Subject: Computer Vision

Year: 2023

Student Name: Hengyi Ma

Student ID: a1875198

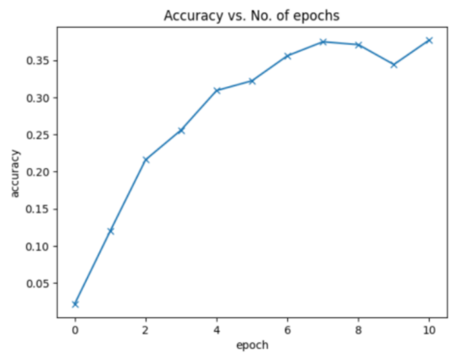
Competition Name: Animal Classification Competition

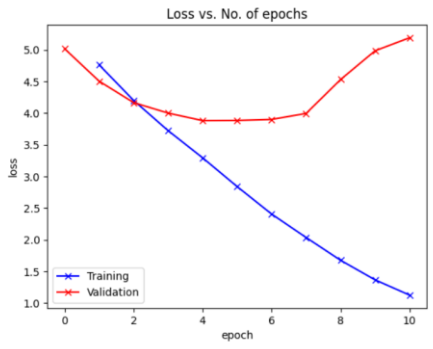
Final Results:

ACC: 0.975 FLOPs:0.41G

This task" Animal Classification Competition" requires us to classify the data we have into 151 different kinds of animal classes, with nearly 6500 pics in dataset. It is a small dataset however, which means we need to take following things into consideration: How can we get a good model with such small dataset and how can we avoid overfitting, which is common for model trained by small dataset.

But forget about dataset, firstly we need to clarify this is a classification problem. And let's take a look at the baseline model. The data argumentation includes resizing pics to fit the model, randomHorizontalfliping, centercroping, normalization and transferring them to tensor, which is common in pytorch CV tasks. Then we split the data and load the data, here the baseline use Top-3 method, which is also common in evacuating accuracy. And the loss function is cross-entropy, which is widely used in multi-classification problem and is a very good loss function actually. The **baseline model** is a CNN model, including 4 conv layers, 4 pooling layers and a full connectivity layer, through 4 conv layers we extract the features and enhance the channels. Then the model sets epochs to 10 and lr=0.001. After training, we can see the consequence as follows:

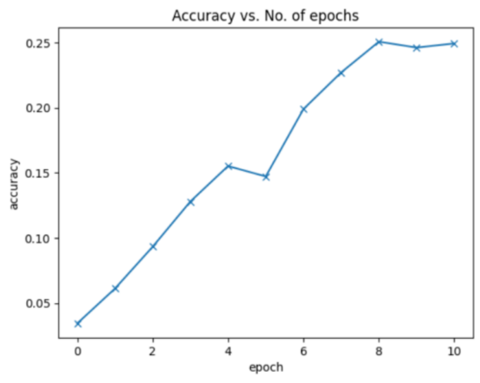


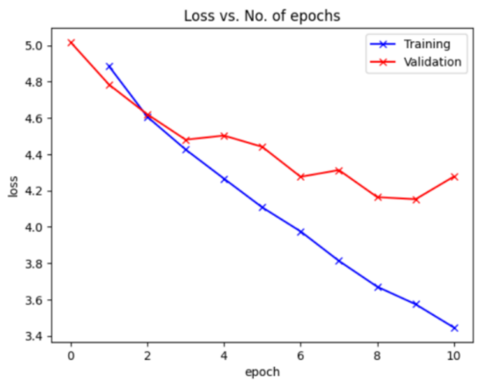


I also set more epochs to see the proformance and it seems epochs 10 is the best one, which is nearly 0.377 accuracy on validation set, but we can see from fig2 that the model have a severe overfitting problem.

The result of this baseline model is 0.374 accuracy and 0.69G flops—since the CNN model here is simple.

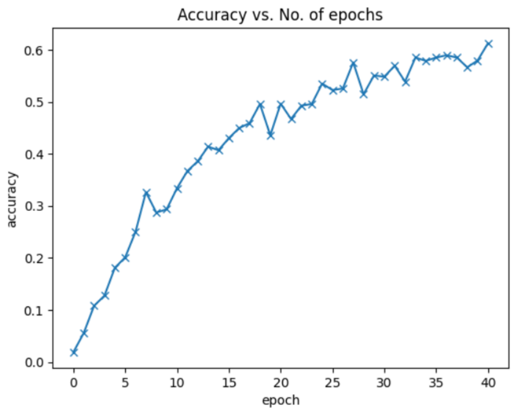
So at first, I tried to **modify this model** and solve the overfitting problem, I change the batch size to 32 to simplify the calculation, but also use TOP-3 accuracy to make it easy to compare with baseline. Add some data augmentation method to mitigate data homogeneity and add 2 batch-normalization layer into baseline model, after trying several parameter to get the best performance:

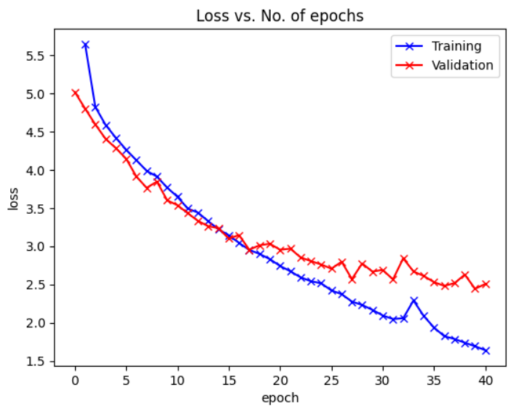




We solve the overfitting to some degree, but output is even worse. This may because we add noises in BN layers, and we don’t have a large datdaset.

Then I consider to change to another structure, like the famous **inception-4** model which can learn different range of features, but inception-4 is a very large model, which may not fit our data, so we make a simpler one: only use Stem module,A module and add a dropout layer to avoid overfitting,to simplify the model to fit our data,I also delete some conv layers in it, and set epochs to 40 and lr=0.001, here is the result:



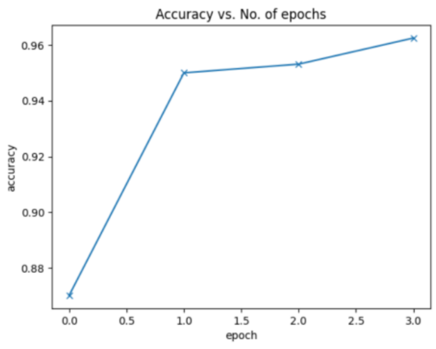


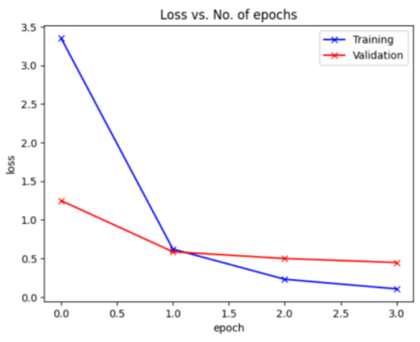
The test accuracy is nearly 0.67, 0.3 higher than baseline model and overfitting is ok, the number of Flops is 0.97,complicated than baseline, it’s a quite good one.

But here comes a problem: Actually it’s still a big model with a lot of layers, and we can see if we build a model by ourselves, it seems the model has to learn from a very low ability.

So I want to find if we can use a pre-trained model to make the model have a good initial output, and I find the **MobileNetV3-Small**, a lightweight convolutional neural network architecture proposed by Google in 2019 that aims to achieve efficient image recognition and classification on devices with limited computational resources. There are some advantages for us to use pre-trained MobileNetV3-Small, like simpler models, more flexible parameter for auto-learning and more stable activation function h-swish. As the input size of MobileNetV3-Small is different, we make some changes: resizing to 224 and setting batch size to 32, we still use TOP-3 accuracy to compare with baseline.

Then we want set the parameters like learning rate and epochs, through trying different parameter sets, we have: When lr=0.001, the output oscillates and best result is 0.9, when lr=0.0005 the output overfits and when lr=0.0001 we reach the best output at epoch 4 with 0.96 accuracy, without overfitting problem.





The accuracy of the model on testing set is 0.975 and the number of Flops is just 0.41G, which means we have an outstanding output, and we have lower calculation than baseline. Here we can know that using a pre-trained lightweight model is better than build one by ourselves since the dataset we have is limited, the MobileNetV3-Small behave good comparing with baseline model. However, such models may not behave perfect on big dataset and complicated task for its structure and may cause overfitting problem.