Why can't we just add more layers to increase performance?

- To accomplish increasingly complex tasks, researchers often create deeper models by adding more layers to their networks. However there's a notable limitation with this approach, and one significant challenge is the vanishing gradient problem.
- Neural networks learn by adjusting their weights based on the loss between the predicted and actual values. This adjustment process is called backpropagation, where the gradient of the loss with respect to the weights is calculated and the weights are updated accordingly.
- Gradients can diminish significantly as they propagate backward through the layers, as the gradient is computed by multiplying the gradients at each layer. These gradients can become so small that they "vanish", which leads to negligible updates in the weights, which can stall learning.
- The models discussed below employ innovative techniques to overcome such challenges and construct more robust networks.

ResNet50

- Residual Networks (ResNets) are designed to address challenges in training deep neural networks, particularly the vanishing gradient problem. Resnets mitigate this issue through the use of shortcut connections that allow gradients to flow directly to earlier layers without passing through all intermediate layers.
- The building block of a ResNet is the residual block. This block includes both a shortcut path and a main path. The shortcut path bypasses the convolutional layers and directly connects the input to the output, while the main path processes the input through the convolutional layers. The output of the main path $F(x)$ is then added to the output of the shortcut path (x), yielding the output of the residual block $y=F(x)+x$.
- The goal is to make the output of the main path $F(x)$ as close to zero as possible so that the output y approximates the input x. The network learns to adjust the weights in the main path to minimize $F(x)$. Consequently, the output y becomes primarily influenced by the shortcut path, which involves few layers, preventing the issue of the vanishing gradients.

InceptionNet

- InceptionNet, also known as GoogleNet, uses inception modules, which employ parallel filters of different sizes at the same layer to capture features at various spatial scales. An inception model typically consists of 1x1, 3x3, and 5x5 convolutional filters followed by a max pooling layer. The filters are applied in parallel, and their outputs are concatenated before being passed to the next module in the network.
- The core concept behind the inception modules is to leverage filters of different sizes to capture information at both global and local scales: larger kernels are useful for information that is distributed globally, while smaller kernels are better for information that is distributed locally.
- By combining filters of various sizes, the InceptionNet network can represent a wide range of features, making it ideal for computer vision applications.

Efficient Net

- EfficientNet employs a compound scaling approach to efficiently scale up models by maintaining a fixed ratio across three key dimesions: width, depth, and image resolution. This scaling method uniformly scales all three dimensions with a fixed ratio, contributing to a harmonious adjustment of model size.
- The compound scaling is controlled by a single parameter phi, which determines the model size. Larger phi values tend to result in larger and more powerful models.
- This technique has resulted in increased performance in various computer vision tasks.