Neural Network model.compile() parametres in detail

In TensorFlow/Keras, the model.compile() function is used to configure the model's training process. It requires specifying three key parameters:

1. **Optimizer**: Controls how the model learns by updating the weights.
2. **Loss**: Defines the objective function to minimize during training.
3. **Metrics**: Monitors additional performance measures like accuracy during training and evaluation.

1. Optimizer:

The optimizer decides how to update the model's weights based on the loss value. Common optimizers include:

S. NO	Optimizer	When to Use	Description
1	**adam**	Default choice for most tasks	Combines advantages of AdaGrad and RMSProp; adapts learning rates per parameter.
2	**sgd**	Simple, large datasets, quick convergence	Stochastic Gradient Descent; updates weights by moving in the opposite direction of the gradient.
3	**rmsprop**	For noisy, non-stationary data	Root Mean Square Propagation; adapts learning rate based on a moving average of squared gradients.
4	**adagrad**	For sparse data (e.g., text data)	Adapts learning rate based on the frequency of parameters; good for sparse data.
5	**adadelta**	For more adaptive learning rate updates	Variant of AdaGrad that aims to reduce its learning rate decay problem.
6	**adamax**	For very large models	Variant of Adam based on the infinity norm, less sensitive to large gradients.
7	**nadam**	For faster convergence	Adam combined with Nesterov Momentum; adjusts updates based on the momentum of the previous updates.

2. Loss:

The loss function measures how well the model's predictions match the true values. It guides the optimizer to adjust weights to minimize this value.

S. No	Loss Function	When to Use	Description
1	**sparse_categorical_crossentropy**	For classification problems with integer-encoded labels (e.g., MNIST digit classification)	Use when target labels are integers (e.g., 0, 1, 2).
2	**categorical_crossentropy**	For classification problems with one-hot encoded labels	Use when target labels are one-hot encoded (e.g., [0, 1, 0]).
3	**binary_crossentropy**	For binary classification problems (e.g., spam detection)	Computes loss for binary labels (0 or 1).
4	**mean_squared_error**	For regression tasks (predicting continuous values)	Computes the average squared difference between predictions and actual values.
5	**mean_absolute_error**	For regression tasks with less sensitivity to outliers	Computes the average absolute difference between predictions and actual values.
6	**hinge**	For binary classification with SVM-like decision boundaries	Loss function used for Support Vector Machines.

3. Metrics:

Metrics are used to evaluate the model's performance during training and testing. They don't influence the training process but provide insights.

S. NO	Metric	When to Use	Description
1	**accuracy**	For classification tasks	Measures how often predictions match labels.
2	**binary_accuracy**	For binary classification tasks	Measures accuracy for binary (0 or 1) predictions.
3	**categorical_accuracy**	For multi-class classification with one-hot labels	Measures accuracy for multi-class, one-hot encoded labels.
4	**mean_squared_error**	For regression tasks	Measures the average of the squared errors.
5	**mean_absolute_error**	For regression tasks	Measures the average of the absolute errors.
6	**AUC**	For evaluating binary classifiers	Computes the Area Under the ROC Curve.
7	**Precision**, Recall	For evaluating precision and recall in classification tasks	Precision measures true positives among predicted positives; recall measures true positives among actual positives.

Conclusion

Choosing the Right Parameters:

1. **Optimizer**:
   Use **Adam** for most cases as it's adaptive and works well across many tasks.
   Use **SGD** with a manual learning rate when you need more control over training.
   For very noisy data, try **RMSProp** or **Adagrad**.

2. **Loss**:
   Use **sparse_categorical_crossentropy** if your labels are integer-encoded and you have a multi-class classification task.
   Use **mean_squared_error** or **mean_absolute_error** for regression tasks.

3. **Metrics**:
   Use **accuracy** for a general measure of performance in classification tasks.
   Use **AUC**, **precision**, or **recall** for imbalanced datasets.