Today, I finished the eighth tutorial on Deep Learning, which focused on using TensorFlow 2.0 and Keras to create a Neural Network for recognizing handwritten digits. Upon completing this tutorial, I was tasked with improving the model by experimenting with various loss functions. The original LLM provided in this tutorial, which performed accuracy of 0.9755, was constructed using:

🔸 Dataset: MNIST. A Large database of handwritten digits that is commonly used for training various image processing system.

🔸 Training set: 60,000 28 x 28 gray scale images of the handwritten 10 digits.

🔸 Test set: 10,000 28 x 28 gray scale images of the handwritten 10 digits.

🔸 Hidden layers: Only one hidden layer in this model. Set to 100 output, 786 input shape (flattened 28x28 two dimensional array), ReLu activation function.

🔸 Output layer: Set to 10 output, 786 input shape (flattened 28x28 two dimensional array), Sigmoid activation function.

🔸 Compiler: adam

🔸 Epochs: 10

🔸 Loss function: sparse\_categorical\_crossentropy

🔸 Metrics: accuracy

1. binary\_crossentropy (0.9797): I initially chose not to experiment with binary\_crossentropy as it is typically used for binary classification (where the output is 0/1, or TRUE/FALSE), and thus requires the y\_test value to be either 0 or 1. Given that the original model in this tutorial involves 10 classifications with y\_test values ranging from 0 to 10, I initially deemed this loss function inappropriate. However, upon testing, the Binary loss function series surprisingly outperformed others, achieving an accuracy of 0.9797  In fact, this was the highest accuracy achieved.
2. A screenshot of a graph

   Description automatically generated
3. binary\_focal\_crossentropy (0.9794):  This is an improved version of binary\_crossentropy designed to address the class imbalance problem. Despite my expectation that it would perform better than binary\_crossentropy, it did not. The accuracy dropped slightly to 0.9794.

A screenshot of a graph

Description automatically generated

1. categorical\_crossentropy (0.9792): Initially, I experimented with this loss function, similar to the original one, which was also categorical (i.e., there is one or more classes). This function achieved an accuracy of 0.9792, which is better than the original model’s performance.

I encountered an error message: ValueError: Shapes (32, 1) and (32, 10) are incompatible. This occurred because the categorical\_crossentropy loss function requires the target data (y\_train and y\_test) to be one-hot encoded. If the target data isn’t one-hot encoded, the sparse\_categorical\_crossentropy loss function should be used, as demonstrated by Dhaval Patel in the tutorial. Consequently, I had to convert y\_train and y\_predict to one-hot encoding to utilize the categorical\_crossentropy loss function.

A screenshot of a graph

Description automatically generated

1. categorical\_focal\_crossentropy (0.9764): The model achieved an accuracy of 0.9764. Similar to the comparison between binary\_focal\_crossentropy and binary\_crossentropy, categorical\_focal\_crossentropy performed slightly worse than categorical\_crossentropy. This is noteworthy as categorical\_focal\_crossentropy is designed to adjust for class imbalance.

A screenshot of a graph

Description automatically generated

1. categorical\_hinge (0.9750):This loss function is designed to handle cases where the classes are not linearly separable. Accurary is 0.9750.

A graph of numbers and a graph

Description automatically generated with medium confidence

1. cosine\_similarity (0.970): Computes the cosign similarity between the labels and predictions. Accuracy is 0.970.

A screenshot of a graph

Description automatically generated