Flow of the program:

Input Data:
Dataset consists of 2000 labeled images, such that each image had more than one labels (multi-labeled images)

Problem Description:
This problem is a case of "Multi-Instance, Multi-label learning for Scene Classification", as mentioned in a paper published by Zhi-Hua Zhou and Min-Ling Zhang.

One approach is of the MIML algorithm as mentioned in the above paper, ie, Multi-Instance Multi-Label. We know that every image has an average of 1.24 labels, so effectively we can have a single label for every image by increasing the input data by 2000 * 1.24 = 2480 images, such that in this case, every image has a single label. So essentially, creating a bag of labels to have multi-label learning as the bridge, ie, create 5 different models for each of the labels, i.e one vs the rest, and then later combine the results of these labels together.

The other approach is directly taking the label vectors for each instance as a multi-hot encoded labels having the presence of 1 at multiple indices and feed it into a CNN to obtain the results - a straightforward approach. I continued with using this approach.

Reading Data and Pre-processing:
- Convert image to grayscale (color channel dimension hence reduced from 3,ie, RGB to 1)
- Resize the images to 128 X 128.
- Normalize the image pixel values.

Building the CNN Architecture:
Created two convolutional and pooling layers, and then flattened it and connected it to a fully connected dense layer after which connected to a dropout layer and finally to an output layer, which returned the logits values for the five neurons in the output layer.

Loss Function of CNN:
This is the main differentiating point in the entire CNN architecture. For a simple single label images with multi-classes, we would always use the softmax layer with the logits as the input and use a categorical cross-entropy loss for that. What softmax does it that it takes the logits values, and squishes/normalizes the logits, such that the sum of the probabilities of the output layer is always 1. That is to say that it will always return that label/class which has the maximum

probability amongst the rest of the classes. So this has to be strictly used for just a one-label classification

But this being a multi-labeled scenario, softmax won't work, and hence cannot decide a threshold value as well to indicate that if the probabilities go beyond a certain value, need to include that class. So, used a sigmoidal function with a binary cross entropy loss, which will give independent probabilities for each class, so that sum of the probabilites of all the labels/classes are not necessarily constrained to summing to 1. Every class/label has its own individual and independent cross-entropy function.

Model Training:
The training was done in batches of size 250. Calculated the train accuracy, validation/test accuracy and the loss value, and printed it for the epochs at the steps of 5 epochs. The backpropogation was done by the Gradient Descent algorithm. The accuracy achieved in the initial values of the epoch(5 to 10) was about 75%.