◾ Loss Function :

➢ A loss function, also known as a cost function (or objective function), is a mathematical function that measures how well a model's predictions match the actual target values (ground truth) for a given set of input data.

➢ It is the method ,how error is calculated.

(Loss function vanda error / loss kasari calculate garni vannin bujhinxa)

➢ Some loss functions are :-

- **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and true values in regression tasks.
- **Mean Absolute Error (MAE)**: Measures the average absolute difference between predicted and true values in regression tasks.
- **Binary Cross-Entropy (Log Loss)**: Quantifies dissimilarity between binary true labels and predicted probabilities in binary classification.
- **Categorical Cross-Entropy**: Measures dissimilarity between true and predicted class distributions in multi-class classification.

(Links For more information about more loss functions :-
https://www.tensorflow.org/api_docs/python/tf/keras/losses
https://pytorch.org/docs/stable/nn.functional.html#loss-functions)

◾ Choosing an optimizer:

➢ The

optimizer is responsible for updating the model's parameters (weights and biases) during training to minimize the loss function.

➢ It does this by computing gradients, which indicate the direction and magnitude of parameter adjustments needed to reduce the loss.

➢ Some of the optimizers are :-

- **Stochastic Gradient Descent (SGD)**: Basic optimizer that updates model parameters using gradients.
- **Adam (Adaptive Moment Estimation)**: Popular optimizer combining adaptive learning rates and momentum.

- **RMSprop (Root Mean Square Propagation)**: Adapts learning rates based on recent squared gradients.
- **Adagrad (Adaptive Gradient Algorithm)**: Adapts learning rates for each parameter based on historical gradients.

➤ Documentation links for optimizers:-

1. Tensorflow optimizers
2. Pytorch optimizers
3. Keras optimizers

◾ Learning rate :

➢ The learning rate is a hyperparameter that determines the step size used by the optimizer when updating the model's parameters.

➢ A high learning rate can lead to large parameter updates, which may result in overshooting and convergence issues. (overfitting)

➢ A low learning rate can cause slow convergence or getting stuck in local minima. (underfitting)

➢ Finding an appropriate learning rate is crucial for training success. It's often chosen through hyperparameter tuning.