1. train.py

It has 2 classes, *AlexNet()* and *train_self_model*.

AlexNet(nn.Module). It sets the basic structure and operations of AlexNet, except that in the last layer the output is 200. Its method *forward()* performs the forward operation of AlexNet.

train_self_model(). It performs training and validation of self-defined AlexNet. In *__init()__*, it classifies the validation images into different folders. Also get the class labels of the training data. Moreover, it also load the data, set the train/validation batch sizes, set the hyper parameters like learning rate, loss function, and optimizer. The *train()* method performs training and validation for the self-defined AlexNet. The process is, for each epoch, to perform *forward -> backward -> update* for each training batch of all training batches, and then perform validation for each validation batch. The whole process is repeated for *max_epoch* times.

I only have CPU on my laptop, and it takes more than 1 hour to train the network for one epoch. When I finished the code I found myself have no time to finish the whole training process.

In saving model, I used the function *'torch.save(self.state_dict(), /dir/to/save/model/)'.*

2. test.py

It's used to load the trained model and to grab continuous frames from camera, get the network prediction for respective frames, and then display the frame with the prediction overlayed on it.

The function to load the trained model is '*myAlexNet.load_state_dict(torch.load(/dir/containing/model/))*'

cam(): it fetchs images from the camera, and continuously outputs the predicted result.

One output image is