Preparation

**Prerequisites**

* Python (3.6)
* Numpy (1.14.5)
* Scipy (1.0.0)
* Matplotlib (2.1.2)
* Sklearn (0.19.2)
* Pytorch (0.4.1)/CUDA 9.0
* torchvision (0.2.1)
* scikit-image (0.13.1)
* PIL (5.1.0)
* OpenSlide 3.4.1(Please don't use 3.4.0 as some potential issues found on this version)/openslide-python (1.1.0)
* tensorboardX (Standard along tensorboard that also works for PyTorch. This is mostly used in monitoring the training curves.)

Most of the dependencies can be installed through pip install with version number, eg.

*$ pip install numpy==1.14.5*

Or just simply using:

*$ pip install numpy*

A ‘requirements.txt’ file is also provided, so you can install most of the dependencies at once:

*$ pip install -r requirements.txt -i https//pypi.python.org/simple/*

For PyTorch please consider downloading the specific wheel binary version then use the following command:

*$ pip install torch-0.4.1-cp36-cp36m-linux\_x86\_64.whl*

**Data**

* **Whole slide images**

The dataset is the whole slide images (WSI) in \*.tif format from the Camelyon17 challenge. There has dataset of Camelyon17 and Camelyon16, and we used Camelyon16. [https://camelyon17.grand-challenge.org/Data/] You can download from either Google Drive, or Baidu Pan. Note that, one slide is usually ~100Kx100K pixels at level 0 and 1GB+ on disk. There are 400 slides in total from two independent datasets collected in Radboud University Medical Center (Nijmegen, the Netherlands), and the University Medical Center Utrecht (Utrecht, the Netherlands), together about 700GB+, so make sure you have enough disk space. The tumor slides for training are named as tumor\_xxx.tif, where xxx ranges from 001 to 110. The normal slides for training are named as normal\_xxx.tif, where xxx ranges from 001 to 160. The slides for testing are named as test\_xxx.tif where xxx ranges from 001 to 130. In our study we only use the training set. When you download all the slides, please put the tumor slides and normal slides for training/validation/testing under corresponding files under “/wsi/raw\_data/”. For example, if you use normal\_001.tif for training, put it in directory “/wsi/raw\_data/train/”. Note that tumor\_001.tif to tumor\_070.tif and normal\_001.tif to normal\_100.tif are provided by Radboud University Medical Center (Nijmegen, the Netherlands), so they belong to training or validation set. tumor\_071.tif to tumor\_110.tif and normal\_101.tif to normal\_160.tif are provided by University Medical Center Utrecht (Utrecht, the Netherlands), which should belong to test set.

* **Annotations**

The Camelyon16 organizers provides annotations of tumor regions for each tumor slide in xml format. We've converted them into simpler json format, located under “/wsi/all\_jsons”. Regardless of it is tumor or normal, when you need any WSI as train, valid or test data, please put its’ json file in corresponding file in “/wsi/jsons”, eg. If you use normal\_001.tif for training, put normal\_001.json into “/wsi/jsons/train”. Each annotation is a list of polygons, where each polygon is represented by its vertices. Particularly, positive polygons mean tumor region and negative polygons mean normal regions. The data on the website is in .xml form. You can also use the following command to convert the xml format into the json format.

*$ python /wsi/bin/camelyon16xml2json.py tumor\_001.xml tumor\_001.json*

* **Patch Images**

Although the original 400 WSI files contain all the necessary information, they are not directly applicable to train a model. Therefore, we have to sample much smaller image patches, e.g. 256x256, that a typical deep neural network can handle. Efficiently sampling informative and representative patches is one of the most critical parts to achieve good tumor detection performance. To ease this process, we have included the coordinates of pre-sampled patches used in the paper for training within this repo. They are located at “/wsi/cords”. Each one is a txt file, where each line within the file is in the format like tumor\_024,25417,127565 that the last two numbers are (x, y) coordinates of the center of each patch at level 0. tumor\_train.txt and normal\_train.txt contains 200,000 coordinates respectively, and tumor\_valid.txt and normal\_valid.txt contains 20,000 coordinates respectively. Note that, coordinates of hard negative patches, typically around tissue boundary regions, are also included within normal\_train.txt and normal\_valid.txt. With the original WSI and pre-sampled coordinates, we can now generate image patches for training deep GAN models. Run the six commands below to generate the corresponding patches:

*$ python ./wsi/bin/patch\_gen.py ./wsi/raw\_data/train ./wsi/coords/tumor\_train.txt ./wsi/patches/tumor\_train/*

*$ python ./wsi/bin/patch\_gen.py ./wsi/raw\_data/train ./wsi/coords/normal\_train.txt ./wsi/patches/normal\_train/*

*$ python ./wsi/bin/patch\_gen.py ./wsi/raw\_data/valid ./wsi/coords/tumor\_valid.txt ./wsi/patches/tumor\_valid/*

*$ python ./wsi/bin/patch\_gen.py ./wsi/raw\_data/valid ./wsi/coords/normal\_valid.txt ./wsi/patches/normal\_valid/*

*$ python ./wsi/bin/patch\_gen.py ./wsi/raw\_data/test ./wsi/coords/tumor\_test.txt ./wsi/patches/tumor\_test/*

*$ python ./wsi/bin/patch\_gen.py ./wsi/raw\_data/test ./wsi/coords/normal\_test.txt ./wsi/patches/normal\_test/*

where “/wsi/raw\_data/train” is the path to the directory where you put all the WSI files for training as mentioned above, and “/wsi/patches/tumor\_train/” is the path to the directory to store generated tumor patches for training. Same naming applies to other five commands. By default, each command is going to generate patches of size 256x256 at level 0 using 5 processes, where the center of each patch corresponds to the coordinates.

Training

**Training Model SST proposed by Cho [1]**

Training Procedure

1. Using GAN training samples, train ResNet34 for binary classification.
2. Based on the trained ResNet34:

Using current generator to get generated image () for input (x);

Compute the reconstruction loss;

Feed x and to ResNet34 to get features and compute feature preservation loss;

Feed x and to current discriminator to obtain GAN-loss;

Based on three losses to train GAN.

* **ResNet34**

With the generated path images, we can now train the model ResNet34, which is TranResNet34, by the following command:

*$ python ./TranResNet34/train\_with\_validation.py*

Please modify “./wsi/patches/tumor\_train'” (normal\_train, tumor\_valid, normal\_valid) respectively to your own path of generated patch images. Please also modify './wsi/jsons/train' (valid) with respect to the path of training (validation) repo on your machine. Typically, train\_with\_validation.py will generate a train.ckpt, which is the most recently saved model, and a best.ckpt, which is the model with the best validation accuracy and the model will be saved in file “/TranResnet34/save\_models”.

* **GAN**

With the generated patch images and trained TranResNet34 model, we can now train the GAN by following command:

*$ python ./sst\_train.py*

Please modify “./TranResnet34/save\_models/TranResnet34\_params\_50.pkl” to where you save your pretrained classifier. Please also modify “./wsi/patches/tumor\_train” (normal\_train) respectively to your own path of generated patch images. Please modify './wsi/jsons/train' (valid) with respect to the path of training (validation) repo on your machine. Typically, sst\_train.py will generate a D\_params.pkl and G\_params.pkl, which is the most recently saved model and the model will be saved in file “/save\_models/sst/”.

**Training our proposed model**

Training Procedure

1. Using GAN training samples, train ResNet34 for binary classification.
2. Based on the trained ResNet34:

Using current generator to get generated image () for input (x);

Compute the reconstruction loss;

Feed x and to ResNet34 to get features and compute feature preservation loss;

Feed x and to current discriminator to obtain GAN-loss;

Based on three losses to train GAN.

* **ResNet34**

With the generated path images, we can now train the model ResNet34, which is TranResNet34, by the following command:

*$ python ./TranResNet34/train\_with\_validation.py*

Please modify “./wsi/patches/tumor\_train'” (normal\_train, tumor\_valid, normal\_valid) respectively to your own path of generated patch images. Please also modify './wsi/jsons/train' (valid) with respect to the path of training (validation) repo on your machine. Typically, train\_with\_validation.py will generate a train.ckpt, which is the most recently saved model, and a best.ckpt, which is the model with the best validation accuracy and the model will be saved in file “/TranResnet34/save\_models”.

Note that if you have trained Restnet when training SST model, you don’t need to train ResNet again.

* **GAN**

With the generated patch images and trained TranResNet34 model, we can now train the GAN by following command:

*$ python ./dscsi\_train.py*

Please modify “./TranResnet34/save\_models/TranResnet34\_params.pkl” to where you save your pretrained classifier. Please also modify “./wsi/patches/tumor\_train” (normal\_train) respectively to your own path of generated patch images. Please modify './wsi/jsons/train' (valid) with respect to the path of training (validation) repo on your machine. Typically, sst\_train.py will generate a D\_params.pkl and G\_params.pkl, which is the most recently saved model and the model will be saved in file “/save\_models/dscsi/”.

TESTING

**Testing Model SST proposed by Cho [1]**

The main testing results from a trained model for WSI analysis is the classification result that represents whether the model thinks the image is tumor or normal. The testing image is from a second dataset-- Utrecht University and it can prove the generalization capacity of the model.

By using the following command:

*$ python ./sst\_test.py*

Please modify “./TranResnet34/save\_models/TranResnet34\_params.pkl” to your own path of saving pre-trained ResNet34 model and “./save\_models/sst/G\_params.pkl” to your path of saving SST model. Please also modify “./wsi/patches/tumor\_test” (normal\_test) respectively to your own path of generated patch images. Please also modify “./wsi/jsons/test” with respect to the full path to the repo on your machine. Typically, sst\_test.py will generate the result of classification task, including Accuracy, Precision, Recall and AUC, and save the generated images along with their original counterpart into file “/demo/sst/”

**Testing our proposed model**

The main testing results from a trained model for WSI analysis is the classification result that represents whether the model thinks the image is tumor or normal. The testing image is from a second dataset--Utrecht University and it can prove the generalization capacity of the model.

By using the following command:

*$ python ./dscsi\_test.py*

Please modify “./TranResnet34/save\_models/TranResnet34\_params.pkl” to your own path of trained ResNet34 model and “./save\_models/sst/G\_params.pkl” to your path of saving SST model. Please also modify “./wsi/patches/tumor\_test” (normal\_test) respectively to your own path of generated patch images. Please also modify “./wsi/jsons/test” with respect to the full path to the repo on your machine. Typically, sst\_test.py will generate the result of classification task, including Accuracy, Precision, Recall and AUC, and save the generated images along with their original counterpart into file “/demo/dscsi/”.

Generating

If you want to use obtained model from training to generator images. You can use sst\_generator.py and dscsi\_generator.py.

By using the following command:

*$ python sst\_generator.py* # It generates images using SST model.

*Or:*

*$ python dscsi\_generator.py* # It generates images using our proposed model.

Please modify the “source\_path”, “save\_path” as well as “model\_path” in the python file respective to where you store your source images, where you want to save generated images and where you save your pre-trained model.

[1] Hyungjoo Cho, Sungbin Lim, Gunho Choi, and Hyunseok Min. Neural stain-style transfer learning using gan for histopathological images. CoRR, abs/1710.08543, 2017.