2020 Summer Intern Report

Xinyu Jiang

ABSTRACT

Keywords: stain normalization, conditional generative adversarial network, tumor classification, python

1. INTRODUCTION

1.1 Background

In the diagnosis of cancer of a majority of organs and the biomarkers data-driven neural network identification fields, the Hematoxylin and eosin(H&E) stained whole-slide image (WSI) is admitted as a general acceptance standard. The WSIs from various institutes or labs from one or multi datasets often hold different stain-style patterns. However, this stain-style pattern plays an indispensable role in the neural network based classification or segmentation tasks and analysis of histopathology. The lack of the stain normalization (SN) in training or test set leads to a poor performance of neural networks with a low test accuracy and high error generation rate. In order to overcome this challenge, stained normalization algorithms have been added in the training.

1.2 Program

Our goal is to develop a decentralized normalization method by integrating patchs from multi datasets to generate a "interpolation", such as stain style distribution with GAN, which has not been researched yet. What's more, color pattern can be guided by the patch-level label, but the label-dependent color normalization methods were only mentioned in a minority of researches in previous work. The contribution of our program is listed as follows. 1. We propose a novel generative adversarial neural network for color normalization. 2. We substitute the traditional method, learning a precise distribution of single dataset, for a new method, generating a stain style probability distribution on multi datasets. The new method does not depend on the prior knowledge of the datasets, which is labor saving.

2. METHODS

2.1 Tumor Classification Neural Network

We name the set of stain-style as S, then assign a histopathological image set of stain style $s \in S$ as X^s, the label set as y and a tumor classification neural network training on images of stain style s as $f^s: X^s \to y$. And the loss function can be represented as

$$L^s_{tumor} = \mathbb{E}_{(x,y)\sim P^s_{data}(x,y)}[l(f^s(x), y)], \tag{1}$$

where $\mathbb{E}(\cdot)$ represents the expectation, $P^s_{data}(x, y)$ represents the probability distribution of $X^s \times y$, $l(f^s(x), y)$ represents an image-level loss function.

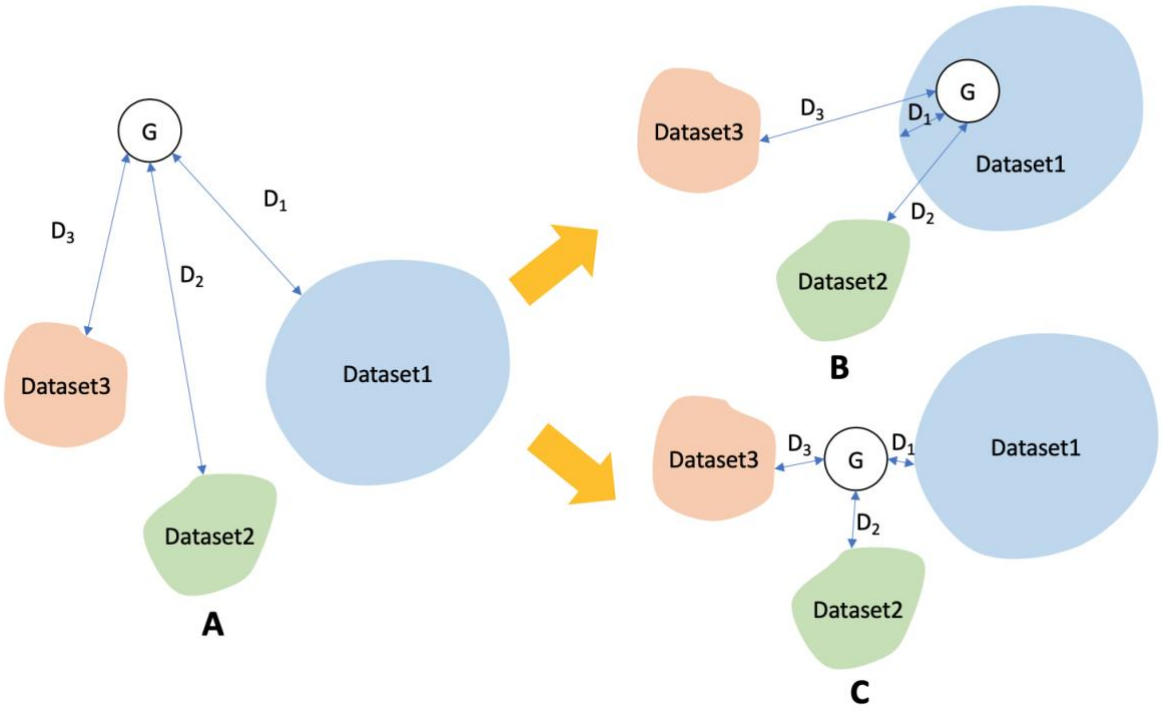

Figure 1. A) Three Datasets stands for the initial distributions of three datasets, the G represents the initial generator and the white circle is the probability learnt by the generator. The D_s denotes the discriminator trained with the s^{th} dataset and the length of blue array stands for the value of loss function $\mathcal{L}_{GAN}^{s}$ ($s = 1, 2, 3$). B) Pattern of the previous work,[1-4] where the generator learns a specific probability distribution of a single dataset (e.g. $s = 1$) by minimizing the $\mathcal{L}_{GAN}^{1}$. C) Model of our work, where there is only one generator which learns a stain style probability distribution from multi datasets by minimizing an aggregated loss function $\mathcal{L}_{GAN}$.

2.2 Color Normalization Neural Network on Multi Datasets with GAN

A generative adversarial network (GAN) contains two components, which are a generator G and a discriminator D. The generator G takes a random noise as input. And it tries to match the generated picture with a target distribution, while the discriminator tries to distinguish between the generated picture and the training set. If the output of the generator is conditioned by additional input images, this GAN is a conditional GAN (cGAN).[5] We use cGAN model to produce a target stain style distribution incorporating information from multi datasets.

The implementation of the program is based on the code from GitHub.[6] There is one generator and three discriminators in the model. The generator accepts pictures and a random noise as input, while each of the three discriminators accept the pictures which the generator produces as well as a database with a certain stain style. The generator attempts to make the discriminator believe that the generated pictures are the pictures from datasets, while the discriminators attempt to distinguish the two. They modify their inside argument in basis of the loss function. The detail of the loss function is discussed in a research essay that will be public soon, which will not be mentioned in this article.

2.3 Implement an auto-drawer of the loss function which changes with the increase of the training epoch times

In order to make the final result visible, we write a python script to generate a line chart based on the loss function, accuracy rate and training epoch times data.

3. RESULTS

3.1 Datasets and Experimental Settings

We evaluate the system on the large patient cohorts from The Cancer Genome Atlas (TCGA), which is available at [*]. This dataset includes three subsets of colorectal cancer: 1) TCGA-STAD (n=432 samples), 2) TCGA-COAD (n=460 samples), 3) TCGA-READ (n=171 samples).

All experiments are performed device Nvidia Tesla V100. We use the GPU edition with cuDNN= 7.6.5 and CUDA= 9.2_0. The coding language is python=3.6.7 and the system is Linux. And the pytorch framework, tensorflow-gpu=1.10.0, keras=2.2.2 is used to implement the proposed GAN model.

3.2 Running Process

```
progress  epoch 1  step 149  image/sec 4.7  remaining 2143m
discrim_loss1 0.51105654
discrim_loss2 0.47909743
discrim_loss3 1.443672
discrim_loss 2.4338248
gen_loss_GAN 4.768282
gen_loss_L1 0.8528702
recording summary
progress  epoch 1  step 199  image/sec 5.0  remaining 1995m
discrim_loss1 0.5030082
discrim_loss2 0.41879192
discrim_loss3 1.6108284
discrim_loss 2.532628
gen_loss_GAN 5.637332
gen_loss_L1 0.9084767
progress  epoch 1  step 249  image/sec 5.4  remaining 1858m
discrim_loss1 0.46646437
discrim_loss2 0.36477217
discrim_loss3 1.7077318
discrim_loss 2.5389678
gen_loss_GAN 6.2005506
gen_loss_L1 0.96361095
recording summary
```

Figure 2. This is a terminal page in the training process of the proposed GAN. The loss of the three discriminators and the generator were demonstrated by 50 epochs.

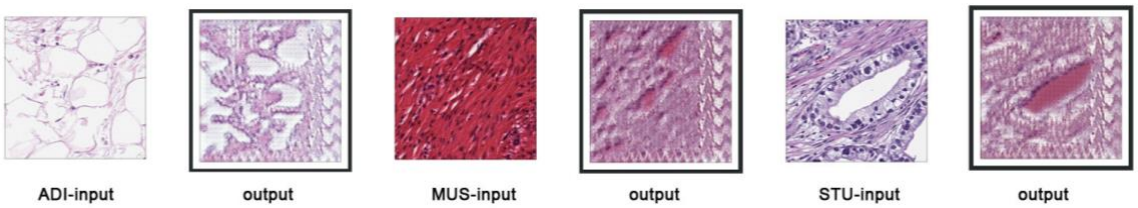

Figure 3. Three illustrative examples of the generated new pictures, and the input pictures are from three datasets with different stained style.

4. DISCUSSION

There were two main difficulties which we met in the program. One was solved, and the other one remained to be solved in our future work.

Firstly, the program code was copied from others and then altered, which means we need to learn from their code and set the development environment same as theirs so that the code can run on our computer. And the code was written in 2017, a lot of packages it used were of old edition. It took a lot of work to download them and update the code to fit in the packages. Meanwhile, we attempted to run the code on GPU rather than CPU which allowed the computation to be faster. And the code was run on a remote server, the setting of some packages was fixed. The whole development environment configuration was a tough job.

Secondly, theoretically, the generated picture should be more precise with the increase of the training epoch times and the decrease of the generator loss value. However, when the training epoch times was bigger than 10, most of the generated pictures contained a certain graph. When the training epoch times was bigger than 15, the generated pictures were all the same. We tried several ways to improve this situation. We tried to increase the l1 parameter proportion, which helped the generated picture to retain more information from the input picture. And we also decreased the learning rate and add some layers in the networks framework. But these ways had little amelioration on the final result. Due to the limited time, we would like to change the networks framework of discriminator to a Resnet framework. The promotion of complexity would help the GAN capture more information of the pictures and avoid overfit.

5. CONCLUSION

Developing a decentralized normalization method via integrating all patches from multi datasets to generate a "interpolation", such as stain style distribution with GAN, was a new research topic. And this program put forward with a new solution to this topic, a novel generative adversarial neural network for color normalization. And we substitute the traditional method, learning a precise distribution of single dataset, for a new method, generating a stain style probability distribution on multi datasets. The model we produced realized the basic function of the theory model, which could be seen in Figure 3. But the final performance appeared an out-fit problem and could still be improved. We thought about the countermeasure for that and would probably test them in the future.

6. EXPERIENCE

This internship taught me a lot. Due to the epidemic situation of COVID-19, all the works were done online, which was a challenge and also a chance. We used a remote server to run the code and held meetings regularly on zoom.us. I learnt to how to use the remote server, how to configure the development environment and how to use Linux screen orders.

As for the program content, I firstly read several papers, learnt the theory and the framework of GAN and some basic kinds of GAN, like Cycle-Consistent Generative Adversarial Networks, Conditional Generative Adversarial Networks and others. We finally choose cGAN model to implement our program. And then I copied several cGAN codes from GitHub and picked one to realize it on our computer. After organizing the datasets to training-sets and test-sets, I replaced the original input with our input files and catered the framework to our theoretical model. Then I added some details to our code, for example, saving the model every 50 epochs, saving loss function data to a new file, reading the data from files and draw line charts and so on. We checked the generated pictures and attempted to improve the performance of model, which helped me review the framework of Convolutional Neural Network and some amelioration ways of GAN performance.

Thanks for the help of my tutor, Yiqing Shen. He guided me, taught me the idea of the program and helped me with all the problems during this period patiently. It was my honor to participate in this program.

REFERENCES

[1] Cho, H., Lim, S., Choi, G., and Min, H., "Neural stain-style transfer learning using gan for histopathological images," arXiv preprint arXiv:1710.08543 (2017).

[2] Rana, A., Yauney, G., Lowe, A., and Shah, P., "Computational histological staining and destaining of prostate core biopsy rgb images with generative adversarial neural networks," in [2018 17th IEEE Interna- tional Conference on Machine Learning and Applications (ICMLA)], 828-834, IEEE (2018).

[3] Shaban, M. T., Baur, C., Navab, N., and Albarqouni, S., "Staingan: Stain style transfer for digital histolog- ical images," in [2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)], 953–956, IEEE (2019).

[4] Zanjani, F. G., Zinger, S., Bejnordi, B. E., van der Laak, J. A., and de With, P. H., "Stain normalization of histopathology images using generative adversarial networks," in [2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)], 573-577, IEEE (2018).

[5] Mirza, M. and Osindero, S., "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784 (2014).

[6] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A., "Unpaired image-to-image translation using cycle-consistent adversarial networks," in [Proceedings of the IEEE international conference on computer vision], 2223–2232 (2017).