Data Preparation

A GPU, instead of the CPU, is utilised in the computations for speedier execution. This is because a GPU is built to generate high-resolution pictures and video all at the same time, and that they are optimal for developing AI and neural network models since they can handle numerous calculations instantly. Since the CPU acts as a portal between the sources and the GPU's components, we can't eliminate anything from the ML setups. The quicker the data flows, the faster the computations are. We use the Nvidia GTX 16 series of GPUs (1660 TI) with 6 GB of graphics in all computations, even though the efficiency varies from one GPU to another. We tried utilising Nvidia's GTX 1650 Ti, which has 4 GB of graphics, though the earlier works brilliantly with 6 GB of graphics and has superior overall effectiveness.

In our network, we used TensorFlow release 2.5.0, and we also added a number of critical modules, including the Nvidia Graphics, the Cuda Toolkit (v.11.0), cuDNN (v.8.2), and Tensorflow (v.2.5.0). The latest release of Tensorflow that is compatible with that of the Cuda library is 2.5.0, and this release provides the best overall accuracy. Our final dataset comprises three input types, namely rock, paper, and scissors, and 2000 photos each. The dataset is saved in a particular class list that includes just about all the JPEG images that were gathered when the process ended.

We give particular emphasis to the data's format, which is (3, 2000, 96, 96, 3), with 3 denoting the total classes, 2000 denoting the number of photographs in each, 96x96 denoting the image's dimension, and 3 denoting the RBG stream. The photos are therefore stored in a variable named "X" that contains a collection, while the class indices are stored in "Y", which are two separate list variables. The entire arrays are constructed, with the said form of "X" being 6000, 96, 96, 3, and the size of 'Y' being 6000. If we attempt to examine the 2456th picture in 'Y,' for instance, we will see the 456th picture of the 2nd class, i.e., 'Y' = '1' = Paper. If we divide it by 255, it also helps us to feed the computational model with a normalised "X".