|  |  |  |  |
| --- | --- | --- | --- |
| **CLASSIFICATION ALGORITHMS** | **PROS** | **CONS** | **HIGH LEVEL STRUCTURE OF THE ALGORITHM** |
| **ResNet** | * **Improved accuracy** * **Faster convergence** * **Transfer learning**: These models can be effectively used for transfer learning by performing fine-tuning on a smaller dataset, making them useful for practical applications where the availability of labeled data is limited. * **Better generalization**: ResNets have been shown to generalize better than traditional deep neural networks, which is essential for real-world applications where data distribution may change over time | * **Data requirements:** ResNet models typically require large amounts of labeled training data to achieve optimal performance. When training on limited data, the model may struggle to generalize well or may exhibit high variance in performance. Data augmentation techniques can help alleviate this issue to some extent * **Increased complexity** * **Overfitting** | * ResNet architectures come in different variants, such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, indicating the number of layers in the network. Deeper variants tend to have better performance but require more computational resources and may be more prone to overfitting. * RESNET 18 can be described as follows. There is a total of eighteen layers in the network (17convolutional layers, a fully-connected layer and an additional softmax layer to perform classification task). The convolutional layers use 3 × 3 filters and the network is designed in such a way that if the output feature map is the same size then the layers have the same number of the filters. However, filters get doubled in the layers, if the output feature map is halved. The down sampling is per-formed by convolutional layers having a stride of 2. Lastly, there is an average-pooling followed by a fully-connected layer with a softmax layer. Throughout the net-work, residual shortcut connections are inserted between layers.     **FIGURE 01: BUILDING BLOCK OF RESNET** |
| **DENSNET** | * **Overfitting**: The DenseNet design successfully tackles overfitting by lowering the number of parameters and enabling feature reuse, enhancing the model’s capacity to generalize to unknown data. * **Vanishing Gradients:** The DenseNet design mitigates the vanishing gradient issue by allowing gradients to flow across the whole network, allowing the training of deeper networks. * **Redundancy:** The DenseNet design manages redundancy successfully by offering feature reuse and lowering the number of parameters, enhancing the model’s capacity to generalize to unknown data. | * **Memory consumption:** DenseNet models can require a significant amount of memory * **Model size and parameter redundancy:** DenseNet models tend to have a larger number of parameters compared to other architectures * **Training sensitivity to hyper parameters:** DenseNet models can be sensitive to the choice of hyper parameters, such as growth rate and depth. | * 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:   DenseNet-121:    **FIGURE 02: BUILDING BLOCK OF DENSENET-121**   * The architecture of DenseNet is composed of transition layers and dense blocks. Each convolutional layer inside a dense block is linked to every other layer within the block. This is accomplished by connecting the output of each layer to the input of the next layer, producing a “shortcut” link. The transition layers minimize the size of the feature maps across dense blocks that lets the network to grow effectively. |
| **VGG** | * **Accuracy:** VGG brought with it a massive improvement in accuracy and an improvement in speed as well * **Variety:** VGG brought with it various architectures built on the similar concept. * One of the main advantages of VGG is that it is a simple architecture that is easy to implement. | * **Slower:** VGG is slower than the newer ResNet architecture * **Vanishing gradient problem** * The complexity of an identical VGG network caused the degradation problem with respect to RESNET | * It is known for its simplicity and uniformity, using multiple convolutional layers with 3x3 filters and max-pooling layers for down sampling. * The input of VGG is set to an RGB image of 224x244 size. The average RGB value is calculated for all images on the training set image, and then the image is input as an input to the VGG convolution network. A 3x3 or 1x1 filter is used, and the convolution step is fixed. . * There are 3 VGG fully connected layers, which can vary from VGG11 to VGG19 according to the total number of convolutional layers + fully connected layers. The minimum VGG11 has 8 convolutional layers and 3 fully connected layers. The maximum VGG19 has 16 convolutional layers. +3 fully connected layers. * In addition, the VGG network is not followed by a pooling layer behind each convolutional layer, or a total of 5 pooling layers distributed under different convolutional layers |
| **INCEPTION** | * **Improved Information Flow:** The Inception architecture employs the concept of multi-scale feature extraction by concatenating the outputs from different convolutional operations. * **Efficient use of parameters** * **Computational Efficiency** | * **Computational Cost**   Despite being more efficient than previous architectures, the original Inception architecture still required significant computational resources to train and run.   * **Gradient Vanishing/Exploding** * **Limited Scalability**   The Inception architecture was designed for image classification tasks and may be less effective for other computer vision tasks, such as object detection or semantic segmentation. | * The Inception architecture, commonly known as GoogleNet, is a deep convolutional neural network (CNN) model with concurrent convolutional operations with varying receptive field sizes. It combats the vanishing gradient problem by employing 1x1 convolutions for dimensionality reduction and auxiliary classifiers. The design strives for a balance of computing economy and expressive power, making it appropriate for a wide range of computer vision tasks. * The concept of inception layers, forming a concatenated layer using stacks of 1x1 convolutions, 1x1 followed by 3x3 convolutions, 1x1 followed by 5x5 convolutions and 3x3 max pooling layers followed by 1x1 convolutions |