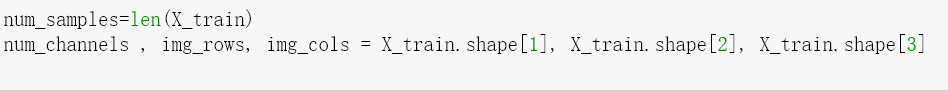
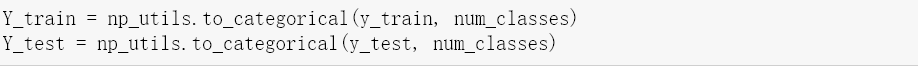
Question 1:

Download the Tiny Imagenet data which has 200 classes and each class has 500 images, 50 validation images and 50 test images. Firstly we changed the size of images to 32X32 format with these code:



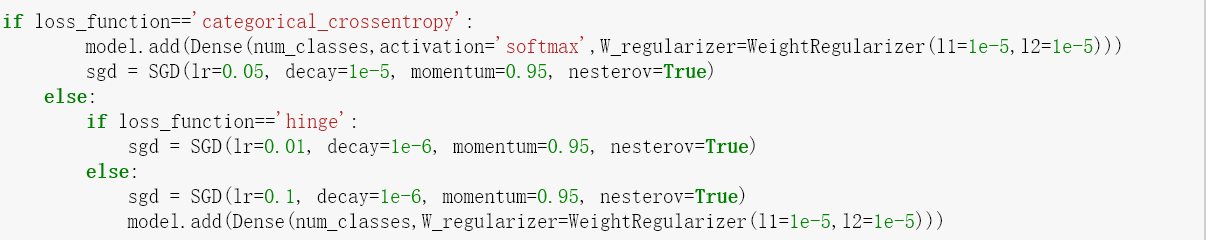
We used “load\_img” to load and split the data into training and test, and then wrangle the data:

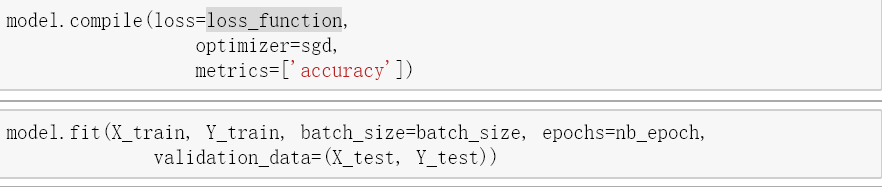
 Then, I set three kinds of loss\_functions like this:

loss\_functions = ['categorical\_crossentropy','squared\_hinge','hinge'], and design a base model with original parameters.

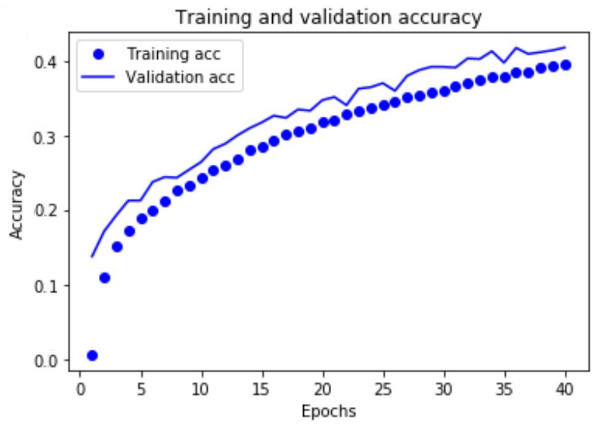


Then I used a for loop to get the best loss\_function, with which to fit a model:





Then plot the score of the model like this:



Finally the training accuracy for the model is 0.3992, and the test accuracy is 0.3723. Somehow, this is not a best model for this dataset, but if we continue tune the parameters, we can get a better model.

Question 1&2:

1. Input data:

Download the Tiny Imagenet data which has 200 classes and each class has 500 images, 50 validation images and 50 test images. Firstly we changed the size of images to 32X32 format with these code:



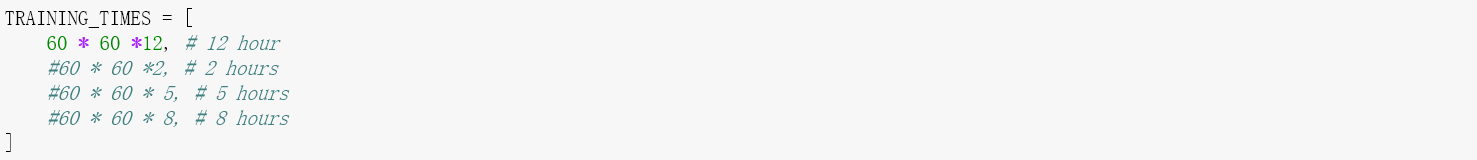
2. Wrangling data:

I made some experiments with Colab to calculate the computation time for this dataset, however the total time is too much for our project. So, we decided to choose part of the training data to do the assignment. We choose 100 images per class to be training set, so the total training image is 20,000. And we set 30 images per class to be validation data, 30 images each class for testing.



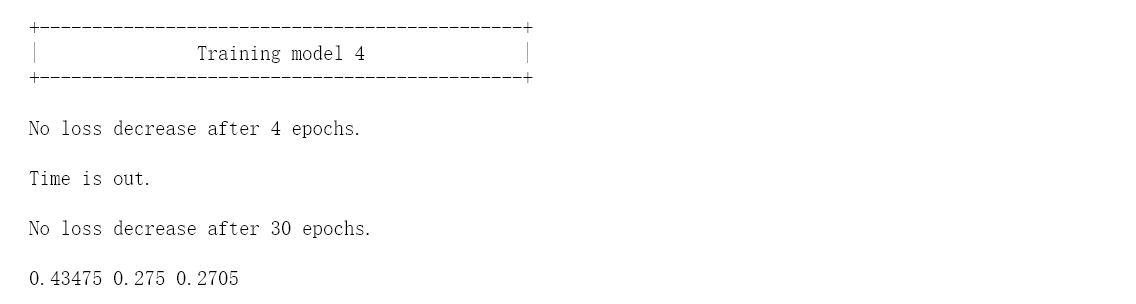
3. Then we fit a model with ImageClassifier and adjust the parameters for the model. This classifier has three kinds of parameters: (1) Searcher: MAX\_MODEL\_NUM, BETA, KERNEL\_LAMBDA, T\_MIN, N\_NEIGHBOURS, MAX\_MODEL\_SIZE, (2) Model Defaults: DENSE\_DROPOUT\_RATE, CONV\_DROPOUT\_RATE, CONV\_BLOCK\_DISTANCE, DENSE\_BLOCK\_DISTANCE, MODEL\_LEN, MODEL\_WIDTH, (3) model trainer: DATA\_AUGMENTATION, MAX\_NO\_IMPROVEMENT\_NUM. MAX\_ITER\_NUM, MIN\_LOSS\_DEC.

But for our case, we just need to adjust a part of the parameters of max\_no\_impr- ovement\_num, and max\_iter\_num, and the time limit.

So we run the auto model with the most iteration number being 15, which means the epochs for each model is at most 15. And if one spcific model doesn’t change the accuracy four consecutive times, than move to the next model. And for the whole auto running process, we should set max running time to get the optimal model in a limited time due to the realistic situation. The “clf.fit” trained out a relative optimal model with optimal structure, and then use “clf.final\_fit” to further train better parameters of the model based on the optimal structure.

At last, auto keras output four model, and the best model with accuracy like this:



Traing\_accuracy: 0.43474, validation\_accuracy: 0.275, test\_accuracy: 0.2705, which is better than problem 1 we got, and these accuracies make sense for the realistic problems. So this model obtained from auto-keras is relatively good for the dataset.