**Model architecture:** I have implemented MobileNetV2 model architecture after reading the paper on the same and some blog post where authors explained the architecture in more details

**Loss Function:** I used cross entropy loss of pyTorch (torch.nn.CrossEntropyLoss)

**hyper parameters:**

BottleneckResBlock\_config = [

# t, c, n, s as mentioned in Mobilenet v2 paper

[1, 16, 1, 1],

[6, 24, 2, 2],

[6, 32, 3, 1],

[6, 64, 4, 1],

[6, 96, 3, 1],

[6, 160, 3, 2],

[6, 320, 1, 1],

]

“MobileNetV2 : Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. All spatial convolutions use 3 × 3 kernels. The expansion factor t is always applied to the input size.”{ As mentioned in paper https://arxiv.org/pdf/1801.04381.pdf }

Number of Epochs = 20  
learning Rate = 1e-3  
weight Decay = 1e-4  
Optimizer: Adam

**Interesting detail:**

I used results of last 5 best epochs and selected the results that for each image that appeared in most of the results file and created an ensemble file as a final submission, this improved my results by more than 5% and helped me cross the A grade benchmark in classification task.

For verification task I changed my data loader to return the path of the image along with image file. Then I generated embedding for all the images at once and stored it in one variable. Then while reading the output file, I just calculated the similarity between the saved embedding and appended the result. This approach reduced significantly the computation as compared to reading the inputs images from the output file one by one, and then generating the embedding.