Vanishing Gradient Problem

The vanishing gradient problem is a significant challenge in training deep neural networks, especially when they are having many layers. It occurs when gradients (the error signal used to update the model's weights) becomes very small as they propagate back through the network during training. This phenomenon can severly hinder the learning process, as the weights in earlier layers of network receive minimal updates, effectively "freezing" those layers and making it difficult for the network to learn meaningful representations.

1) Backpropagation and Gradient Calculation ⇒ In training, gradients are propagated layer by layer. When they become very small, it hampers weight updates, especially in earlier layers, making learning difficult.

2) Causes of Vanishing Gradients ⇒ Small gradients arise from activation function like sigmoid / tanh (hyperbolic tangent), poor weight initialization and network depth, as each layer multiplies these small values, compounding the effect.

3) Impact on Training ⇒ Vanishing Gradients cause slow or stalled training, especially in deep networks, as early layers fail to learn crucial patterns, resulting in poor overall performance.

4) Solutions ⇒ (a) Using different activation function: Functions like sigmoid and tanh tend to squash input values in a narrow range. Sigmoid squashes the input between 0 and 1, and tanh squashes the input in range (−1 and +1). The derivative of these values are very small especially for extreme inputs which causes the vanishing gradient problem, hence we need to make use of activation function like **ReLU** (Rectified Linear Unit) which outputs 0 for negative inputs and the input values itself for the positive values. This function does not saturate in the positive domain, which helps maintain gradient flow. Variants like Leaky ReLU & Parametric ReLU allow for a small gradient when the input is negative, further helping gradient flow.

(b) **Batch Normalisation** ⇒ This technique normalizes the output of each layer to have a mean of zero and standard deviation one, which can reduce internal covariate shift (the changes in layer inputs during training). This helps stabilize the gradient and allows for faster training, which mitigates both vanishing and exploding gradients.

(c) **Weight Initialization Strategies** ⇒ Initializing weights properly can reduce the likelihood of gradients vanishing or exploding. Techniques like **Xavier (Glorot)** and **He** initialization are designed to keep the variance of gradients stable across layers, making the training process effective.

(d) **Residual Networks (ResNets)** ⇒ ResNets add shortcut connections or "skip connections" that bypass certain layers. This allows gradients to flow more directly to earlier layers, which improves the gradient flow and reduces the vanishing gradient problem.

(e) **Gradient Clipping** ⇒ While gradient clipping is more commonly associated with the exploding gradient problem, it can also help stabilize training in deep neural networks, particularly (RNNs). It involves setting a threshold for gradients, preventing them from becoming too large.

Early Stopping

The Early Stopping callback in Keras is a method used to prevent overfitting and improve training efficiency in deep learning models. It monitors a specified performance metric during training (typically validation loss) and stops training if this metric does not improve after a certain number of epochs, which is controlled by the '**patience**' parameter. Key parameters in Early-Stopping callback are—

(1) monitor ⇒ specifies the metric to watch (eg. 'val_loss')

(2) patience ⇒ Defines how many epochs to wait after the last improvement before stopping. If the monitored metric doesnot

improve after a certain number of epochs (within this period), training is halted.

(3) min_delta ⇒ Sets the minimum change to qualify as an improvement. If the change is smaller, it won't reset the patience counter.

(4) restore_best_weights ⇒ If True, restores model weights to the best epoch, making the model as effective as it was before overfitting began.

By stopping early, Early Stopping helps to reduce training time and mitigate overfitting, leading to a model that generalizes better on unseen data.