Programming Exercise Digital Pathology and Deep Learning (Mitosis Domain Generalization)

Ahmed Sheta
22985354
ahmed.sheta@fau.de

***Abstract*—In this report, I would like to answer the questions of the programming exercise of seminar Digital Pathology and Deep Learning to my best knowledge**

I. Task 1

A. Essential characteristics in the data and labels

The data set consists of 200 cases of human breast cancer. Each of the images was cropped from a whole slide image. The data set consists of 200 cases of human breast cancer and is collected from 4 different scanners. . All images come from the same lab (UMC Utrecht) and have thus similar staining intensity.

The images 001.tiff to 050.tiff were acquired with a Hamamatsu XR scanner. The images 051.tiff to 100.tiff were acquired with a Hamamatsu S360 scanner. The images 101.tiff to 150.tiff were acquired with an Aperio CS2 scanner. The images 151.tiff to 200.tiff were acquired with a Leica GT450 scanner.

All the images from 001 to 150 were labeled, whereas the images from 151 to 200 were not labeled. The labels of the dataset contain 2 classes, first class is mitotic figures, and second class is imposter class. The imposter class are the annotations where experts disagreed or that qualify as hard examples for machine learning.

B. The problem in dpdl_defaults.py

The problem is that images with ID from 151 to 200 have no labels, because this training is done in a supervised manner. In order to correct this problem, we want to remove every of those images from the TRAINING_IDS and TEST_IDS. They still provided these unlabeled images to provide a reference for additional visual representation and to provide an opportunity to implement different methodologies like unsupervised domain adaptation approaches or weakly supervised learning.

C. RetinaNet implementation and important parameters in network.py

The constructor of this network generates RetinaNet ResNet-50 [1] network which is widely used for object detection. This network uses ResNet-50 architecture as its backbone for feature extraction followed by RetinaNet for object detection. The model retinanet_resnet50_fpn_v2 is pretained and initialized with off-the-shelf COCO weights and the model head is initialized randomly. The 5 backbone layers were frozen, meaning that there are not trainable. It is beneficial to fine-tune the backbone, since the COCO images are far different from the histopathology whole-slide images (WSI. In order to unfreeze them we set the constructor's parameter trainable_backbone_layers to the number of layer we want to unfreeze. The important parameters are the number of classes, the learning rate and the detect threshold value. The learning rate could be optimized with Adam optimizer, and the detect threshold value could be optimized as well.

II. Task 2: Data sampling during training

A. Sampling using mitotic figures

There is a high class imbalance in the dataset; The training set contains 1721 mitotic figures (MF) and 2714 hard examples (non-mitotic figures) [2]. Random patch sampling will make it hard for the network to crop patches that have mitotic figures. In order to alleviate that problem, we use the my_improved_sampling_func that has a probability of 50% to sample a patch that centers a random mitotic figure, if the region of interest (RoI) contain any mitotic figures. Please kindly find my code implementation in training.py file. We want to use that sampling on the training dataset but not on the validation dataset as it might provide overconfident results.

B. Sampling using mitotic figures and imposter figures

The imposter class are the annotations where experts disagreed on whether it is a mitotic figure or not. They are not useful for the network because it contains both mitotic and non-mitotic figures, which could very well mislead the network in identifying the right discriminate features for the mitotic figures or worse, classify the true mitotic figures into another class. I believe they are still provided to give the opportunity to to leverage other machine learning methods, like unsupervised learning where we can expand on these data by clustering them into 2 different classes, true mitotic and non-mitotic.

III. Task 3: Inferences for full RoIs

We want generate overlapping patches from the WSI in a deterministic manner using a continuous index. Since the mitotic figures are almost half of the imposter figures as mentioned in Task 2, I believed it was best to double sample patches that contain overlapping mitotic figures. There I sampled every mitotic figure twice; One patch would be have it on the bottom right corner while the other patch have it on the top left corner. Despite trying to implement a stride method with

overlapping 50% overlap between patches along the width dimension but that does not guarantee that the patch would have overlapping mitotic figure within the overlap region. Kindly find my implementation in file midog_dataset.py.

IV. Task 4: Training and tuning

The best achieving model was trained for 2400 epochs which took 18 hours and 8 minutes using the A100 graphic processing unit (GPU). Regarding the hyper-parameters, the learning rate began from 0.0001 and was being optimized using Adam. The batch size was 30, despite the relatively big capacity of the A100, it wasn't possible to train on a bigger batch size. The bigger the batch size is, the less erratic the training loss becomes. The patch size was the default value of 512, the number of patches in both cases of the training and the validation was 10. Since the COCO dataset is very different from the histopathology WSIs, it was beneficial to fine tune the whole backbone network. Hence, I unfroze the 5 layers to ensure robust performance and convergence. From the short review of the model, the results of the testing emerge: f1 score is 0.795, precision is 0.965, and recall is 0.676. While training the model, figure 1 highlights the course of the loss minimization for 2400 epochs, it is reasonable to deduct that the model began to converge earlier than the end of 2400 epochs. This is also proved while I trained another network for only 800 epochs and achieved a testing f1 score of 0.715, which is a relatively good score considering training only for one-third the amount of the epochs of the best achieving model I trained. Figure 2 illustrates how the validation loss is minimized, we can see that they are similar that of the training loss. Figures 3, 4, and 5 represent the validation f1 score, validation precision and, the validation recall curves respectively. I have found it best to screenshot the plots from tensorboard in order to include the smoothing curves to convey meaningful information, using vector graphics would not have enabled that. Considering the aforementioned results of testing, the high precision score of 0.965 suggests that the model is able to accurately identify positive results and avoid false positives, while the relatively lower recall score of 0.676 suggests that the model may be missing some positive cases. Overall, the F1 score of 0.795 indicates that the model is performing fairly well, but there may be some room for improvement, particularly in terms of recall. I have trained another network for 3000 epochs, which took 22 hours and achieved a validation f1 score of 0.785, relatively lower than the best achieving model although it was trained for more epochs.

V. Conclusion

Using a guided patch sampling that have 50% chance of containing mitotic figures if there were some had a positive impact on the training process; With random patch sampling, the network was having a hard time receiving the positive samples to be trained on. Considering the huge class imbalance in this exercise, RetinaNet with its focal loss has proved to be robust to the class imbalance. On top of that using oversampling during testing to ensure overlapping patches having overlapping mitotic figures aimed to alleviate the class imbalance problem.

References

[1] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[2] M. Aubreville, C. Bertram, M. Veta, R. Klopfleisch, N. Stathonikos, K. Breininger, N. ter Hoeve, F. Ciompi, and A. Maier, "The dataset of midog challenge," https://imig.science/midog2021/the-dataset/, 2021.

Appendix

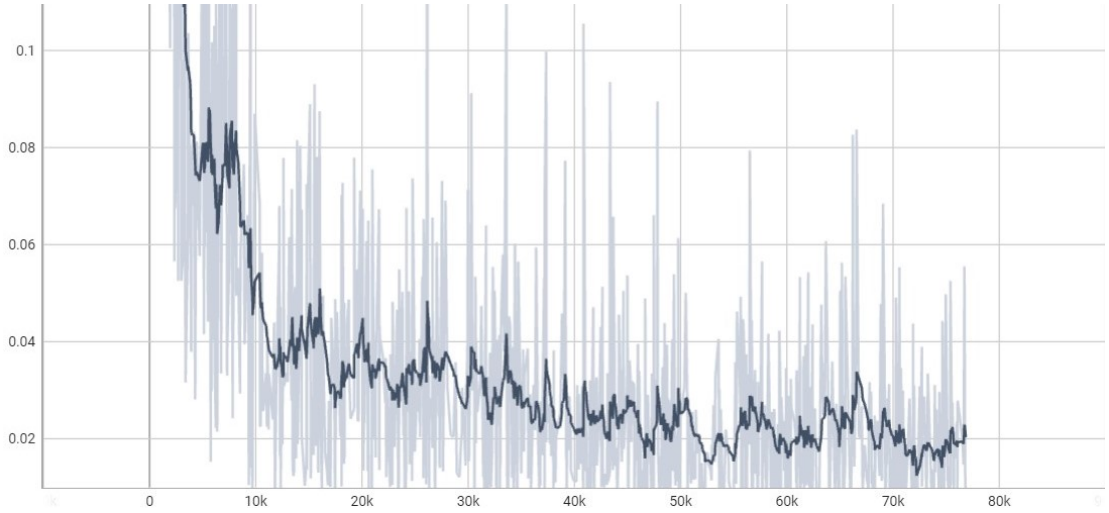

Fig. 1: Training loss. X-axis represents the number of steps, Y-axis represents the loss, and the smoothing scalar is 0.89.

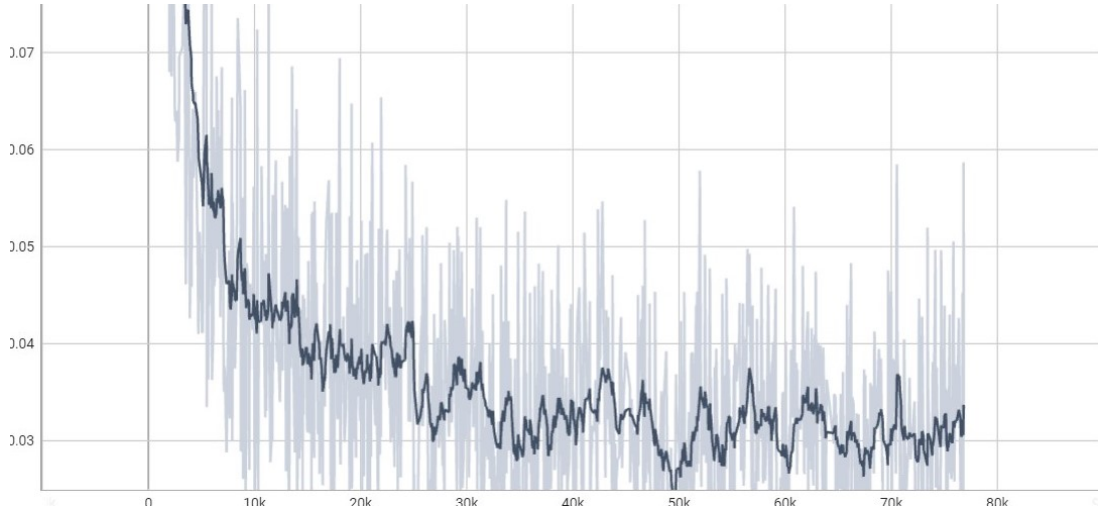

Fig. 2: Validation loss. X-axis represents the number of steps, Y-axis represents the loss, and the smoothing scalar is 0.89.

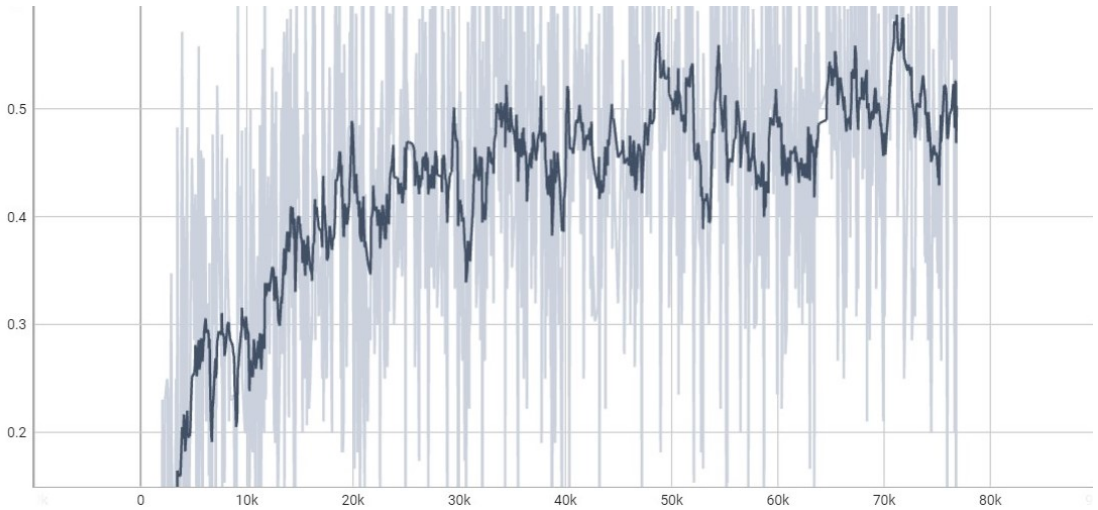

Fig. 3: Validation F1 score. X-axis represents the number of steps, Y-axis represents the f1 score, and the smoothing scalar is 0.89.

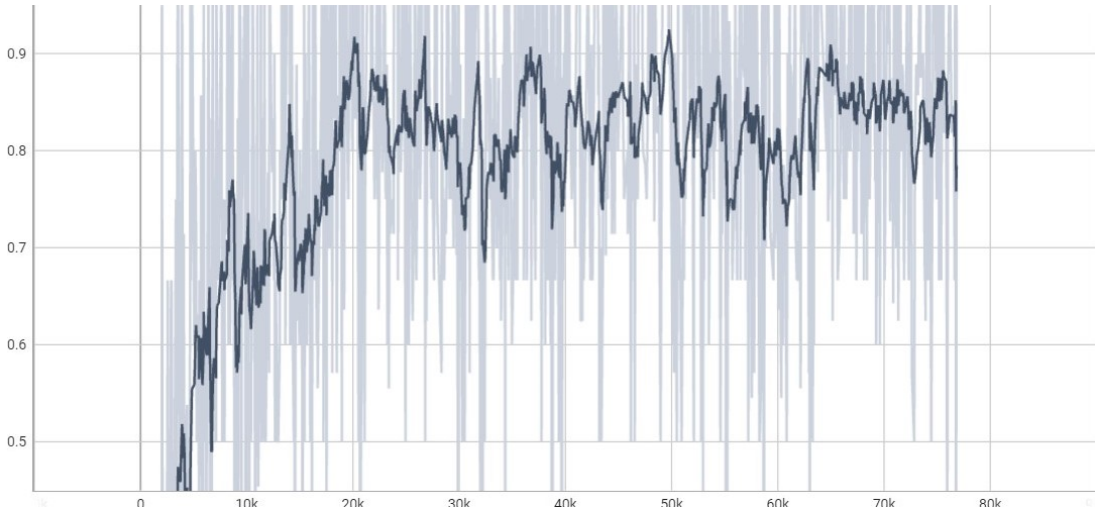

Fig. 4: Validation precision. X-axis represents the number of steps, Y-axis represents the precision, and the smoothing scalar is 0.89.

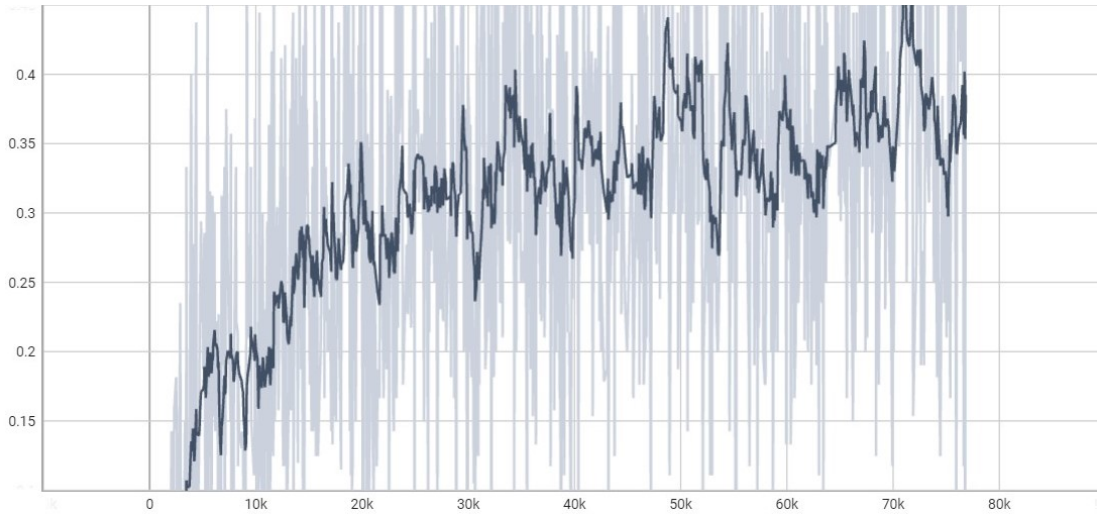

Fig. 5: Validation recall. X-axis represents the number of steps, Y-axis represents the recall, and the smoothing scalar is 0.89.