ELEC5306： Video Intelligence and Compression

Project #2: Challenge in Video Restoration

Due: **31 May 2020** 11:59PM

1 Objectives

This laboratory aims to give you a chance to practice your knowledge in neural networks. In this laboratory you will:

- Use PyTorch to load images and train neural networks for video compression artifact removal (JPEG 2000, HEVC(x265)) in CPU/GPU.
- Investigate different settings (eg. batch size, learning rate, loss type, initialization method) and try different tricks to get good performances.

2 Dataset

The dataset is cropped from part videos of vimeo-90K dataset [5]. The training set consists of 599 videos (7 frames per video), and the validation set consists of 139 videos. The test set is not released for evaluating final submissions.

Note you can only use the dataset we provide to you for your assignment however you can **do your own splitting besides the provided training/val split to better tune your network.** If you split data in your own way rather than ours, you need to specify in your report and presentation. Note that the validation videos cannot be utilized for training, otherwise the result is invalid.

All related data can be downloaded from:
https://www.dropbox.com/s/i1o74xou8bm74al/data.zip?dl=0

3 Experiments

Take the video compression artifact removal (JPEG 2000) as example, your task is to build neural networks to restore compressed videos. Firstly, you can start with training single frame networks, which can also be called image restoration network [2, 6, 7]. Simple CPU/GPU version of ARCNN/FastARCNN/DnCNN can be downloaded from

https://github.com/chriszhenghaochen/ELEC5306

Generally, **DnCNN with depth 15 is chosen as the baseline**. After training single frame networks, you can utilize multiple frames to further improve the performance if your CPU memory is large enough [3, 1, 5, 4]. The baseline model, you can download from:

https://drive.google.com/open?id=188TXtdhj8tnJyIBr8xsYW-7b2H-SWpfj

You are required to do following things:

1. Try to finetune the parameters to improve the performance from baseline model
2. Try to deeply investigate the model (**model.py**) to come up with novel improvement (e.g. new design based on current network, new network besides ARCNN/FastARCNN/DnCNN)
3. The testing time for your model should be less than 2 mins.

In the following, there are some hints about using Pytorch based on the demo code we provided. Some basic operations like constructing layers are all mentioned in last tutorial. Other questions about Pytorch can be answered by searching Pytorch function:

https://pytorch.org/docs/stable/index.html.

To use Colab, we do provide the colab running code, you can download from:

https://drive.google.com/open?id=1ETnahHe9EoZzCWXFvfqa6EUz8O3ty2di

3.1 Hints about using Pytorch

Create Dataset class and load data. You can create your own Dataset class (eg. ArtifactDataset) inheriting the Dataset class from Pytorch in the file"dataset.py". This class usually have one " getitem " function and " len " function. After loading images and constructing Dataset, we can utilize the Dataset to build Dataloader, which is used for enumerating batches while training.

```
DDataset = ArtifactDataset(xs)
DLoader = DataLoader(dataset=DDataset, numworkers=4, droplast=True, batchsize=batchsize,
shuffle=True)
```

Save and load model. Model files contain all of the trained parameters. Optimizer files contains the training status. We prefer to save both model and optimizer use the following functions:

```
# save the network to some file:
torch.save({
'epoch': epoch + 1,)
'state_dict': model.statedict(), 'optimizer': optimizer.statedict()},
os.path.join(savemodelpath, 'checkpoint%03d.pth.tar'%(epoch + 1))
# first define the network structure model =
DnCNN()
# load the parameters to the defined network:
checkpoint = torch.load(resumemodeldir) resume_epoch =
checkpoint['epoch'] model.loadstatedict(checkpoint['statedict'])
optimizer.loadstatedict(checkpoint['optimizer'])
```

Use GPU. If you would like to use the GPU to accelerate your program, the first thing you need to make sure is that your computer is capable of running CUDA (the package to enable running codes on NVIDIA graphic cards). If the following command prints '**True**', then you can use GPU to accelerate your program.

```
print(torch.cuda.isavailable())
```

When using GPU, your whole network, and the input images and labels should be moved to the GPU. This can be achieved by **.cuda()** method.

```
model.cuda()
images, labels = images.cuda(), labels.cuda()
```

If you need to move the tensors back to cpu (for example, you need to store the loss or print some weights on screen), then you can use the method **.cpu()**.

4 Submission

You are supposed to finish this project within a group of 2-3 people. Your submission should include the following files and follow the instructions:

- The python file '**dataset.py**' which contains the class 'Dataset' and other functions relating to handling data. (necessary)
- The python file '**model.py**' which contains the class 'Network' that defines your network layers (method **init**) and forward process (method **forward**). (necessary)
- The python file '**utils.py**' which contains any other related functions. (if you have)
- The python file '**main.py**' which is used to train your network. (necessary)
- The python file '**main_ test.py**' which is used to test your result. Please follow the given template to finish your test file. (necessary)
- The trained network paramater file '**project2.pth**'. This file should be too large (hundreds of MBs) to submit through *Canvas*, so you need to submit a text file named '**project2.txt**' that only contains a Google Drive link to the file '**project2.pth**' in your google drive. We are able to check the time stamp of the file on Google Drive, so do not update your file after the deadline, otherwise you will be punished as late work. (necessary)
- Your report '**project2.pdf**'. The report should be no more than 6 pages (excluding references), which should include introduction, previous work, your backbone network definition and your improvement, experiments and analysis, training details, and your conclusion and future work, etc. When you write the authors, please write your names and your SIDs. You should also indicate your contributions in the appendix. You can use the latex template in CVPR 2018 as your formatting reference: http://cvpr2018.thecvf.com/files/cvpr2018AuthorKit.zip. (necessary)

The files (including the 4 necessary python file and one optional python file, one text file, and one pdf file) should be contained in a .zip file named as '**project2 YourGroupNumber.zip**' with no spaces in the file name and submitted through *Canvas*. Please use two digits in the file name. For example, group 1 should write 'project2 01.zip' and group 20 should write 'project2 20.zip'. Your marks will be given according to your codes, your written report and your presentation. For detailed marking scheme, please refer to Appendix: Marking Scheme.

After your submission, we will run your codes and see the results of your model on the test dataset. Before the last course, we will release a ranking on *Canvas*, and the top three teams will be given bonus scores. In Week 13, you will be given 5 minutes to do a presentation and answer questions, and the top three teams will be given a bit more time to show their methods and results.

Appendix: Marking Scheme

The total marks for Project 2 is 20 in your final score, and your codes, report and the presentation account for 30%, 40% and 30% respectively. The numbers in the following chart are in percentages. Please note that the marking scheme may be updated in details.

1 codes	1.1surpass the baseline model & test in sufficient time[1]	10
	1.2 good performance [2]	10
	1.3 well commented codes	5
	1.4 proper improvement[3]	5
2 report	2.1 introduction	5
	2.2 previous work	5
	2.3 implement the network and report results	5
	2.4 make some modifications to get better performance[4]	10
	2.5 conclusion and future work	5
	2.6 good references	5
	2.7 good writing	5
3 presentation	3.1 analyse of the network you are using	7
	2.3 clear slides	8
	3.3 clear descriptions	10
	3.4 good answers to the questions (if any)	5
	3.5 (bonous)to the impression of the lecturers [5]	10
4punishment	4.1 cheating	-100
	4.2 late submission per day	-5
	4.3 bad coding	-20
	4.4 absent presentation	-20

[1] . Your model should surpass the baseline model we provided (5 marks) , and your test time must less than 2 mins to get the basic marks (5 marks)

[2] You can get the full score if you can get the performance better than 50% of the class. There is a bonus of 5% for the groups that are ranking top 3.

[3] You should try to investigate different neural network models to get improvement, if you can come up with a better network (not just ARCNN, FastARCNN and DrCNN) you can get full mark (5 marks) for proper improvement. otherwise the marks for this part will based on how you contribute to improve the performance. Note simply tuning the hyper parameters does not count as proper improvement, you will get 0 in this part

[4] You should clear illustrate what have you done (e.g. new model, new design etc..)to improve performance, this part will also be marked based on 1.4,

[5] The 10% will be added to the final score if there are any impressive presentation performance, i.e. online presentation, videos or animations.

References

[1] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. Yang. Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *arXiv preprint arXiv:1810.08768*, 2018.

[2] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[3] G. Lu, W. Ouyang, D. Xu, X. Zhang, Z. Gao, and M.-T. Sun. Deep kalman filtering network for video compression artifact reduction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 568–584, 2018.

[4] Y. Tian, Y. Zhang, Y. Fu, and C. Xu. Tdan: Temporally deformable alignment network for video super-resolution. *arXiv preprint arXiv:1812.02898*, 2018.

[5] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *arXiv preprint arXiv:1711.09078*, 2017.

[6] K. Yu, C. Dong, C. C. Loy, and X. Tang. Deep convolution networks for compression artifacts reduction. *arXiv preprint arXiv:1608.02778*, 2016.

[7] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.