DSP2019—Project Report

R08921053 電機研一

※The result form of my experiments

	Spectrogram	Optimizer	Loss function	epoch	Accuracy (val-set)	Accuracy (test-set)
1	Window=tukey	SGD	CrossEntropy	400	0.15	x
2	Window=Hamming Mode= phase	SGD	CrossEntropy	120	0.443	0.481
3	Window=Hamming Mode= magnitude	SGD	CrossEntropy	50	0.871	0.874
4	Window=Hamming Mode=magnitude	Adam	CrossEntropy	30	0.863	0.875
5	Window=Hamming Mode=magnitude (some adjustment)	SGD	CrossEntropy	50	0.825	0.845
6	Window=Hamming Mode=magnitude (some adjustment)	Adam	CrossEntropy	30	0.866	0.860

※Settings for generating spectrograms

Firstly, all of my methods to generate spectrograms are by the function, "scipy.signal.spectrogram()". And I have to thank TA to provide me this useful tool. So, I am going to introduce each kind of setting about spectrogram() below:

(1) fs=1000Hz, window=('tukey', 0.25)

Actually, this is the original setting that providing by the website[1]. However, no matter I try adjust fs(1000Hz/3000Hz), or even the loss function and optimizer in the model, the loss value only can reduce from 3.0 to 2.4 after 300~400 epochs. And the accuracy in "val" set is about 0.15. That is, the loss function cannot converge to 0 in this machine learning model.

In this case, I think the main reason to fail is: the characteristic on each picture is too weak to identify. On the view of system, each picture is almost the same! As a result, it is really difficult to distinguish each category of the picture, even to train the machine learning model.

(2) fs=1.0, window=('hamming'),nfft=256, mode='phase'

Now, I try to follow the advice given on the “DSP HOMEWORK.pdf” and try to use the phase spectrum and hamming window with length of fft is 256.

In this case, the pictures after spectrogram are much more clear than case(1). Even with my eyes, I can observe the white streak on almost every picture. Of course, the characteristic of picture is better than last cases. I predict the result would be better.

However, the result of this case is very bad, only 0.44 accuracy! I guess the reason is all of the pictures only have a lot of noise but not the real characteristic. Hence, even I can see something on the picture, but the difference between each picture is not apparent, and I need to use a lot of epochs to train the model. Somehow, the result still not good

(3) fs=1Hz, window=('hamming'), mode='magnitude'

Again, thank to the TA. After this failing experience, TA provided me a better setting, with magnitude spectrum and hamming window. In this case, the picture in each category is very clear. I can even directly distinguish them by my eyes. And so, the result of this spectrogram is very successful. Just after 50 epochs, the loss value reduces from 3.0 to 0.15. And the accuracy in “val” set is about 0.87, which is really high!

After I submit the result in *.csv type to the online judge system, the accuracy is about 0.874, which is very close to the result on “val” set. This also reveals that I my model didn’t over-fit on the “val” set.

Since this case has the best result among all case, so I decide to try another model on this case, that is Adam optimizer and CrossEmtropyLoss loss function. Actually, we really get a better result compare to the formal optimizer.

(4) Fs=1Hz, window=(‘hamming’), nperseg=256, noverlap=128, nfft=256, mode=’magnitude’

This time, I try to improve the performance of magnitude spectrum with hamming window. So, I try to adjust some parameter on the spectrogram and use SGD, Adam optimizer model to test whether the result would be better or not. However, the result is almost the same with case (3).

※Settings for your neural network

At this part, we have already generated the spectrograms, and we have a series of pictures in “val”, “train”, “test” set. We are going to use the typical machine learning model to provide the answer on test.npy.

(1) optim.SGD(net.parameters(), lr=0.05); nn.CrossEntropyLoss()

In this case, I use the basic setting on the tutorial providing by TA. So, with the basic two layer convolution neural networks with “relu” layers. Furthermore, I use the SGD optimizer and entropy loss function. Entropy loss function is to use log function on input, and then take softmax active function to afford the probability. More detail can see [2].

(2) optim.Adam(net.parameters(), lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0, amsgrad=False); nn.CrossEntropyLoss()

In this case, I know that nn.CrossEntropyLoss() is a suitable loss function on this category of problem, so I would still prefer to use this loss function. However, from the article[3], I get the point that Adam is the update version of RMSprop, this means Adam is essentially the same as RMSprop. So, finally I decide to use Adam as the optimizer as my setting.

According to my experiment, the accuracy between Adam and SGD is almost the same. But Adam is really fast to let the loss function converge. So the advantage of Adam is speed. I would prefer to use Adam in the future because the accuracy are very similar finally.

※What you have learned for this project

(1) Difficulties I encounter

It really frustrated me that the result of spectrogram—the .jpg file are looking the same totally. Despite I assume it is not a problem and use them to train the typical 2-layers model, it just doesn’t work and the loss function cannot converge. I stop on the stage and try to change the frequency of spectrogram, optimizer, and even loss function. But none of the behavior is meaningful and nothing change after 2 days. Something worse is I don’t nothing where is the problem or bug!

In this case, I have no choice but to consider the last option— to change the type of spectrogram. However, I don’t know too much on this function and there are really too many parameters to be adjusted. Again, thank to the instruction from TA, I try to use the spectrum with hamming window function. And after some effort on training the model and deal with the output type, finally I successfully output the .csv file and upload to the online judge system.

(2) interesting things you find

Something interesting is that: The appearance of picture is the most important point at all—The most apparent picture setting may not has best result. If you have some trouble on spectrogram to generate a lot of noise on the picture, the result would be very bad! Even there are a lot of white spot on the picture, but the difference is not very huge. So, I think that is the reason why case (2) only has accuracy 0.44 and case (4) has accuracy 0.87. But, if you cannot see anything on the picture, just like case 1, then the model may be not able to distinguish difference in high probability, which is the worst case!

(3) special techniques you apply

I try to adjust the parameter in magnitude spectrum with hamming window. Like the length of FFT or number of overlap. But it looks almost the same as other cases. So, this implies we need some huge changes on the spectrogram to improve model.

※Reference

[1] setting of "scipy.signal.spectrogram"
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.spectrogram.html
[2] PyTorch 学习笔记（六）：PyTorch 的十八个损失函数
https://zhuanlan.zhihu.com/p/61379965
[3] Optimizer 优化器
https://morvanzhou.github.io/tutorials/machine-learning/torch/3-06-optimizer/