# Assignment 3 (100 points)

For assignment submission, you will a report file (.pdf) + two python script files (.py) on Canvas. The necessary output and results of codes should be included in the report file as well as the responses to text and analysis questions. Your submitted files **must** follow these guidelines:

* No other dataset than the provided datasets should be used.
* Training, validation and testing splits should be the same as the ones provided.
* Code outputs in your delivered report should be reproducible.
* Printing out the evaluation metric evidence that your model achieves the evaluation requirement. Optionally, you can also add plot of the evaluation metric changing over the course of training process.
* Provide code associated with the conclusions you make in your analysis as well as code that is used to generate plots, images, etc. involved in your analysis.
* All code must be your own work. Code cannot be copied from external sources or other students. You may copy code that are pre-defined in the given codes if you think it is useful to reuse in another question.
* All images must be generated from data generated in your code. Do **NOT** import/display images that are generated outside your code.
* Your analysis must be your own, but if you quote text or equations from another source please make sure to cite the appropriate references.
* Your input with code will be marked with comments ###your code starts here (Ex 1.x) ### and ###your code ends here### to specify where you need to write your code.

## Notes:

* PyTorch needs to be downloaded and installed properly (check <https://pytorch.org/get-started/locally/>)
* You should use PyTorch 1.7 or later.
* It is strongly recommended to create a python virtual environment and install the required packages there to have a clean environment that isolates your project’s dependencies from system-level packages and avoids conflicts with other environments. For doing so, you can check <https://chpc.utah.edu/documentation/software/user-installed-python/venv.php>. To avoid any issues in installing of packages, please use CHPC’s setup script (part 1.1 of the above web page).
* You should load your virtual environment and cuda module each time you login to the servers
  + module use /uufs/chpc.utah.edu/common/home/<uID>/<my\_py\_modules>/ (<uID> is your account name which is your uID by default, <my\_py\_modules> is the directory you specified in CHPC’s wizard for putting your virtual environments in)
  + module load <my\_env> (<my\_env> is the name of the virtual environment you created using CHPC’s wizard)
  + module laod cuda/11.8.0 (assuming you installed pytorch with cuda 11.8.0 package)
    - Install PyTorch after loading cuda
* If you need to import a different package than the ones already imported, please check with the TA if you can do so.
* Read all the exercise descriptions and also the comments in the codes as they contain hints and information that you may need to use to complete assignment.
* Due to the large size of some of the datasets, they are already deployed on CHPC shared scratch disks, and you should be able to access them. For the exercises that need such datasets, the full paths of the data files are mentioned in the exercise description.
* The accuracy requirement for each question is there to make sure you have performed sufficient amount of experiments to achieve a good result. Part of the grade is based on this.

## Tips for training deep learning models:

* We assume a GPU of at least 4GB of memory is available. If you want to try running the assignment with a GPU that has less than that, you can try changing the argument passed when calling the define\_gpu\_to\_use function. If you are getting out-of-memory errors for the GPU, you may want to check what is occupying the GPU memory by using the command nvidia-smi in your slurm script, which gives a GPU's usage report. However, if you are using your own Windows machine, the nvidia-smi command used in the define\_gpu\_to\_use function will not work. You can skip running this function but please check to make sure your GPU has a sufficient amount of free memory.
* Here are a few PyTorch details not to forget:
* Toggle train/eval mode for your model
* Reset the gradients with zero\_grad() before each call to backward()
* Check if the loss you are using receives logits or probabilities and adapt your model output accordingly.
* Reinstantiate your model every time you are starting a new training so that the weights are reset, if you plan to reuse the variable name.
* Pass the model's parameters to the optimizer.

## Exercise 0 - Setting-up Infrastructure (Total of 0 points)

### Exercise 0.1 - Installing Libraries:

Follow the steps below to install a few more additional libraries:

* Open another terminal and activate your virtual environment source <your\_virt\_env\_root\_folder>/bin/activate.
* Install the following libraries via pip using these commands: pip install scikit-image, pip install scikit-learn, pip install imageio, and pip install imagecodecs.

### Exercise 0.2 - Requesting GPU Usage:

First, we check which host the code is running on (CHPC, CADE, …) as different hosts have different configurations for getting compute jobs.

Then, we define a function to specify the suitable gpu to request (you have seen this in previous assignment as well)

This function locates an available gpu for usage. In addition, this function reserves a specified memory space exclusively for your account. The memory reservation prevents the decrement in computational speed when other users try to allocate memory on the same gpu in the shared systems, i.e., CADE machines.

***Note:*** If you use your own system which has a GPU with less than 4GB of memory, remember to change the

specified minimum memory.

## Exercise 1 - DRIVE and STARE Dataset (100 points)

In this exercise, we are going to implement a segmentation model to extract blood vessels in retinal digital images. We will utilize two well-known retina image datasets, DRIVE ([link](https://www.isi.uu.nl/Research/Databases/DRIVE/)) and STARE ([link](http://cecas.clemson.edu/~ahoover/stare/)).

First, we need to download these datasets (**you can ignore this step since the datasets are downloaded and come with the assignment package**). Please follow the steps below to download the images and then put them in the same directory as this notebook:

* Register an account with DRIVE dataset - [link](https://grand-challenge.org/accounts/login/?next=https%3A//drive.grand-challenge.org/participants/registration/create/)
* Download datasets.zip file at [link](https://drive.grand-challenge.org/Download/)
* Download stare-images.tar from [link](http://cecas.clemson.edu/~ahoover/stare/probing/stare-images.tar)
* Download labels-vk.tar from [link](http://cecas.clemson.edu/~ahoover/stare/probing/labels-vk.tar)

Next, we are going to define a few functions to extract images from the downloaded file for DRIVE and STARE datasets, respectively.

### Exercise 1.1 [A] - Data Augmentation (12 points)

* Data augmentation is an essential step in training deep models. Using data augmentation usually helps to improve the performance of a deep model. Please write a short reasoning why data augmentation helps to improve the performance.
* For this exercise, we will be using horizontal and vertical flipping. Your task is **first** to provide a short reasoning why horizontal and vertical flipping is suitable for this dataset. **Second**, please list at least one other data augmentation method that could also be used for this dataset.

Next, you will implement the horizontal and vertical flipping transformations by filling out the missing codes in the respective code file.

### Exercise 1.2 - Computing Weight Vector (10 points)

In many real-world applications, the dataset tends to be imbalanced. It is also the case for this retina image dataset. In this dataset, there are more background samples than there are foreground samples.

#### Exercise 1.2.1 [C]

In this exercise, your task is to examine the imbalanced between foreground and background by providing the following analyses in the code to estimate the following:

* The percentage of positive label (foreground) in this dataset
* The ratio of negative label (background) to positive label (foreground)

#### Exercise 1.2.2 [C]

Now, we are going to use the ratio of negative label to positive label as weight vector for the loss function. In the code file, please provide an implementation of a weighted loss function for a binary classification problem.

### Exercise 1.3 - Implementing U-Net (60 points)

#### Exercise 1.3.1 [C]

In this section, you are going to implement a well-known segmentation model, called U-Net. Your implementation should follow the architecture as described in the paper (<https://arxiv.org/pdf/1505.04597.pdf> - Fig. 1 and Section 2) with a few modifications below:

* The input should have 3 channels, and the output should have only one channel (binary output).
* Adding 2D batch normalization layer between convolution layer and Relu transformation, i.e., changing CONV->RELU to CONV->BN->RELU.
* Padding the convolution layers so that the outputs of the convolution layers have the same spatial size as the inputs. With this modification, the cropping operation before the concatenation in the skip connection can be removed.
* Upsampling operation should be implemented with the torch.nn.ConvTransposed2D layer. More details to understand what they meant in the paper can be found in the video here (<https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/>) starting at 2:22.
* Reducing the number of channels of **ALL** internal layers 4 times. For example, the number of channels in the first convolution layer in the paper is 64. The first convolution layer in your implementation should have 16 channels instead.
* No need to implement the initialization of weights as described in the paper. The default weight initialization from PyTorch is sufficient.
* Your U-Net implementation should be named model\_ex13 and the best model should be called best\_model\_ex13.
* Your implementation of U-Net should achieve F1 score of at least 0.75 on the validation set.

#### Exercise 1.3.2 [C] – Implement the training function

Next, your task is to complete the training\_stage function below. You must use the provided masks when computing the loss. In other words, you must only compute the loss for the pixels that have values of 1 in the corresponding mask images.

#### Exercise 1.3.3 - Visualizing output (8 points)

In this section, please plot the predicted output of a few samples from the validation set as well as the corresponding ground-truth. In addition, please provide a short analysis on the type of mistakes that you are able to distinguish. For example, where in the image does the model wrongly identify as foreground? where in the image does the model wrongly identify as background?

### Exercise 1.4 - Implementing U-Net Without Skip Connections (18 points)

#### Exercise 1.4.1 [C]

In this section, your task is to implement a similar U-Net model, called model\_ex15, as specified in Exercise 1.3 with the exception that this model does **NOT** contain any skip connections. To compensate for not having the extra channels coming from the skip connection, you need to double the number of output channels in the upsampling layer.

#### Exercise 1.4.2 [C]

Again, please provide a similar visualization as Exercise 1.3.

#### Exercise 1.4.3 [T]

As seen qualitatively from the visualization section and quantitatively from the F1 score, the U-Net model without skip connections performs poorly specifically in the thin section of the foreground. In this exercise, please provide a short reasoning of

1. which causes the model to performance poorly in the thin section of the foreground?

2. Why the skip connections help to overcome that limitation?