# Kaggle Competition Report

Training a neural network usually has three parts: (1) Data preparation, (2) Network design and (3) Model training and evaluation.

For data preparation, I first convert the X_train data which is an .npy file contains 3220 pictures with 150*150 pixels into the data type of “float32”/255 for the CNN to compute. And for the y_train data, I conduct one hot label for four different groups of sandstone. Next part is about setting up the neural network.

For Network design, there are different types of layers:

The Conv2D layer helps process the input images. It extracts the spatial features of those images.

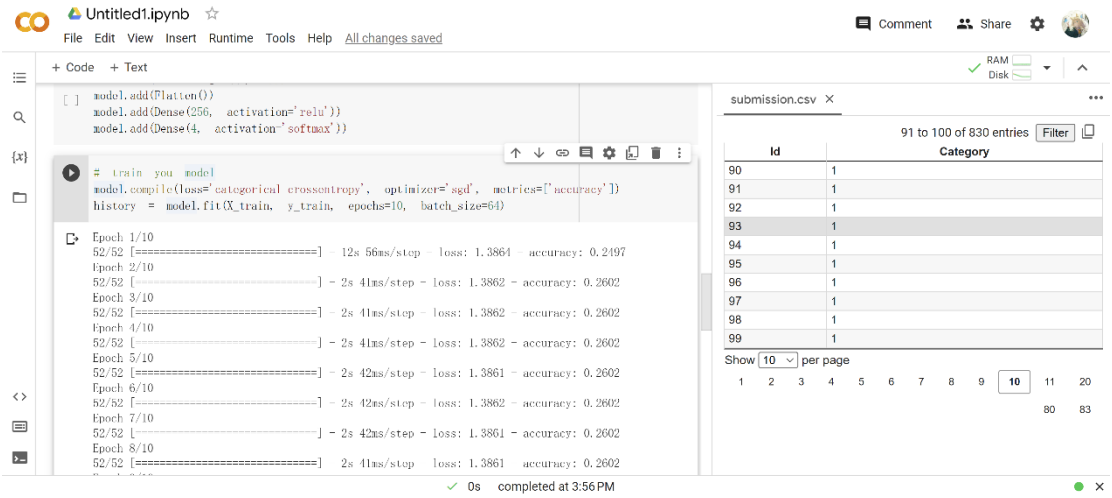
A MaxPooling2D layer with a pool size of 2x2 is after a Conv2D layer. They help to reduce the spatial dimensions of the feature maps, making the model computationally more efficient.

A Dropout layer with a rate of 0.25/0.4 is applied after each MaxPooling2D layer to prevent overfitting.

Batch Normalization layers are after the MaxPooling2D layers. They normalize the activations, which improve the robustness of the model and speed of the computation.

The Flatten layer is used to convert the high-dimensional maps into a 1D vector.

After the steps above, I ran the model and the result looks very bad: After the first few epochs, the training accuracy is convergent to 0.2602 and the result for the predication of the test data is all the same for one training and prediction (e.g. [2,2.....2,2] or [0,0,.....,0,0]). At this time, I submitted the result and the accuracy is 0.24698. I suspect that the computation is convergent to the local minimum since the accuracy is always the same. I tried to modify the neural network including the adding and deleting the layers, adding some regularization, adjusting the learning rate. However, the result remains the same. So, I think that it is not something about the network but the input data. Then I looked at my X_train data once again. I found that there is an "/255" of that and the data are all black and white, there is no need to do that. As a result, the data is too small for this datatype and for the CNN to compute. At this time, I submitted the result and the accuracy is 0.51927.

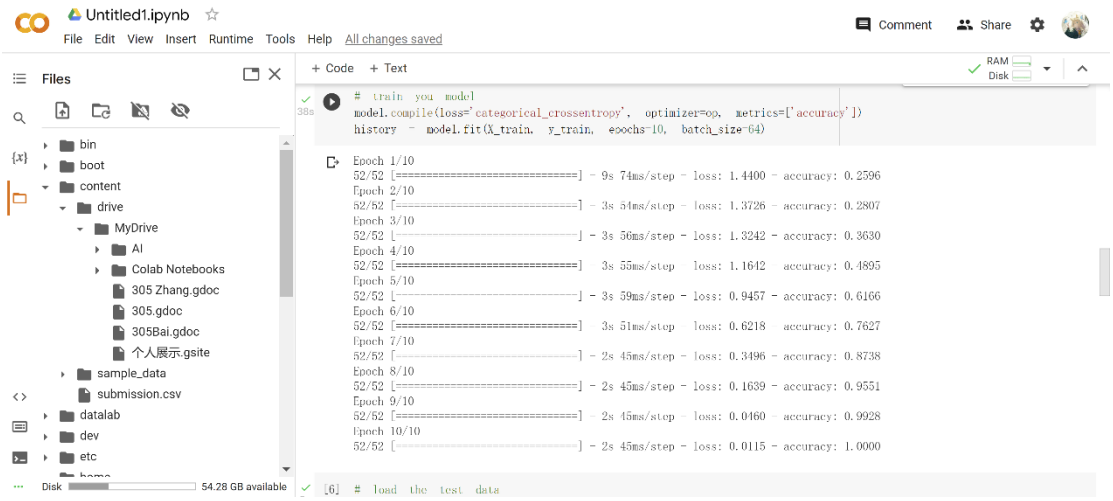



Picture 1. Error caused by /255

The second submitted result is somehow a more reasonable result compared

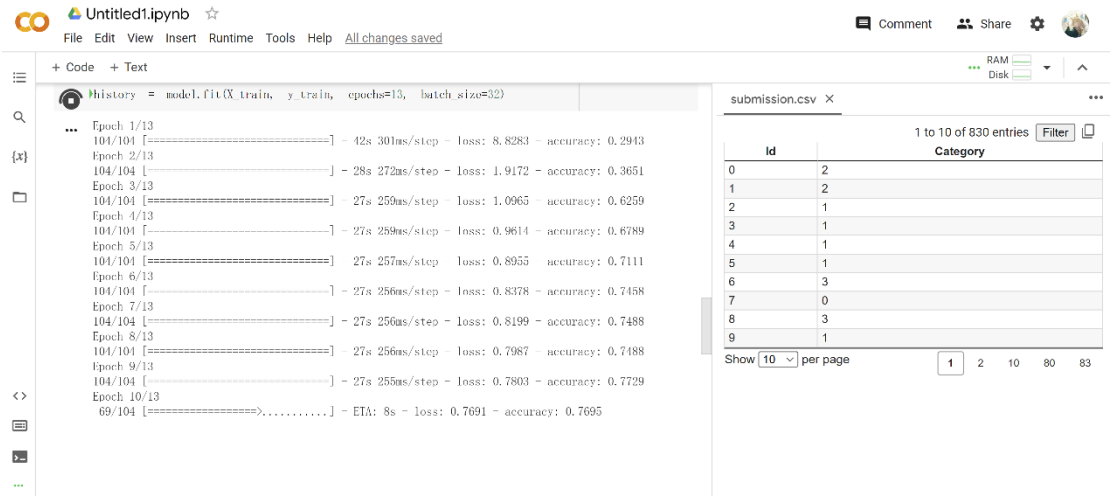
to the first one (0.24698). The rest work is about making some modifications to the convolutional neural network. I tried to make the number of neurons in each layer like a “inverted triangle” (the first layer has the most the neurons and the second layer has less, and so on) since the original one could cause overfitting problem. And then I add one more conv2D layer and some Dropout layers into CNN. After all these modifications, I make the prediction once again. This time the accuracy is 0.87108. The layers are:

(1) the first layer is conv2D with 512 neurons, (2) the second layer is MaxPooling2D with the parameter = (2,2) ,(3) the third layer is Drop with the parameter=0.25, (4) the fourth layer is conv2D with 256 neurons, (5) the fifth layer is MaxPooling2D with the parameter = (2,2) ,(6) the sixth layer is Drop with the parameter=0.25, (7) the seventh layer is conv2D with 128 neurons, (8) the eighth layer is MaxPooling2D with the parameter = (2,2) , (9) the ninth layer is conv2D with 64 neurons, (10) the tenth layer is MaxPooling2D with the parameter = (2,2) ,(11) the eleventh layer is Drop with the parameter=0.4, (12) the twelfth layer is flatten layer, (13) the thirteenth is Dense layer with 256 neurons, (14) the fourteenth is Dropout layer with parameter = 0.3. After that is batch normalization and output layer.




Picture 2. Training Process

Then I tried to add some regulation to the model since I suspected that the error is caused by overfitting. So I added L1 regularization to all the Conv2D layers with the parameter = 0.01. The result turned out that it was not good: the computation time is increased but it is not the trend of accuracy. I thought it was because of the wrong regularization parameter but I tried 0.001 and the result is not good as well. So I just delete all the regularization.




Picture 3. Model with Regularization

After all steps down, I finish my model of sandstone classification.