Batch Gradient Descent:

In Batch Gradient Descent, the entire training dataset is used to compute the gradient of the loss function with respect to the model weights. The weights are then updated in the opposite direction of the gradient:

$$\theta = \theta - \alpha \nabla J\theta$$

Where:

- θ is the weight vector.
- α is the learning rate.
- $\nabla J(\theta)$ is the gradient of the loss function $J(\theta)$ with respect to θ.

Stochastic Gradient Descent (SGD):

In Stochastic Gradient Descent, only a random data point is chosen to compute the gradient and update the weights:

$$\theta = \theta - \alpha \nabla J_i(\theta)$$

Where:

- i is the index of the randomly chosen data point.
- $\nabla J_i(\theta)$ is the gradient of the loss function $J(\theta)$with respect to θ based on the i-th data point.

Mini-Batch Gradient Descent:

In Mini-Batch Gradient Descent, a small number kk of randomly selected data points (mini-batch) is used to compute the gradient and update the weights:

$$\theta = \theta - \alpha \frac{1}{k} \sum_{i=1}^{k} \nabla J_i(\theta)$$

Where:

- k is the size of the mini-batch.
- $\sum_{i=1}^{k} \nabla J_i(\theta)$is the sum of gradients of the loss function $J(\theta)$with respect to θ over the k data points in the mini-batch.

RMSProp:

In RMSProp, the main idea is to adaptively change the learning rate (αα) for each parameter based on the magnitude of the gradient. The update formulas are described as follows:

$$s_{t+1} = \beta s_t + (1-\beta)(\nabla \mathrm{J}(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{s_{t+1}} + \epsilon} \nabla \mathrm{J}(\theta_t)$$

Where:

- s_t is the exponentially weighted moving average of squared gradients for each parameter at time step t.
- β is a decay factor.
- ϵ is a small constant to prevent division by zero.

Adam:

Adam combines ideas from both RMSProp and Momentum. It uses exponentially decaying averages for both the first moment (momentum) (v_t) and the second moment (squared gradient) (s_t). The update formulas for Adam are as follows:

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla \mathrm{J}(\theta_t)$$

$$s_{t+1} = \beta_2 s_t + (1-\beta_2)(\nabla \mathrm{J}(\theta_{\ t}))^2$$

$$\widehat{m}_{t+1} = \frac{m_{t+1}}{1-\beta_1^{t+1}}$$

$$\hat{s}_{t+1} = \frac{s_{t+1}}{1-\beta_2^{t+1}}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{s}_{t+1}} + \epsilon}$$

Where:

- m_t and s_t are the exponentially decaying averages of the gradient and squared gradient, respectively.
- B1 and β2 are decay factors.
- ϵ is a small constant to prevent division by zero.

Momentum:

The Momentum optimization method is one of the popular optimization techniques in machine learning. The update formula for the Momentum method is described as follows:

$$v_t = \beta v_{t-1} + (1 - \beta)\nabla \mathrm{J}(w_t)$$

$$w_{t+1} = w_t - \eta v_t$$

Where:

- w_t is the weight vector at time step t.
- $\nabla \mathrm{J}(w_t)$ is the gradient of the loss function at w_t.
- η is the learning rate.
- β is the momentum coefficient, typically set between 0 and 1.
- v_t is the momentum vector at time step t, computed based on the gradient and the previous momentum.

Continual Learning and Test Production in Machine Learning:

1. **Continual Learning:**
    - **Definition:** Continual Learning is a machine learning approach that enables a model to maintain its learning and adapt continuously when there are changes in the input data or task requirements.
    - **Challenge:** A major challenge in Continual Learning is catastrophic forgetting, where the model forgets previously learned knowledge when trying to learn new information. This poses a challenge when the model needs to adapt to new data without compromising its retention of old knowledge.
    - **Solution:** Methods such as Elastic Weight Consolidation (EWC), Learning Without Forgetting (LwF), and Progressive Neural Networks (PNN) have been developed to mitigate catastrophic forgetting during continual learning.
2. **Test Production:**
    - **Definition:** Test Production involves generating test sets to ensure that a machine learning model can evaluate its performance on new data and handle challenging scenarios.
    - **Importance:** Having a representative and diverse test set is crucial to ensure that the model not only memorizes the training data but also generalizes and applies knowledge to new situations.
    - **Strategies:** For Test Production, strategies such as Cross-Validation, Stratified Sampling, and Random Sampling are employed to create representative test sets and address issues like overfitting or underfitting on real-world data.

When building a machine learning solution, combining Continual Learning and Test Production ensures that the model not only has the ability to learn continuously but also can evaluate its performance on new and diverse data. This is crucial in real-world applications where the environment and requirements may change over time.