Homework 6
601.482/682 Deep Learning
Spring 2023

March 16, 2023

Due 11:59pm on Apr 5, 2023

Please submit 1) a single zip file containing your Jupyter Notebook and PDF of your Jupyter Notebook to "Homework 6 - Notebook" and 2) your written report (LaTeX generated PDF) to "Homework 6 - Report" on Gradescope (Entry Code: DJGEWX)

Important: You must program this homework using the PyTorch framework. We highly recommend using Google Colaboratory.

Important: If you don't have local GPU access, you should port the provided Python scripts to Colaboratory and enable GPU in that environment (under Edit->Notebook Settings). Training should converge in less than 30 min. If your model does not make significant updates in that time, you should re-examine your code. Either way, this is a reminder to start the assignment early.

1. *Unsupervised Pre-training.* In this problem, you will attempt the 2017 Endoscopic Instrument Challenge.[1] You are given a pre-processed dataset consisting of endoscopic frame images (*not* in sequential order). The goal is to train a network which takes each RGB frame as an input and predicts a pixel-wise segmentation mask that labels the target instrument type and background tissue. Additionally, we introduce an unsupervised pre-training method and compare the performance of training on a small labeled dataset with/without pre-training. This is relevant for real-life medical image problems, where there is usually a shortage of data labels.

   Data Folder We have provided a well-structured dataset. It consists of '/segmentation' and '/colorization'. In each sub-folder, there are '/train' and '/validation' for training purposes.

   The **main goals** of the homework are as follows. Concrete TODOs are enumerated on the next page.

   - *Complete the Network structure for segmentation task (a-c).* The network structure we provide is a simplified U-Net, which is a very popular framework in medical image segmentation task. Read and understand the code in `unet.py`, the implementation is missing the last layer and the last activation function. Next, fill in the missing components for 1(a). For the segmentation task, train ONLY with the frames in '/segmentation/train' and validate and test with the frames in '/segmentation/validation' and '/segmentation/test' respectively. The original input image is a $256 \times 320 \times 3$ RGB image. The ground truth label is a grey-scale image that has the same dimension, where different gray values indicate the instrument type or background tissue.
   - *Pre-training by self-supervised colorization (d).* Image colorization training[2] is a common way of self-supervised learning.[3] The idea is to take grey-scale images as an input and predict colorized images, similar to filling in colors in a draft painting. You must

[1] `https://endovissub2017-roboticinstrumentsegmentation.grand-challenge.org`

[2] Larsson, G., Maire, M., & Shakhnarovich, G. (2017). Colorization as a proxy task for visual understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6874-6883).

[3] Jing, L., & Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence.

use the **same** U-Net structure as above to train a model for this colorization task. Then use the pretrained weights as initialization for the segmentation task.

For the colorization task, you need to train on '/colorization/train_cor' and validate on '/colorization/validation_cor'. You are given `mapping.json` that contains the label of each grey level. We have also provided grey-scale image for each input in each subfolder of '/colorization' for your colorization task input.

Now that you know the broad goals and data sets, please complete the following TODOs.

(a) Train a segmentation network using the frames in the '/segmentation/train' folder. Please complete the DICE score function to evaluate your model, and write from scratch a DICE loss function as your network loss[4]. (*Hint: You need to convert the grey-scale label mask to one-hot encoding of the label and then calculate the DICE score for each label*). Please train the network until convergence (should take around 30 min) using the default provided hyperparameters and provide a figure of training loss and validation loss w.r.t. epochs (in a single figure). Please report your performance (DICE score) on the test dataset, you should expect a DICE score > 0.5. (*Hint: Using BatchNorm might help you achieve better performance.*)

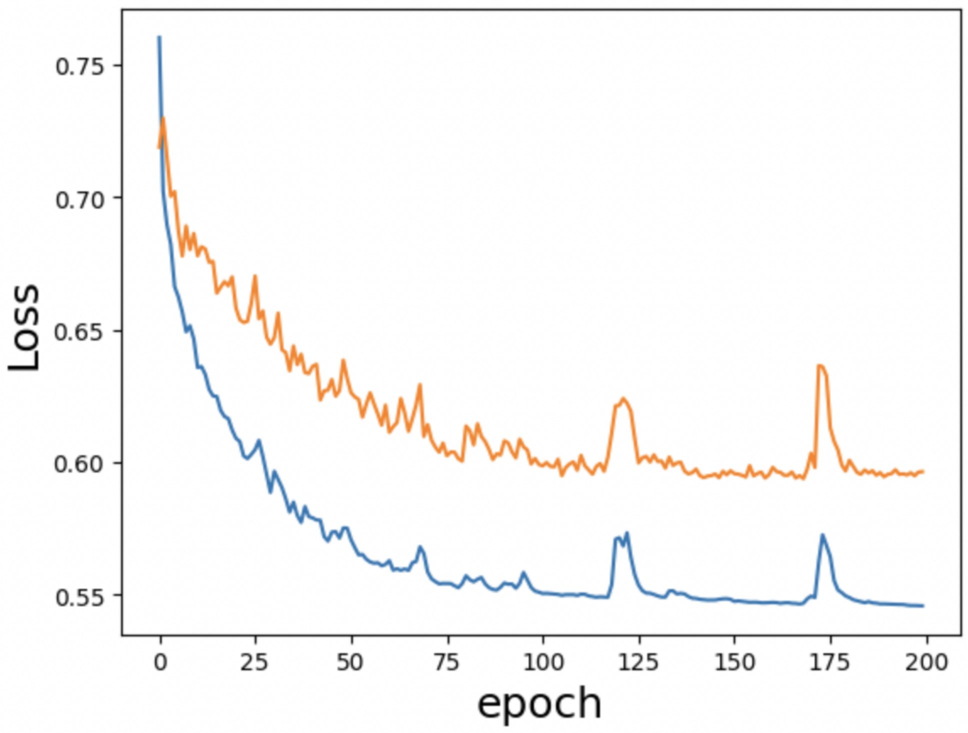

Figure 1: Loss v.s Epoch
The orange curve is validation loss. The blue curve is the training loss

DICE score on Test dataset: **0.6119**

I use a default hyperparameter with a batch size of 10 and a learning rate of 0.001. I run 200 epochs in total.

(b) Introduce meaningful data augmentation (e.g. vertical and horizontal flips) and train the network until convergence using the same hyperparameters as (a). Please plot the training loss and validation loss on a single figure again and report test dataset performance, you should expect a DICE score > 0.6.

[4]Read more about the DICE score: `https://medium.com/datadriveninvestor/deep-learning-in-medical-imaging-3c1008431aaf`

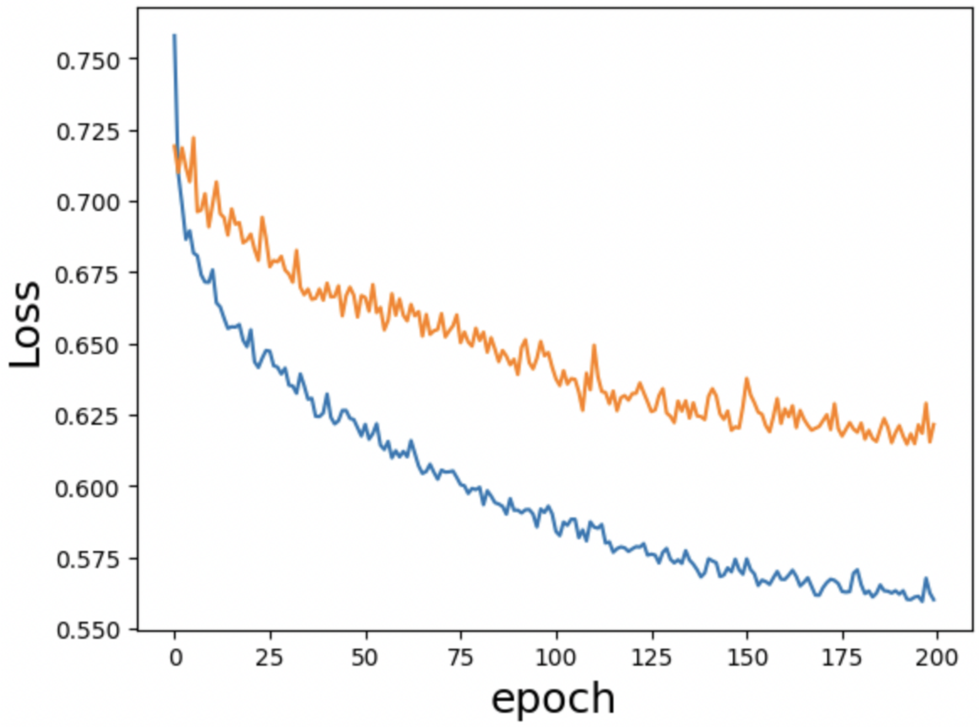

Figure 2: Loss v.s Epoch
The orange curve is validation loss. The blue curve is the training loss

DICE score on Test dataset: **0.6305**

I apply vertical flips with 0.5 probability, and horizontal flips with 0.5 probability. I use the same hyparameter as in 1(a).

(c) Train on the colorization task using frames from the '/colorization/train_cor' folder. Use hyperparameters that seem reasonable (based on your previous experiments) and mean squared error as your loss function. Please provide a figure of training loss w.r.t. epochs until your model converges. Then save your model to initialize the network for the next task.

I use a batch size of 7 and a learning rate of 0.00005. I train my model for 50 epochs.

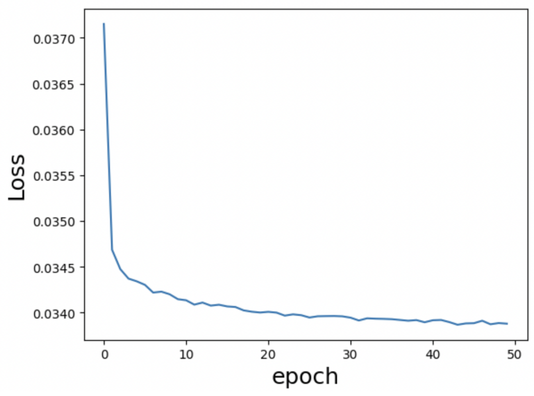

Figure 3: Training Loss v.s Epoch

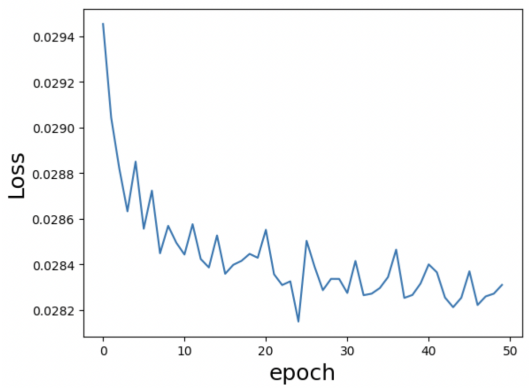

Figure 4: Validation Loss v.s Epoch

(d) Load the colorization pre-trained model and start training for the segmentation task using the frames in the '/segmentation/train' folder. Make sure you are using the same hyperparameters as you did in the former task, and please clearly state them in your report. Plot the figure of training loss and validation loss. Report test dataset performance. Do you see a difference with the former result in (b)? (*Hint: Since this is a relatively simple dataset, you might not actually observe differences in performance.*)

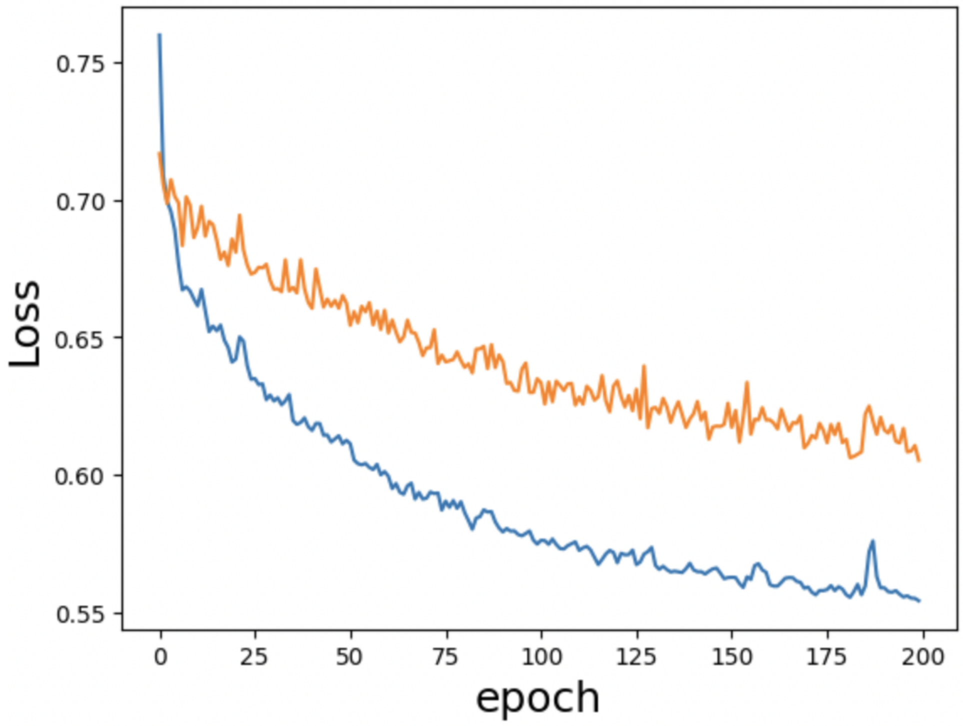

Figure 5: Loss v.s Epoch
The orange curve is validation loss. The blue curve is the training loss

DICE score on Test dataset: **0.6922**

I use the same hyperparameter as I use in 2(b): a batch size of 10, and a learning rate of 0.001. I run 200 epochs in total and apply horizontal and vertical flips to my dataset. I retrain the last layer so it gives an output channel of 6 classes, not 3. My DICE score increases by 0.06 after pretraining the model by colorization. There's no significant difference, but I can still see some improvements.

2. *Transfer Learning.* Please download the fashion MNIST dataset [5] as used in HW4 and download the VGG16 model (`https://pytorch.org/docs/stable/torchvision/models.html`).

   (a) Randomly initialize all parameters in VGG16 and try to train your model to learn the Fashion MNIST classification task. What's the accuracy you achieve? Please report your test accuracy on the test dataset. You should expect an accuracy $> 85\%$.

   Test accuracy: **0.9209**

[5]The full FashionMNIST dataset can be downloaded from the official website here: `https://github.com/zalandoresearch/fashion-mnist`

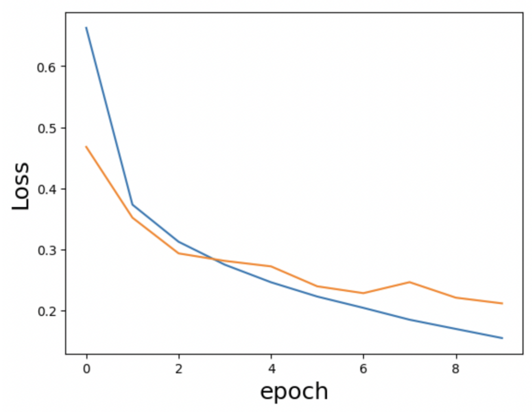

Figure 6: Loss v.s Epoch

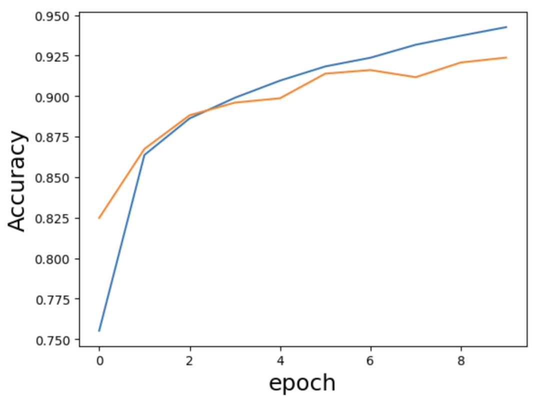

Figure 7: Accuracy v.s Epoch

The orange curve is validation loss/accuracy. The blue curve is the training loss/accuracy.

(b) Load the pre-trained VGG16 model from torch vision models. Freeze all but the last layer: randomly initialize the last layer of your network and fine-tune this. What accuracy do you get now? Please again report your test accuracy on the test dataset. You should expect an accuracy $> 60\%$.

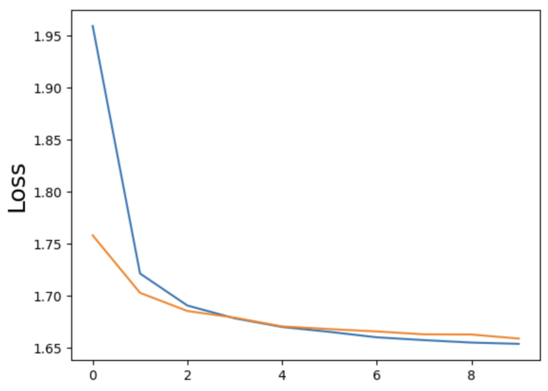

Figure 8: Loss v.s Epoch

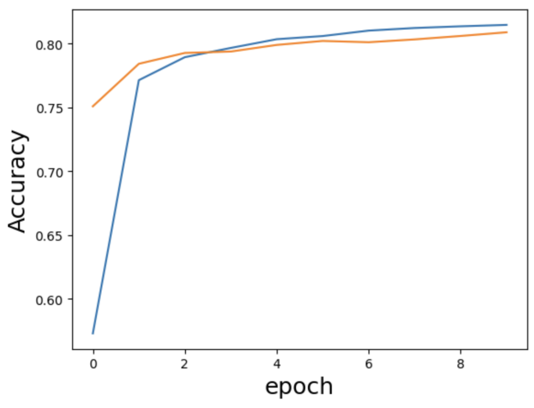

Figure 9: Accuracy v.s Epoch

The orange curve is validation loss/accuracy. The blue curve is the training loss/accuracy.

Test accuracy: **0.8009**

(c) Now, imagine a scenario in which you want to train the VGG16 model on an entirely new dataset and will fine-tune either the model from (2a) or (2b). Which pre-trained model is the preferred starting point for your new use case?

The model (2b) is preferred. VGG16 is trained on ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

However, our model in (2a) only trained on FashionMNIST, which only contains 60,000 grayscale images with 10 classes. Our model only trained for 2 hours on GPU.

In transfer learning, we take advantage of the early layers as effective feature extractors to the new dataset. Model (2b) will be a much better feature extractor given the diverse data it has seen. On the other hand, our model in (2a) only saw grayscale images, which will not be a good feature extractor for other tasks.