Qidi Han (6814891757)
EE-569
Homework #6 part 2
April-22-2020

Problem 1: Competition

I. Abstract and Motivation

As the first part of the project, we build the PixelHop++. This is a very clear way to do the image process, and we can see the detail of each process. But in our homework, this is not the best result when we use the data that proved by the TA. Therefore, modify the data may give us less model size and best answer. There are many ways to modify the PixelHop++. I will describe the process in the approach section. I will post my result and parameter setting in the experimental result section, then I will discuss the result in the explanation part.

II. Approach and Procedures

After I read more paper about improving of PixelHp++, there are two ways to improve the PixelHp++. The first way is adding more layers. The second way is keeping the test accuracy but reduce the model size. I tried both way of my experiment. I will describe my process below. But doing every step, I will replace KMeans_Cross_Entropy function with Compute function. This Compute will only use 1/3 time of KMeans_Cross_Entropy function. But the accuracy will also reduce, it will reduce 2% at initial setting. Every task, I also use zero-centered and normalization. I already fully explain the detail in my hw5 part 2.

The first way is reducing the model size and running time for the model. There are several things that I tried. The first thing is changing the energy threshold, I reduce the energy threshold, the original energy threshold is 0.001. I change the threshold to 0.0001. As this energy threshold for intermediate nodes decrease, there are more features will extract from the PixelHop++. As I try many different numbers, when the energy threshold for intermediate nodes decrease, the more dimension will create from the PixelHop++. The accuracy didn’t change too much compare with the hw6 part 1. Therefore, I try to reduce the dimension in the feature selection and LAG unit instead of improving the accuracy. As the original data, we use top 50% as our part 1. For reducing model size as the first job, I reduce the feature selection number to 3000, 2000 and 654 in my case. The reason of reduce the feature selection is because when I reduce the dimension for N, the more important information will be deleted, but I have to keep a balance between the N dimension and accuracy. Therefore, I will build model with less feature, also have high accuracy model. After reducing the N dimension, I reduce M dimension as well. I reduce M to 4 instead of 5. I tried 2,3 and 4. The test accuracy is 2% different by using the 3 and 4. Therefore, I choose to use 4 as the M dimension. The M dimension is the dimension for the LAG unit. The more dimension M may have better result. The M represent how many clusters in our cluster KMean classifier. The more M dimension will create more class from the LAG unit. Since, we want to reduce the model size, also I tried many different numbers, 4 is give me a good result. For the random forest layer, it didn’t change too much when I change the number of estimators. I set oob_score to be True, it increases 0.2%. I also slightly increase some number of estimators. The last thing I try is changing the spatial neighborhood size. I still want to use the Max-pooling as (2,2) to (1,1). Therefore, I reduce the spatial neighborhood size from 5 to 3.

For my perspective, I think more PixelHop unit will give us better result, but the model size will become huge. Therefore, the second way is adding one more hop to our original design. The result is not perfect as the first design, but there are so many limitations for running this model, the model size is much larger than the original model. Because the model is too big, I cannot run the model in my local computer, I try to limit the feature by setting the number of AC kernel as small as possible. Therefore, the model has very small number parameter in the module 1. But, I still want to have more features, therefore, I choose increase the parameter size for the module 2. But the whole model size of my design will become very large. There are four layers in my design, the spatial neighborhood size is 7. When I choose 3 or 5 to my spatial neighborhood size, it's so hard to do the choose a good max pooling. Therefore, I choose 7 as the spatial neighborhood size. Tt will reduce 6*6 dimensions every time I run the PixelHop. The max pooling has kernel 3*3 with stride =1. Therefore, it will reduce 4*4 dimensions when I run each max pooling. The input layer is 32*32. The second layer will be 28*28. After the first max pooling, it will have 24*24. Then the next PixelHop will have us 18*18. The next max pooling gives me 16*16. This 16*16 will goes to PixelHop 3, the output is 10*10. The max pooling will give 8*8. The last PixelHop give me 2*2 as the last output. I will show my setting in the Figure 2. I will discuss the result in the discussion part. Figure 1 shows the model of my design. Figure 2 shows all the parameter setting with all my test, and the comparison with the original model. I only use 15k parameters in my first module, but it still takes me around 54 mins, the reason of slow process is because there are so many dimensions they will go through, and there are 4 PixelHop. For the second module, it takes around 43 minutes to do it, because the dimension of module 1 is too big. The module 2 also will take larger times. It only take 2 mins to do the moduel 3. But the test accuracy is not good, as I mentioned before, there are too many limitations to my design. If I have more time, I will choose a better design. At least, I will not use (3,1) max pooling. It gives me too much dimensions, I have to delete more dimension to run each PixelHop.

For the test part, I also going to do full and weak supervision. I test 4 different supervision. ¼,1/8, 1/16, 1/32 of the original train images. Figure 3 shows the test accuracy curve with different number train image.

For the running time, Figure 4 show the time with the best test accuracy. The running time will be separate with module 1, module 2 and module 3. The inference time also shows on the figure.

III. Experimental Results

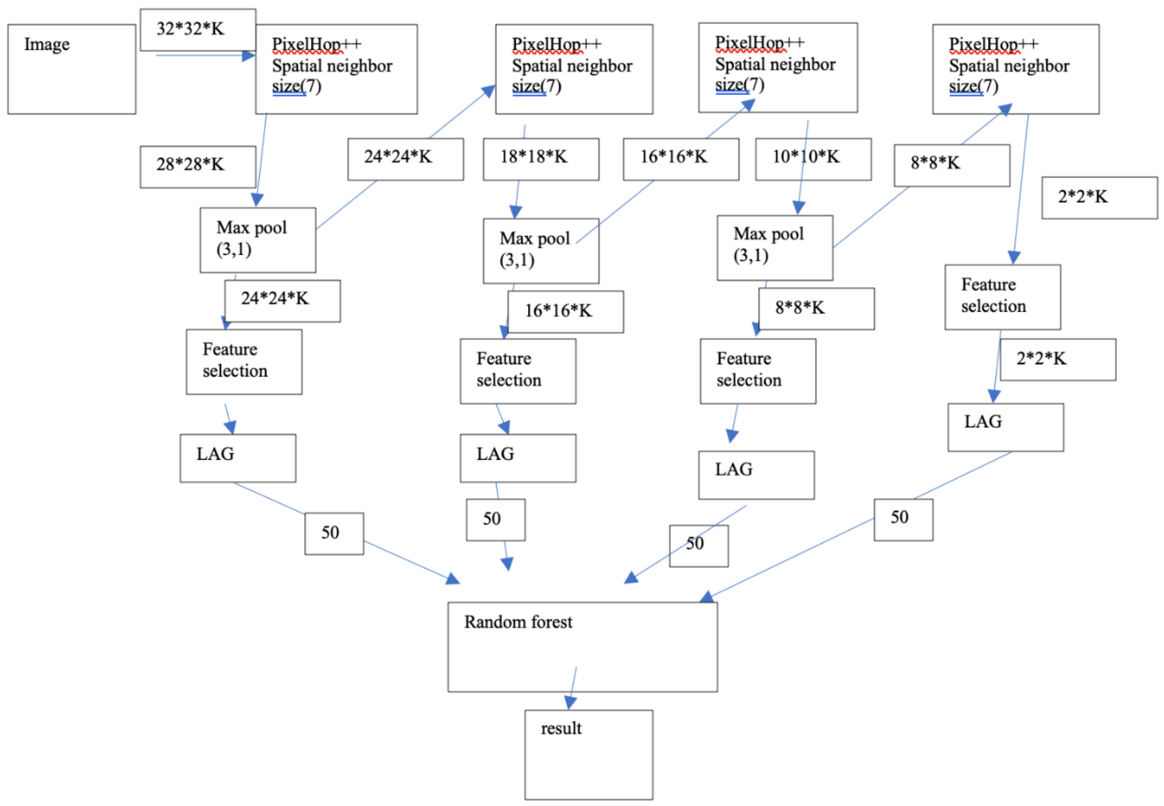


Figure 1 shows the model of my design.

Model number	Hw6 part 1 model	Design above	Model A-data change	Model B-highest accuracy
Spatial Neighborhood size	5	7	5	5
Stride	1	1	1	1
Data augmentation	No	No	Yes RandomCrop(); RandomHorizontalFlip()	No
Max-pooling	(2,2) to (1,1)	(3,3) to (2,2)	(2,2) to (1,1)	(2,2) to (1,1)
TH1	0.001	0.001	0.0001	0.0001
TH2	0.0001	0.0001	0.0001	0.0001
NS	Top 50%	[3000 3000 3000 1000]	[3000 3000 575]	[3000 2000 654]
M in LAG	5	5	5	4
RF	N_estimators=70	N_estimators=70	N_estimators=70	N_estimators=70
Model 1 size	5*5*3*42+ 5*5*274+ 5*5*523 =23075	5*5*3*45+ 5*5*101+ 5*5*144+ 5*5*257 =15925	5*5*3*42+ 5*5*291+ 5*5*575 =24800	5*5*3*42+ 5*5*286+ 5*5*654 =26650
Model 2 size	50*(4116+3425+523+3) =403350	50*(3000+3000+3000+ 1000) =500000	50*(3000+3000+575+3) =328900	50*(3000+2000+654+3) =282850
Model 3 size	3*50*10=1500	4*50*10=2000	3*50*10=1500	3*40*10=1200
Total Model size	427925	517925	355200	310700
Train accuracy	100%	100%	99.99%	100%
Test accuracy	63.95%	62.61%	58.78%	66.41 %

Model number	Model C-Spatial change	Model D-less Model size	Model E-less Model size
Spatial Neighborhood size	3	5	5
Stride	1	1	1
Data augmentation	No	No	No
Max-pooling	(2,2) to (1,1)	(2,2) to (1,1)	(2,2) to (1,1)
TH1	0.0001	0.0001	0.0001
TH2	0.0001	0.0001	0.0001
NS	[2000 2000 1000]	[1300 1500 575]	[100 100 575]
M in LAG	4	4	4
RF	N_estimators=70	N_estimators=70	N_estimators=70
Model 1 size	5*5*3*19+ 5*5*103+ 5*5*330 =12250	5*5*3*42+ 5*5*291+ 5*5*575 =24800	5*5*3*42+ 5*5*291+ 5*5*575 =24800
Model 2 size	50*(2000+2000+558+3) =229400	50*(1300+1500+575+3) =168900	50*(100+100+575+3) =93900
Model 3 size	3*40*10=1200	3*40*10=1200	3*40*10=1200
Total Model size	242850	194900	64900
Train accuracy	99.97%	99.99%	99.99%
Test accuracy	65.21 %	65.69 %	61.87 %

Figure 2 shows all the parameter setting with all my test, and the comparison with the original model.

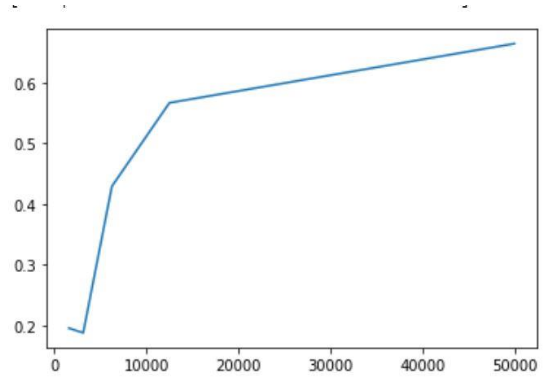
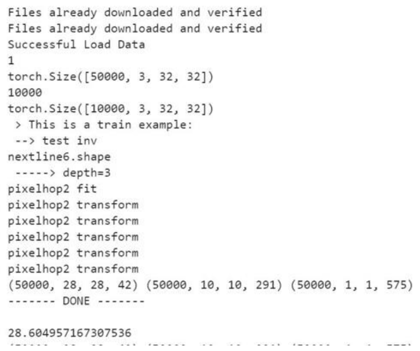


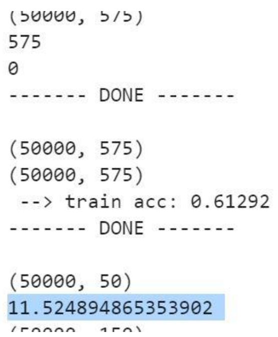
Figure 3 shows the test accuracy curve with different number train image.

Module 1 (in mins)



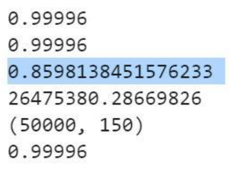
```
Files already downloaded and verified
Files already downloaded and verified
Successful Load Data
1
torch.Size([50000, 3, 32, 32])
10000
torch.Size([10000, 3, 32, 32])
 > This is a train example:
 --> test inv
nextline6.shape
 -----> depth=3
pixelhop2 fit
pixelhop2 transform
pixelhop2 transform
pixelhop2 transform
pixelhop2 transform
pixelhop2 transform
(50000, 28, 28, 42) (50000, 10, 10, 291) (50000, 1, 1, 575)
------- DONE -------

28.604957167307536
```

Module 2 (in mins)



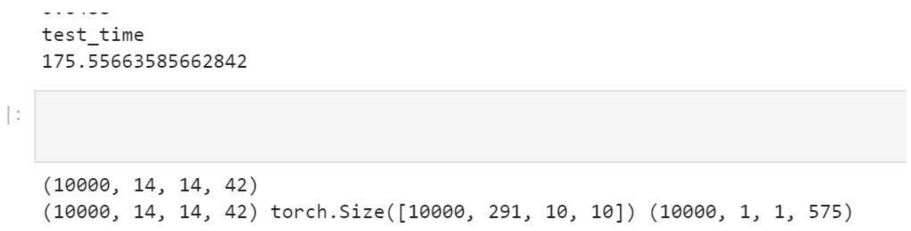
```
(50000, 575)
575
0
------- DONE -------

(50000, 575)
(50000, 575)
 --> train acc: 0.61292
------- DONE -------

(50000, 50)
11.524894865353902
```

Module3 (in mins)



```
0.99996
0.99996
0.8598138451576233
26475380.28669826
(50000, 150)
0.99996
```

Inference time (in second)



```
test_time
175.55663585662842
```

```
(10000, 14, 14, 42)
(10000, 14, 14, 42) torch.Size([10000, 291, 10, 10]) (10000, 1, 1, 575)
```

Figure 4 show the time with the best test accuracy

IV. Discussion

1: Motivation and logics behind your design:
I already fully explain the motivation and logics in the approach section. I read the paper that proved in the homework 6. I also already explain the reason that I choose those parameters. Figure 1 shows the different between each model compare with problem 2a. There is some main idea that I want to mention again. If the number N from the feature selection is reducing, the result will slightly become better, from my perspective, I think it's because when we use top 50%, it contains too much resource and the feature in the N dimension, it didn't help us to extract the most important information. And decreasing the threshold for intermediate nodes, also give us a better result.
2. Figure 3 shows the curve for this part 2. Compare with my homework 5 design, SSL have worth result. For homework 5, when I use ¼ of train images, it still has 81%. But for homework 6, we even didn't touch 80% when I use full supervision. Therefore, the result is not that accuracy. From the overall result, CNN have better result compare with 1/16 and 1/32 of train images. when the train image above 13k image, the SSL also give me around 60% accuracy whatever I change the parameters. and CNN will give me a range that always have 80% accuracy. The influence of the training image numbers is when I train the image by using the model 1. The ¼ and 1/8 of training image still give me a better result, but for

another’s, it gives me a worth result. I’m think because we train the model by using 10k image, the model has good result. But when we use too less image, the model is not robust.
3.
Figure 4 shows the time for training time and inference time. I also already mentioned the time for my own design. The original model will use around 16-28mins for module 1, it depends on the memory of my computer when I run multiple task at one time. It takes 11-13 mins for module 2, because I replace KMeans_Cross_Entropy function with Compute function, it gives me much after process time. For the model 3, it only takes 1-2 mins.
For the inference time, it takes around 175s, I forget change unit when I run the program, it will be around 3-4 mins.
4.
The Figure 2 shows all the model size for each case. My best result in homework is 330k model size with 86.5% accuracy. It’s better than SSL, but when I change the model size to very small in SSL, the accuracy still very high, Model E can shows this result. From my perspective, I think it’s because there are three modules and three different hops, each hop didn’t affect others. Then I reduce the dimension in first hop, the hop in second will not affect too much by reducing the dimension in the first hop. I cannot talk which one is better. We should choose one that fit to our problem.