Detailed Notes on Loss Functions in TensorFlow

What is a Loss Function?

A **loss function** in deep learning is a mathematical function that calculates the difference between the predicted output of the model and the actual target (true label or value). It guides the learning process by telling the model how wrong it is and how it should adjust its internal weights to improve.

---

1. Sparse Categorical Crossentropy

```
from tensorflow.keras.losses import SparseCategoricalCrossentropy
```

When to Use:

- Multi-class classification problems
- Target labels are integers (e.g., 0, 1, 2, ...)

Example Use Case:

- Classifying types of animals (cat=0, dog=1, horse=2) in an image

Explanation:

It computes the cross-entropy loss between the true labels and the predicted probabilities. It is preferred when labels are not one-hot encoded.

Important Parameter:

- `from_logits=True` if predictions are raw scores (logits)

Example:

```
loss = SparseCategoricalCrossentropy(from_logits=False)
y_true = [2]
y_pred = [[0.1, 0.3, 0.6]]  # Predicted probabilities
loss(y_true, y_pred).numpy()  # Output: 0.5108
```

---

2. Categorical Crossentropy

```
from tensorflow.keras.losses import CategoricalCrossentropy
```

When to Use:

- Multi-class classification problems
- Target labels are **one-hot encoded** (e.g., [0, 0, 1])

Example Use Case:

- Handwritten digit classification (one-hot encoded digits)

Explanation:

It calculates the cross-entropy between the one-hot encoded labels and predicted probabilities. Used when your dataset has categorical variables as output.

Example:

```
loss = CategoricalCrossentropy()
y_true = [[0, 0, 1]]
y_pred = [[0.1, 0.3, 0.6]]
loss(y_true, y_pred).numpy()  # Output: 0.5108
```

---

3. Binary Crossentropy

```
from tensorflow.keras.losses import BinaryCrossentropy
```

When to Use:

- Binary classification (output is 0 or 1)
- Multi-label classification (each label is binary)

Example Use Case:

- Spam email detection (spam=1, not spam=0)

Explanation:

Calculates loss between binary true labels and predicted probabilities. Commonly used in logistic regression and binary classifiers.

Example:

```
loss = BinaryCrossentropy()
y_true = [[1], [0]]
y_pred = [[0.9], [0.1]]
loss(y_true, y_pred).numpy()  # Output: ~0.105
```

---

4. Mean Absolute Error (MAE)

```
from tensorflow.keras.losses import MeanAbsoluteError
```

When to Use:

- Regression problems

- You want equal weight for all errors

Example Use Case:

- Predicting house prices

Explanation:

It calculates the average of absolute differences between actual and predicted values. Less sensitive to outliers.

Example:

```
loss = MeanAbsoluteError()
y_true = [2.0, 4.0]
y_pred = [2.5, 3.0]
loss(y_true, y_pred).numpy()  # Output: 0.75
```

---

5. Mean Squared Error (MSE)

```
from tensorflow.keras.losses import MeanSquaredError
```

When to Use:

- Regression problems
- You want to penalize larger errors more heavily

Example Use Case:

- Forecasting temperature, sales, or any continuous value

Explanation:

It calculates the average of squared differences between predicted and actual values. Squaring the error increases the penalty for large deviations.

Example:

```
loss = MeanSquaredError()
y_true = [2.0, 4.0]
y_pred = [2.5, 3.0]
loss(y_true, y_pred).numpy()  # Output: 0.625
```

Summary Comparison Table

Loss Function	Use Case	Label Type	Notes
SparseCategoricalCrossentropy	Multi-class classification	Integer	Faster & memory-efficient

Loss Function	Use Case	Label Type	Notes
CategoricalCrossentropy	Multi-class classification	One-hot encoded	Needs categorical labels
BinaryCrossentropy	Binary or multi-label	0 or 1	Uses sigmoid; classification
Mean Absolute Error (MAE)	Regression	Float	Robust to outliers
Mean Squared Error (MSE)	Regression	Float	Penalizes large errors more

Final Tip

While compiling a model in TensorFlow:

```
model.compile(
    optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

Use `from_logits=True` if the model does **not** have softmax or sigmoid activation at the output. if the model does **not** have softmax or sigmoid activation at the output.

If you'd like, I can also help you visualize these with graphs or share practical datasets for practice.