QUESTION:

What is the difference between sparse\_categorical\_crossentropy and categorical\_crossentropy in Keras?

ANSWER:

In Keras, both sparse\_categorical\_crossentropy and categorical\_crossentropy are loss functions commonly used in training neural networks, particularly for classification tasks. The choice between them depends on how your labels are represented.

1. Categorical Crossentropy (categorical\_crossentropy):

This loss function is used when your targets are one-hot encoded. One-hot encoding means that each label is represented by a binary vector where all elements are zero except for the index corresponding to the class, which is marked with a 1.

Example:

Class 0: [1, 0, 0]

Class 1: [0, 1, 0]

Class 2: [0, 0, 1]

This is typically used when you have mutually exclusive classes, and each sample belongs to exactly one class.

2. Sparse Categorical Crossentropy (sparse\_categorical\_crossentropy):

This loss function is used when your targets are integers. In other words, your target for each sample is just the class index, not a one-hot encoded vector.

Example:

Class 0: 0

Class 1: 1

Class 2: 2

This is useful when your classes are mutually exclusive, but you don't want to or can't one-hot encode your labels. This could be the case if you have a large number of classes and one-hot encoding would lead to memory issues, or if you're dealing with natural language processing tasks where your labels are words or word indices.

In summary, use categorical\_crossentropy when your labels are one-hot encoded vectors and sparse\_categorical\_crossentropy when your labels are integer indices. Both loss functions aim to minimize the difference between the true distribution and the predicted distribution. Which one to use depends on how your labels are represented in your dataset.

There is one subtler difference between categorical\_crossentropy and sparse\_categorical\_crossentropy, which lies in their input format.

1. Categorical Crossentropy (categorical\_crossentropy):

Expects both the true labels and the predicted probabilities to be provided as 2D tensors.

The true labels are expected to be in one-hot encoded format.

The predicted probabilities are expected to be outputted by the softmax activation function, which ensures that the predicted probabilities sum up to 1 for each sample.

2. Sparse Categorical Crossentropy (sparse\_categorical\_crossentropy):

Expects the true labels to be provided as 1D tensors containing integer class indices.

The predicted probabilities are still expected to be provided as 2D tensors, but the true labels are not required to be one-hot encoded.

Like categorical\_crossentropy, the predicted probabilities are typically outputted by the softmax activation function.

In summary, while the primary difference between these loss functions is in how they handle the format of true labels (one-hot encoded vs. integer indices), there's also a difference in the input format they expect, with sparse\_categorical\_crossentropy allowing for simpler representation of true labels as integer indices.

Example:

Suppose you have a classification task with three classes: "cat", "dog", and "bird". You have the following dataset:

Sample 1: Image of a cat, labeled as "cat"

Sample 2: Image of a dog, labeled as "dog"

Sample 3: Image of a bird, labeled as "bird"

Now, let's represent the labels in two different ways:

One-Hot Encoded Labels:

"cat" is represented as [1, 0, 0]

"dog" is represented as [0, 1, 0]

"bird" is represented as [0, 0, 1]

Integer Indices Labels:

"cat" is represented as 0

"dog" is represented as 1

"bird" is represented as 2

Using Categorical Crossentropy (categorical\_crossentropy):

from keras.losses import categorical\_crossentropy

import numpy as np

# True labels (one-hot encoded)

true\_labels\_one\_hot = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])

# Predicted probabilities (softmax output)

predicted\_probs = np.array([[0.9, 0.05, 0.05], [0.05, 0.9, 0.05], [0.05, 0.05, 0.9]])

# Compute loss

loss = categorical\_crossentropy(true\_labels\_one\_hot, predicted\_probs)

print("Categorical Crossentropy Loss:", loss)

Using Sparse Categorical Crossentropy (sparse\_categorical\_crossentropy):

from keras.losses import sparse\_categorical\_crossentropy

import numpy as np

# True labels (integer indices)

true\_labels\_indices = np.array([0, 1, 2])

# Predicted probabilities (softmax output)

predicted\_probs = np.array([[0.9, 0.05, 0.05], [0.05, 0.9, 0.05], [0.05, 0.05, 0.9]])

# Compute loss

loss = sparse\_categorical\_crossentropy(true\_labels\_indices, predicted\_probs)

print("True labels (indices):", true\_labels\_indices)

print("Predicted probabilities:\n", predicted\_probs)

print("Sparse Categorical Crossentropy Loss:", loss)