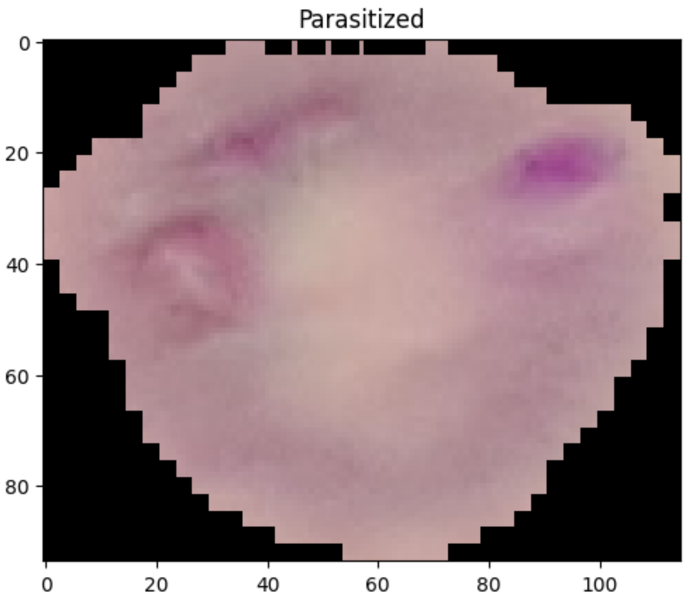
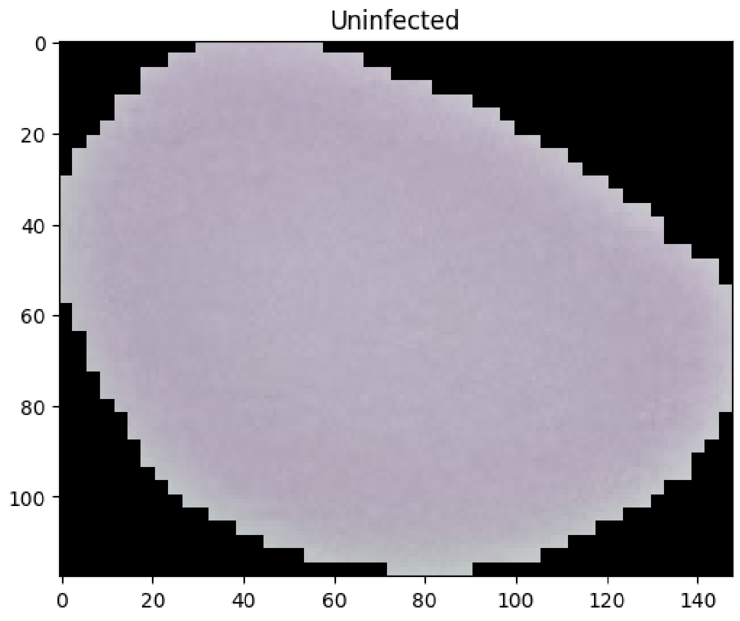
Anushka Sinha | 02718 HW 5

Section 1. Download dataset (10 pts)

Follow the instructions in the notebook to download the Malaria dataset from Google Drive. You can also download it by using this link. Explore the dataset. How many dimensions are there in an image? Show one image of a parasitized blood smear and one image of an uninfected blood smear here.





Dimension of the parasitized image: 115 x 94

Dimension of the uninfected image: 148 x 118

Section 2. Prepare your dataset and data loader (20 pts)

Pytorch is a deep-learning framework that makes our lives easier when we would like to build deep-learning models in Python. There are many Pytorch tutorials around, and its documentation is complete and extensive. **Make sure you read those materials first.**

Implementing the Dataset and Dataloader class for this assignment is straightforward. You will use the ImageFolder class from the torchvision library: ImageFolder. To use this class, the image dataset folders should be of the following structure:

```
├── training data
│ ├── class 0
│ │ ├── image 1.png
│ │ ├── image 2.png
│ │ ├── …
│ ├── class 1
│ │ ├── image 1.png
│ │ ├── image 2.png
│ │ ├── …
├── validation data
│ ├── class 0
│ │ ├── image 1.png
│ │ ├── image 2.png
│ │ ├── …
│ ├── class 1
│ │ ├── image 1.png
│ │ ├── image 2.png
│ │ ├── …
```

If you observe closely, this is how our Malaria dataset is structured. The ImageFolder class automatically infers the labels and makes a Dataset object, which we can pass on to the Dataloader. Since data augmentation is essential for good classification performance, you will need to do image transformations to the dataset class for data augmentation. You should read the Pytorch documentation to learn about the steps for transforming and augmenting images. Try to use different transformations to make your classification performance better.

Note: if you want to normalize your dataset, please use mean=[0.454, 0.426, 0.529] and std=[0.282, 0.269, 0.332] for this dataset. Read the Pytorch documentation to see if it takes in images or tensor types as input.

Section 3. Create your CNN model (20 pts)

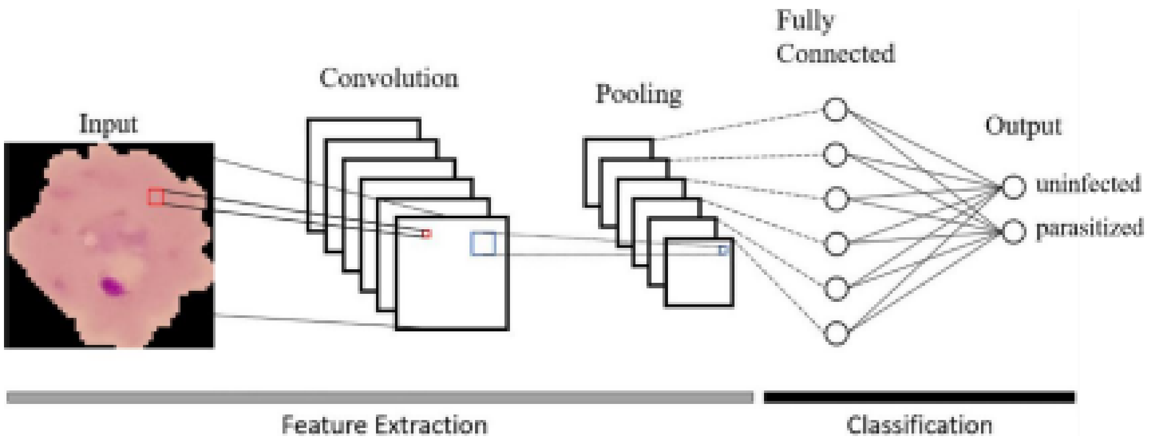
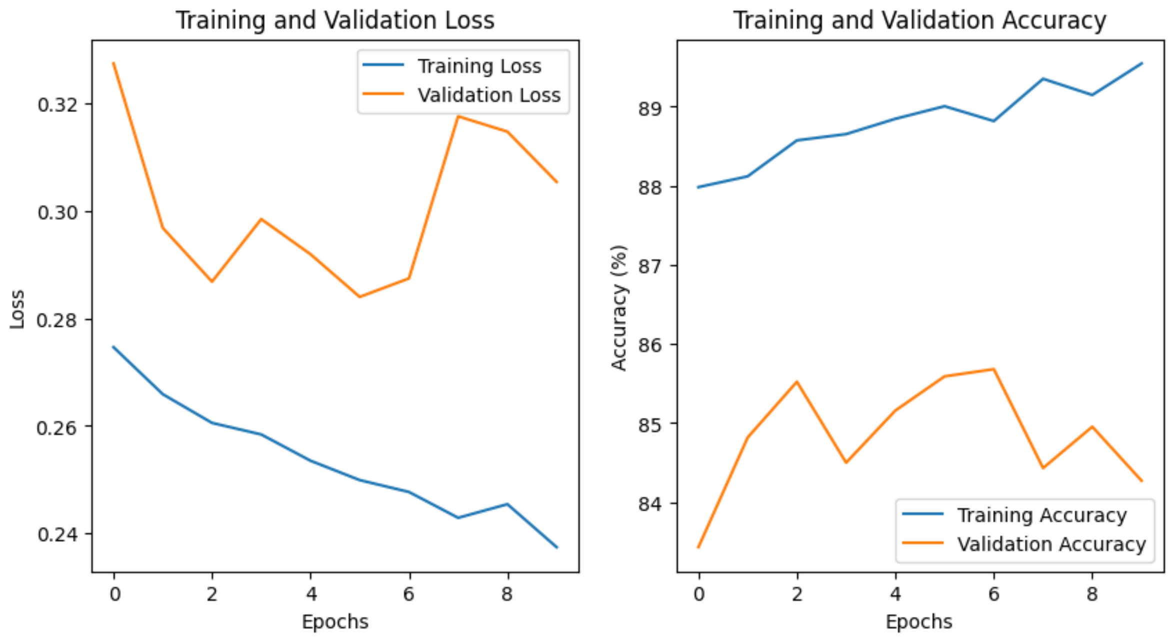


Figure 2 | Example of a CNN model

Set up your CNN model architecture and other hyperparameters (e.g., learning rate, batch size, epoch) that you need. You are free to use any CNN architecture.

Section 4. Train your model (20 pts)

Follow the instructions in the notebook. Make sure you enable a GPU in your notebook. Track your loss and accuracy score on validation data to ensure your CNN is trained properly. Plot your training loss and the validation loss in a figure. Plot your training accuracy and validation accuracy in a figure.



Section 5. Evaluate your model (20 pts)

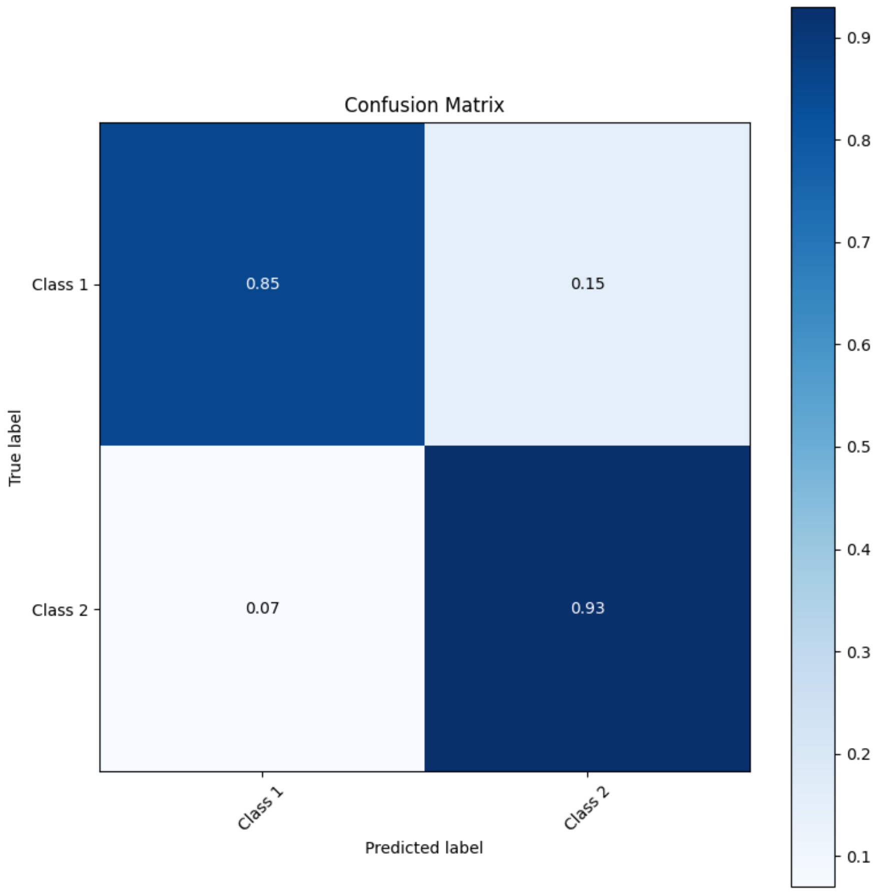
Inference is the ability of a machine learning system to make predictions from data. Now, you have trained your model, and you can use it to make predictions on unseen test data. Create a test dataset and pass it into the test dataloader as you did for the training and validation data. Use your trained CNN to generate the predicted labels for your testing data. Compare the predicted labels with the true labels. What is the accuracy score? Provide a confusion matrix of your prediction result.

Note: think about what data transformations you need to use when creating the test dataset.

The accuracy score is 89%

Confusion Matrix:

```
[[3044  527]
 [ 250 3322]]
```



Section 6. Improve the classification score of your model by tuning hyperparameters (e.g., the size and number of kernels used in the convolution layers, the number of total layers, learning rate, optimizer, loss function, epoch) (10 pts)

Try to optimize your CNN model to reach a classification accuracy on test data higher than 70%. No need to show your work as we will only focus on your accuracy in this section.