Aviv Shisman 206558157 Itay Hassid 209127596

exercise 2

info on code and model

pre training

- we load all of the data sets , such that we take 4000 examples from train file for training, 1000 from validation file for validation and all of the test examples for test.

  (based on what we were told in class and what classmates told us, in case we are wrong we would get better results using all the example of train file…)

- we create the filters and weights or loading an existing pretrained model we created for reducing train time or training from where we left off
- note that we require the files “train.csv” ,“validation.csv” ,“test.csv” to be near our code…

training and validation

- we are looping over the train set “epoch” times , each time we shuffle it
- each epoch is divided so batches of constant size (e.g 100) each batch is sent to “one_batch_train” function which tunes the weights based on the gradients of that batch
- each couple of batches we check validation accuracy and save weights using numpy.pickle
- at the end we write results using test function , all of this process is in “main” function

training each batch

- we calculate the gradients of each example and sum them up
- after calculating gradients we perform a combination of “adam” , “RmsProp” and “momentum” to each weight using the gradients that were calculated from all the examples. (“update_params” function)
- all of this process in in “one_batch_train” function
- the calculation of each gradient is in “forward_backward_for_single”

continue to next page

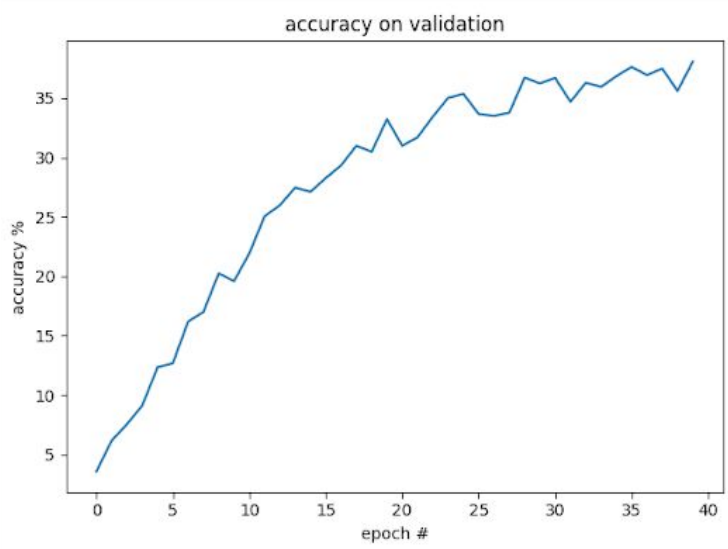
train single example

- to calculate the gradients of each example we perform feedforward and then back prop for each one
- the feed forward include 2 conv layers with “Relu” between them , maxpool layer, fully connected layer and a softmax layer at the end
- after we have the prediction we calculate the gradient in back-prop
- for more info look at the commented code of this function “forward_backward_for_single”

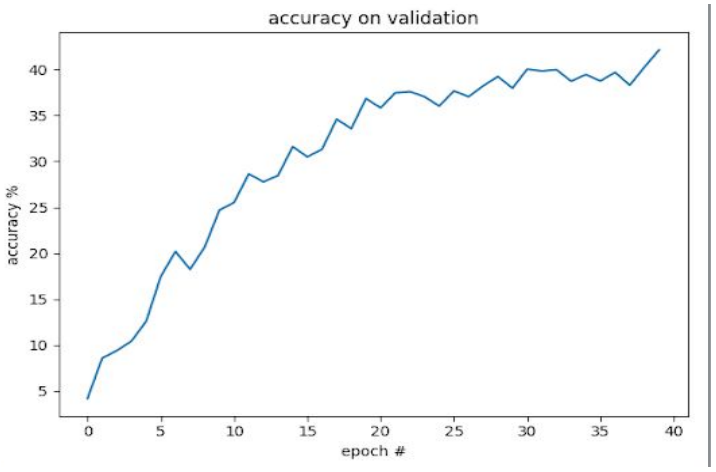
this is the full process of training and for more information please look at the code we provided :)

Graphs and results

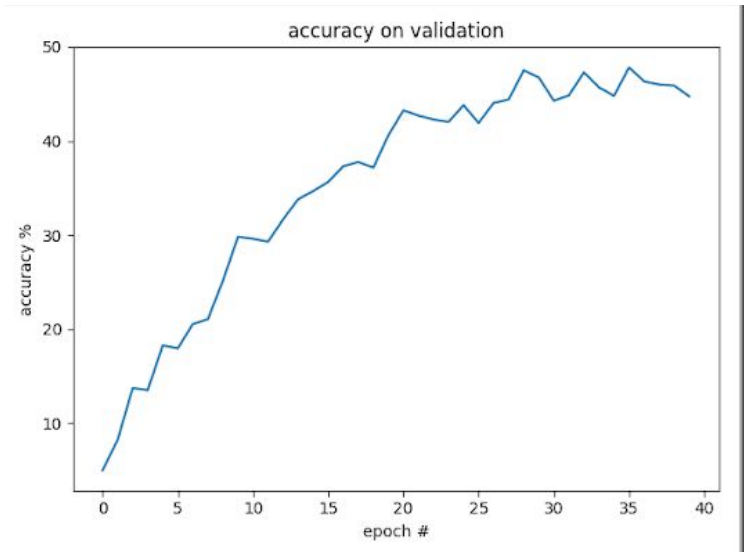
try 1



try 2 - higher hidden layer and solve bags



<u>try 3</u> - add more filters and increase their size



<u>test 4</u> - optimizing lr,filters,and etc