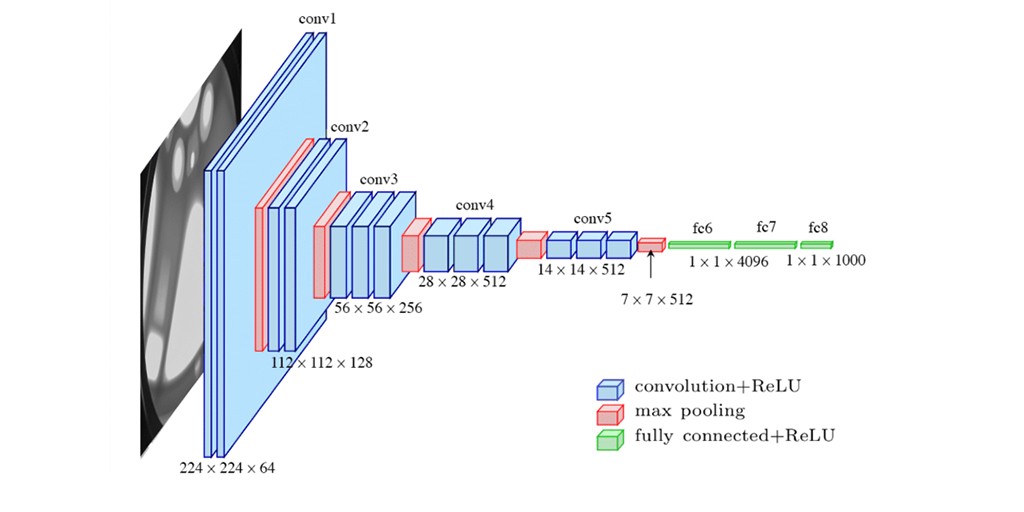
Cutting-edge object recognition models are built on top of the VGG architecture. On a range of applications and datasets other than ImageNet, the VGGNet, which was constructed as a deep neural network, beats baselines. It is also one of the most commonly utilised image recognition architectures today. The VGGNet-16 contains 16 layers and can categorise pictures into 1000 different object categories, including keyboards, animals, pens, and mice. The model also accepts images with a resolution of 224 by 224 pixels.

**Architecture:**

Small convolutional filters are used to build the VGG network. There are 13 convolutional layers and three fully linked layers in the VGG-16.

Let's look at VGG's architecture in more detail:

* **Input:** The VGGNet accepts images with a size of 224x224 pixels. To keep the image input size consistent for the ImageNet competition, the model's authors chopped out the centre 224x224 patch in each image.
* **Convolutional Layers**: VGG's convolutional layers use a small receptive field (3x3), the smallest size that still captures up/down and left/right movement. There are also 1x1 convolution filters that perform a linear transformation of the input. Then there's a ReLU unit, which is a significant AlexNet invention that cuts training time in half. The rectified linear unit activation function (ReLU) is a piecewise linear function that outputs the input if it is positive and zero otherwise. To maintain spatial resolution after convolution, the convolution stride is set to 1 pixel (stride is the number of pixel shifts over the input matrix).
* **Hidden Layers:** The VGG network's hidden layers all use ReLU. Local Response Normalization (LRN) is rarely used in VGG since it increases memory usage and training time. Furthermore, it has no effect on total accuracy.
* **Fully Connected Layers:** There are three fully connected layers in the VGGNet. The first two layers each have 4096 channels, whereas the third layer has 1000 channels, one for each class.



The number 16 in the term VGG alludes to the deep neural network's 16 layers (VGGNet). This indicates that VGG16 is a large network with over 138 million parameters. Even by modern standards, it is a massive network. The simplicity of the VGGNet16 architecture, on the other hand, is what makes the network appealing. It may be argued that its architecture is quite uniform just by glancing at it.

A few convolution layers are followed by a pooling layer that decreases the height and width of the image. When it comes to the number of filters we can employ, we have roughly 64 options, which we can expand to around 128 and then to 256. We can utilise 512 filters in the final levels.

Every step or stack of the convolution layer doubles the number of filters that can be used. This is a key design principle for the VGG16 network's architecture. One of the major disadvantages of the VGG16 network is that it is a large network, which means that training its parameters takes longer.

The VGG16 model is over 533MB in size because to its depth and number of fully connected layers. Implementing a VGG network is hence time-consuming.

Although the VGG16 model is employed in a variety of deep learning image classification challenges, smaller network topologies like GoogleNet and SqueezeNet are frequently preferred. In any case, the VGGNet is an excellent learning building block because it is simple to implement.

In the ILSVRC-2012 and ILSVRC-2013 contests, VGG16 outperformed prior versions of models. Furthermore, the VGG16 result is fighting for the classification task winner (GoogLeNet with 6.7 percent error) and exceeds the ILSVRC-2013 winning submission Clarifai by a significant margin. It scored roughly 11.7 percent using external training data and 11.2 percent without. In terms of single-net performance, the VGGNet-16 model outperforms a single GoogLeNet by roughly 0.9 percent, with about 7.0 percent test error.

**Advantages of VGG16:**

As the number of layers in CNN grows, the model's ability to fit more complex functions grows as well. As a result, more layers imply better performance. This is not the same as an Artificial Neural Network (ANN), where increasing the number of layers does not always result in higher performance.

In both the ILSVRC-2012 and ILSVRC-2013 competitions, VGG16 outperformed the previous generation of models. In terms of single-net performance, the VGG16 architecture came out on top (7.0 percent test error). The mistake rates are listed in the table below.

When working with a VGG network, there are two major downsides to be mindful of. To begin with, training requires time. Second, the weights associated with network architecture are substantial. The trained VGG16 model is almost 500MB in size because to its depth and amount of completely linked nodes. Smaller network topologies are often preferred over VGG16 in many deep learning image categorization challenges (such as SqueezeNet, GoogleNet, etc.)