1. **Supervised learning**

   Without using Dense and Activation, I use the Convolution and Maxpooling method. In this method, I first import the needed function from keras and the data given. First, I create a Sequential called model, then adding convolution(256,3,3,) and Maxpooling alternatively, and sometimes add Dropout. After the last Maxpooling, I use 'relu' and flat the model. Next, adding Dense(1024) and Activation('sigmoid') and make the last Dense(10), which indicate our model's output and softmax it.
   Then begin compile, where the parameter loss=' categorical_crossentropy' and optimizer='Adadelta' and print the accuracy. Besides, I import x_train and y_train and shuffle them together, in case the local minimum. Before running the epoch, I set the ModelCheckpoint and earlystopping, which only save the best model, and stop if the model is not improving in a while. I also add regulization, but it seems useless for no accuracy increasing. And I fit the x_train and y_train for batch_size=100 , nb_epoch=60 , validation_split =0.2, and callbacks of check. After that, I evaluate the model and print the result, which can have a val_acc of 0.6, and save the model by model.save(filepath). Other function I add after are Gaussian noise, BatchNormalization.

2. Semi-supervised learning I

   In this work, I load the unsupervised learning model and begin to predict the unlabel data with the predict function. I set a threshold about 0.95, if the probability of an unlabel data to a specific class is higher than the threshold, I consider it a label data, and concatenate to the label data, x_train. I use amax to find the maximum value in every sample, and its argument, and I add the unlabel data's argument into y_train, and make their dimension the same and begin fitting.
   After fitting, I only save the best model, and right now I finish one iteration. In the next iteration, I do it again, predict the unlabel data and add the high probability one into label data. But it take me so long to finish one iteration. From 4pm to 9pm only two iteration.

3. Semi-supervised learning II

   I use autoencoder to do the method two. In the encoded stage, I use convolution and maxpooling to encode my imagine. Then, I use convolution and upsampling to decode my encoded data. After that I create a model to input and output the encoded and decoded data, and compile with adadelta and categorical_crossentropy. After reshaping and changing the data type to float and divide by 255., adding some noise figure to my x_train, and fit the x_train and add some parameter such as shuffle, validation_split ,callbacks etc… then after encoding, I have a .h5 file call encode_model, and I take this model to go semi-supervised learning. Loading and opening data and model, I use the model to predict the unladel data's probability to one class. And I set a threshold, once the probability is higher, it will become label data. And I have the iteration to semi-suprevised.

4. Compare and analyze your results

My unsupervised learning can about 0.6 of val_acc, but I have a few problem in semi-supervise, no matter in method one or two. I consider, in my first model, there is too many parameters, so even one iteration it can run as lone as perhaps two hous, and it lose the real meaning of semi due to its inefficiency. Thus, I use less convolution and maxpooling layers and make the model concise. The processing is much faster, I can finish 70 epochs within 3mins compare to an 20mins epoch. Although the unsupervised model' result may not seen good, however it can go faster in semi-supervised model learning. On the other hand, autoencoder is much harder. Though I can have a good performance on encoding and decoding for about val_acc = 0.6,that good for me, but it become a mistake when I load this trained model to predict the unlabel date, the accuracy and val_accuracy both seen like it goes to a local minimum, but seen every time go to the place. After, I modify my model, which it can learn better than before, but still poor in performance.

Compare these training method, I thick unsupervised is the most easiest way to have a good but not great model, and in reality there is no such many "label data" for the model to learn. It's simple and fast but not practical. And the semi-supervised method I, it seems like the machine can learn by itself, but takes a much more time to train and learn. Though it has higher accuracy practically, but my model does not performance well even when I write my report. And the second semi-supervised method, I did not use K-means, instead, I put the encoded-decoded model to predict unlabel data and do lots of iteration to let it train by itself.