Let x be the input to the dropout layer, and let p be the dropout probability (the probability of setting an input to zero). During training, each element xi of the input x is independently set to zero with probability p, and scaled by 1/(1-p) to maintain the expected value of the output.

Mathematically, the forward pass of the dropout layer can be represented as:

During training: yi = (xi \* Bernoulli(1-p))/(1-p)

Where Bernoulli(1-p) represents a random variable following the Bernoulli distribution with parameter 1-p.

During inference: yi = xi

During the backward pass, the gradient is simply scaled by the same mask used during the forward pass. Therefore, the backward pass can be represented as:

∂L/∂xi = (∂L/∂yi) \* maski

Where:

- L is the loss function.

- ∂L/∂yi is the gradient of the loss with respect to the output of the dropout layer.

- maski is the mask applied during the forward pass.

This ensures that only the non-zero activations contribute to the gradient flow during training.

During inference, the dropout layer is usually disabled, so yi = xi, and there is no scaling or masking applied.

These mathematical formulations capture the behavior of the dropout layer both during training and inference.

Batch normalization is a technique used in neural networks to normalize the inputs of each layer. It operates on a mini-batch of data during training and aims to stabilize and accelerate the training process.

Here's how batch normalization works:

1. For each feature in the mini-batch, calculate the mean and variance.

2. Normalize the features using the mean and variance.

3. Scale and shift the normalized features using learnable parameters (gamma and beta) to allow the model to learn the optimal scale and shift for each feature.

4. During training, these statistics (mean and variance) are calculated for each mini-batch and used to normalize the inputs.

5. During inference, a running average of the mean and variance calculated during training is used for normalization.

Mathematically, the batch normalization transformation can be represented as follows:

Given a mini-batch of activations x, the mean μ and variance σ^2 are calculated as:

μ\_B = (1/m) \* Σ(x\_i)

σ^2\_B = (1/m) \* Σ(x\_i - μ\_B)^2

Where m is the number of samples in the mini-batch.

Then, the normalized activations x̂\_i are obtained by:

x̂\_i = (x\_i - μ\_B) / sqrt(σ^2\_B + ε)

Where ε is a small constant added for numerical stability.

Finally, the output of the batch normalization layer is:

y\_i = γ \* x̂\_i + β

Where γ and β are learnable parameters, and y\_i is the output of the batch normalization layer.

This process helps to ensure that the activations of each layer in the network stay within a stable range during training, which can lead to faster convergence and better generalization.