LT2326
Assignment 1
KAMANEH AKHAVAN
Part 1:

At first preparing the data was so challenging, data were in two types:
As can be seen firstly, a folder, which contains 845 images of size 2048x2048 that had been used for training in the original task; and secondly, two files that contain the information of the images, 'info.json' and 'train.jsonl'. The first step was knowing the files and the relation between them. So I tried to find out the relation between the info.json and train.jsonl.I tried to compare the one line in each file and I found that there is a connection with the image_id in both. As a first step, I tried to open the info.json and extract the image_id and put them in the other file and then open the train.jsonl (which has more details about the images)and compare it with the image_id file and then save the image's information with the same image_id. By comparing the names of the images in the folder with the names in the data file, there were 845 images that could be used for assignment. They were separated into a train dataset, containing 80% of the images, an validation dataset containing 10% of the images, and an experimental dataset containing 10% of the reminders. Train.jsonl contained information about the images. The images used in this challenge contain Chinese characters marked with a polygon that surrounds each one. The information used for the assignment was the image name and coordinates of the polygons. There were also polygons that included other characters. These non-Chinese characters were never used, although they were saved because the original idea was to do so. By obtaining the names of the images and the coordinates of the polygons containing the Chinese characters, the goal was to load the images and obtain the tensors that represent each of their pixels. At the same time, we needed to get what our models wanted: display the probability of the image containing a Chinese character (which would be 1 if a character exists and 0 if not). Tensors were created with the function. However, due to the large size of the images (2048x2048), they can never be loaded. For this reason, the image size was changed to 200x200 pixels before it became a tensor. This function, after working on creating the models and seeing that the first layer of both models is a convolution layer, also changed the last dimension of the image to the first position. Expected output was generated for images with the last function. In this case, we retained the original image size and produced a one-dimensional tensor containing 0 or 1, depending on whether the pixel was in a polygon (and therefore contained a Chinese character). It took a long time to load the data and it takes about 15 minutes for each load. And although I had to refresh the page once so that I would not get the error of CUDA out of memory.

Part2

In this task, two CNN models were trained using a smaller portion of the given dataset from .After the data pre-processing for supervised training, the models architecture, and the evaluation of the results/outputs of the trained models. The main Python codes for pre-processing, model design and training, and evaluation are in the kamanehakhavan.ipynb notebook. The two saved models model1 and model2 are in the same GitHub directory. Note that if using test images as input, the images should be converted to a PyTorch tensor of shape (N, H, W, 3). The input/output of the two models also slightly differs: model1.pt which the architecture is inspired by https://d2l.ai/chapter_convolutional-neural-networks/lenet.html , consists of multiple layers that increases the number of channels while downsizing the images. The downsized images then go through two linear function and a sigm function, then upsampled back to 2048*2048.It takes resized images (N, 256, 256, 3) and outputs a prediction (N, 2048, 2048, 1); while model2.pt has the middle layers similar to the ones mentioned on

LT2326
Assignment 1
KAMANEH AKHAVAN

https://ravivaishnav20.medium.com/visualizing feature-maps-using-pytorch-12a48cd1e573 with the 17 convolutional layers with different parameters from, in the hope that these deeper layers would be able to extract the features, the Chinese characters. The full ResNet with BasicBlock seemed complex so I did not attempt to replicate the whole architecture. Instead I only added the abovementioned layers . Then layer 17 shrinks the tensor to (num_channels, 1, 1), in this model being (512, 1, 1). The final two layers then linearise and take the sigm, and upsample the 1D tensor to 2048*2048.It should be able to take images of unspecified size (N, H, W, 3) and outputs two predictions, both in the shape (N, 2048, 2048, 1).

Part 3: testing and evaluation The test function prints out the accuracy of the models, as well as the total error, computed with mean squared error, and its plot, after upsampling the images back to 2048x2048. The accuracy for both model slightly the same, for the first one was 99.55% and for the second one was about 99.56. The mean squared error is almost the same.However, the results in the validation of the test function look different: Model 1 has a slightly smaller square root mean square error and fewer peaks, although they reach a higher value. This could indicate that using a lower learning rate during exercise could yield slightly better results, although the test set did not confirm that the accuracy of the two was very close.Finally, the "pic_generator" function visually displays the pixel probabilities on the original image. Applies this model to separate images and returns the model output. We see that it is a little better and clearer for the first model than the second model. And according to the output of the model, it can recognize only some letters.

References

https://realpython.com/working-with-files-in-python/

https://towardsdatascience.com/how-to-split-data-into-three-sets-train-validation-and-test-and-why-e50d22d3e54c

https://matplotlib.org/stable/tutorials/introductory/images.html

https://pytorch.org/docs/stable/generated/torch.reshape.html

Illustrated: 10 CNN Architectures | by Raimi Karim | Towards Data Science

6.6. Convolutional Neural Networks (LeNet) — Dive into Deep Learning 0.17.1 documentation (d2l.ai)