Dropout ⇒ Dropout is a technique used during training in neural networks to prevent overfitting. It works by randomly "dropping out" or turning off a subset of neurons in each layer during each forward pass, meaning these neurons are temporarily ignored. This forces the network to learn more robust features because it cannot rely on any particular neuron or combination of neurons to make predictions.

Key points about dropout:

(a) Improves Generalization ⇒ Dropout helps prevent overfitting by reducing co-adaptations among neurons, encouraging the model to develop more generalized patterns.

(b) Only active in training ⇒ Dropout is typically only applied during training. During inference, all neurons are used, but their weights are scaled to reflect the dropout used during training.

In Keras, "dropout" layer is used to implement dropouts in neural networks. We can simply add "Dropout" layers in between our other layers, typically after fully connected (dense) or convolutional layers, to help reduce overfitting.

Key parameters in Dropout layers:

(1) rate (p): It specifies the fraction of input units to drop, which is float between 0 and 1. For Eg: p = 0.3 means 30% neurons in that layer will be randomly turned off during each training step.

(2) seed: This optional parameter ensures reproducability.

How dropout scaling works

During inference, all neurons are used, but their outputs are scaled to match the effect of dropout during training. The scaling is done by multiplying each neuron's output by the same probability used for dropout during training, effectively balancing the neuron's influence on the model's predictions. If $p = 0.5$, then their outputs are scaled by $(1-p)$ i.e., 0.5, (outputs are multiplied by 0.5). This adjustment ensures that the model's output during inference reflects the same overall distribution and strength of the activations as it saw during training.

Regularization

Regularization is a technique used in training neural networks to prevent overfitting, which happens when a model performs well on training data and poorly on new, unseen data. The main goal of regularization is to make the model generalize better by adding constraints or penalties that prevent it from relying too heavily on specific patterns in the training data.

(A) L1 regularization ⇒ The L1 regularized loss function includes a term. $\lambda * \|W\|_1$, where $\|W\|_1$ represents the sum of the absolute values of the weights. It shrinks the coefficients and makes few of them exactly 0, so those features associated with those weights are eleminated and thus it helps in feature selection and promotes sparsity.

(B) L2 regularization ⇒ The modified loss function with L2 regularization includes a term, $\lambda * \|W\|_2^2$, where λ is a hyperparameter that controls the penalty strength, and $\|W\|_2^2$ is the sum of squares of all weights. It also shrinks the weights but doesnot make them exactly zero.
Sometimes L2 regularization is also termed as "weight decay".

*In Keras, we can use kernel regularizers parameter along with Dense and Convolutional layers to apply regularization