Question 1. What is the difference between adam optimizer and gradient descent?
Answer:
During training a neural network, the weights are updated using the optimization algorithms Adam optimizer and gradient descent. Yet there are some significant variations between them:
Update rule:
By subtracting a multiple of this gradient from the current weights, gradient descent calculates the gradient of the loss function with respect to the weights. The update step size is determined by the learning rate. The gradient of the loss function is also calculated by the Adam optimizer, but it does so using a more intricate update mechanism that includes a moving average of previous gradients and previous squared gradients.

Learning rate:
The learning rate in gradient descent is constant and the same for all weight changes. If the learning rate is not set properly, this may result in oscillations or sluggish convergence. In contrast, Adam optimizer adapts the learning rate for each weight based on the magnitude of the gradients and the history of the gradients. Faster convergence and improved performance may result from this.

Momentum:
By adding momentum, which builds up previous gradients to speed convergence in directions with a steady gradient, gradient descent can be improved. Similar to momentum, Adam optimizer's update rule incorporates this idea to smooth out updates and eliminate oscillations.

Regularization:
Adam optimizer includes L2 regularization by default, which can help prevent overfitting by adding a penalty term to the loss function that discourages large weights. Gradient descent can also be modified to include regularization, but it is not built into the algorithm by default.
Overall, Adam optimizer is a more advanced and adaptable optimization algorithm than simple gradient descent, but it may require more computational resources and can be more sensitive to the choice of hyperparameters.

Question 2. Define binary cross entropy as cost function.

Answer:

A frequent cost function in binary classification issues is binary cross-entropy. The difference between the true distribution of the binary target and the projected probability distribution is measured by the binary cross-entropy loss. Typically, the binary target is represented as a vector that has been one-hot encoded and has the values 0 or 1. A model, such as a neural network, that outputs a probability for each class can produce the anticipated probabilities.

The binary cross-entropy is defined as:

- (y * log(p) + (1 - y) * log(1 - p))

Here:

- y is the true binary target
- p is the predicted probability for the positive class

The degree to which the anticipated probability match the actual binary targets is gauged by the binary cross-entropy. A model is trained to produce probabilities that are as close as possible to the true binary targets by minimizing the binary cross-entropy.