Exploding Gradient Problem and Vanishing Gradient Problem, why and how to avoid

Issues in Optimization process

- In the realm of deep learning, the optimization process plays a crucial role in training neural networks.
- Gradient descent, a fundamental optimization algorithm, can sometimes encounter two common issues: **vanishing gradients and exploding gradients.**

What is Vanishing Gradient?

- The vanishing gradient problem is a challenge that emerges during backpropagation when the derivatives or slopes of the activation functions become progressively smaller as we move backward through the layers of a neural network.
- This phenomenon is particularly prominent in deep networks with many layers, hindering the effective training of the model.
- The weight updates becomes extremely tiny, or even exponentially small, it can significantly prolong the training time, and in the worst-case scenario, it can halt the training process altogether.

Why the Problem Occurs?

- The vanishing gradient problem is particularly associated with the sigmoid and hyperbolic tangent (tanh) activation functions because their derivatives fall within the range of 0 to 0.25 and 0 to 1, respectively.
- Consequently, extreme weights becomes very small, causing the updated weights to closely resemble the original ones. This persistence of small updates contributes to the vanishing gradient issue.

Why the Problem Occurs?

- The sigmoid and tanh functions limit the input values to the ranges [0,1] and [-1,1], so that they saturate at 0 or 1 for sigmoid and -1 or 1 for Tanh.
- The derivatives at points becomes zero as they are moving. In these regions, especially when inputs are very small or large, the gradients are very close to zero.
- While this may not be a major concern in shallow networks with a few layers, it is a more pronounced issue in deep networks. When the inputs fall in saturated regions, the gradients approach zero, resulting in little update to the weights of the previous layer.
- In simple networks this does not pose much of a problem, but as more layers are added, these small gradients, which multiply between layers, decay significantly and consequently the first layer tears very slowly , and hinders overall model performance and can lead to convergence failure.

How can we identify?

- Identifying the vanishing gradient problem typically involves monitoring the training dynamics of a deep neural network.
- One key indicator is observing model weights converging to 0 or stagnation in the improvement of the **model's performance metrics** over training epochs.
- During training, if the **loss function** fails to decrease significantly, or if there is erratic behavior in the learning curves, it suggests that the gradients may be vanishing.
- Additionally, examining the gradients themselves during backpropagation can provide insights. **Visualization techniques**, such as gradient histograms or norms, can aid in assessing the distribution of gradients throughout the network.

How can we solve the issue?

- **Batch Normalization :** Batch normalization normalizes the inputs of each layer, reducing internal covariate shift. This can help stabilize and accelerate the training process, allowing for more consistent gradient flow.
- **Activation function:** Activation function like Rectified Linear Unit (ReLU) can be used. With ReLU, the gradient is 0 for negative and zero input, and it is 1 for positive input, which helps alleviate the vanishing gradient issue. Therefore, ReLU operates by replacing poor enter values with 0, and 1 for fine enter values, it preserves the input unchanged.
- **Skip Connections and Residual Networks (ResNets):** Skip connections, as seen in ResNets, allow the gradient to bypass certain layers during backpropagation. This facilitates the flow of information through the network, preventing gradients from vanishing.

How can we solve the issue?

- **Long Short-Term Memory Networks (LSTMs) and Gated Recurrent Units (GRUs):** In the context of recurrent neural networks (RNNs), architectures like LSTMs and GRUs are designed to address the vanishing gradient problem in sequences by incorporating gating mechanisms .
- **Gradient Clipping:** Gradient clipping involves imposing a threshold on the gradients during backpropagation. Limit the magnitude of gradients during backpropagation, this can prevent them from becoming too small or exploding, which can also hinder learning.

What is Exploding Gradient?

- The exploding gradient problem is a challenge encountered during training deep neural networks.
- It occurs when the **gradients of the network's loss function** with respect to the weights (parameters) become excessively large.
- The issue of exploding gradients arises when, during backpropagation, the derivatives or slopes of the neural network's layers grow progressively larger as we move backward. This is essentially the opposite of the vanishing gradient problem.

What is Exploding Gradient?

- The root cause of this problem lies in the **weights** of the network, rather than the choice of activation function.
- High weight values lead to correspondingly high derivatives, causing significant deviations in new weight values from the previous ones.
- As a result, the gradient fails to converge and can lead to the network oscillating around local minima, making it challenging to reach the global minimum point.

What is Exploding Gradient?

As we discussed earlier, the update for the weights during backpropagation in a neural network is given by:

$$\Delta W_i = -\alpha \cdot \frac{\partial L}{\partial W_i}$$

where,

- ΔW_i : The change in the weight W_i
- **α**: The learning rate, a hyperparameter that controls the step size of the update.
- **L**: The loss function that measures the error of the model.
- $\frac{\partial L}{\partial W_i}$: The partial derivative of the loss function with respect to the weight W_i, which indicates the gradient of the loss function with respect to that weight.

What is Exploding Gradient?

The exploding gradient problem occurs when the gradients become very large during backpropagation. This is often the result of gradients greater than 1, leading to a rapid increase in values as you propagate them backward through the layers.

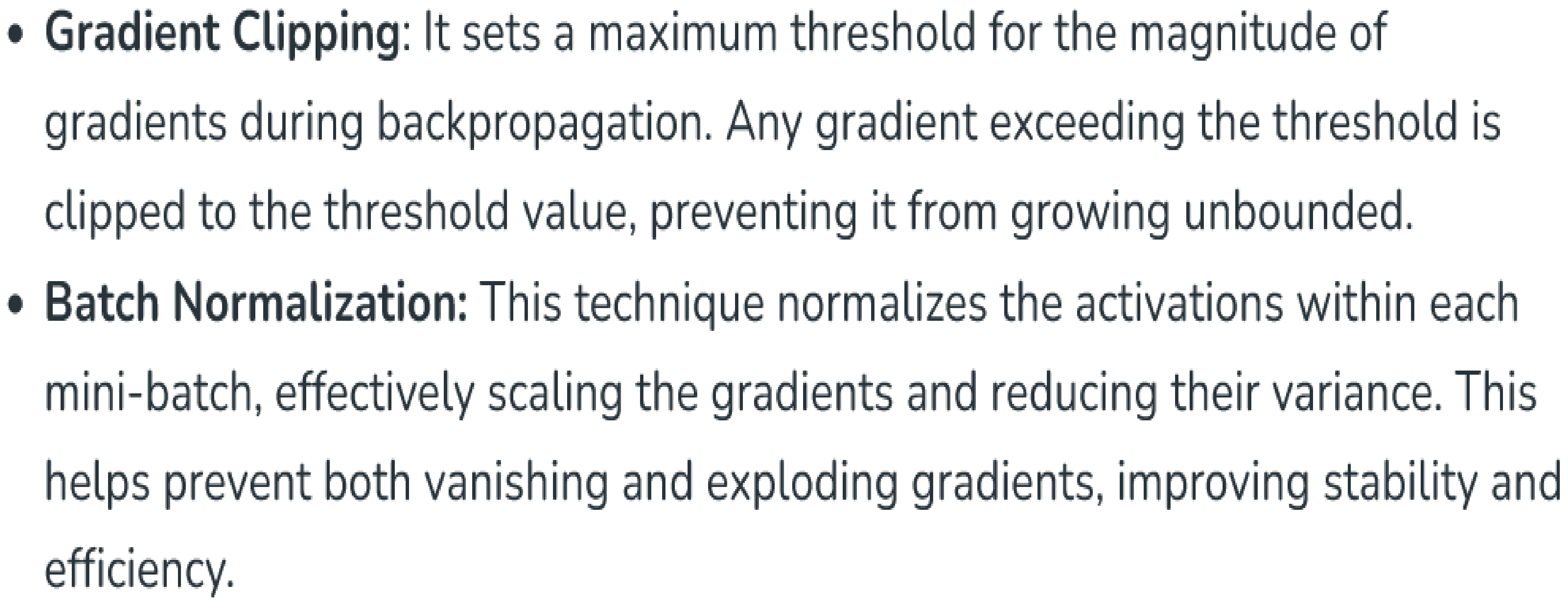
Mathematically, the update rule becomes problematic when $| \nabla W_i | > 1$, causing the weights to increase exponentially during training.

How can we identify the problem?

Identifying the presence of exploding gradients in deep neural network requires careful observation and analysis during training. Here are some key indicators:

- The loss function exhibits erratic behavior, oscillating wildly instead of steadily decreasing suggesting that the network weights are being updated excessively by large gradients, preventing smooth convergence.
- The training process encounters "NaN" (Not a Number) values in the loss function or other intermediate calculations..
- If network weights, during training exhibit significant and rapid increases in their values, it suggests the presence of exploding gradients.
- Tools like TensorBoard can be used to visualize the gradients flowing through the network.

How can we solve the issue?



- **Gradient Clipping**: It sets a maximum threshold for the magnitude of gradients during backpropagation. Any gradient exceeding the threshold is clipped to the threshold value, preventing it from growing unbounded.
- **Batch Normalization:** This technique normalizes the activations within each mini-batch, effectively scaling the gradients and reducing their variance. This helps prevent both vanishing and exploding gradients, improving stability and efficiency.