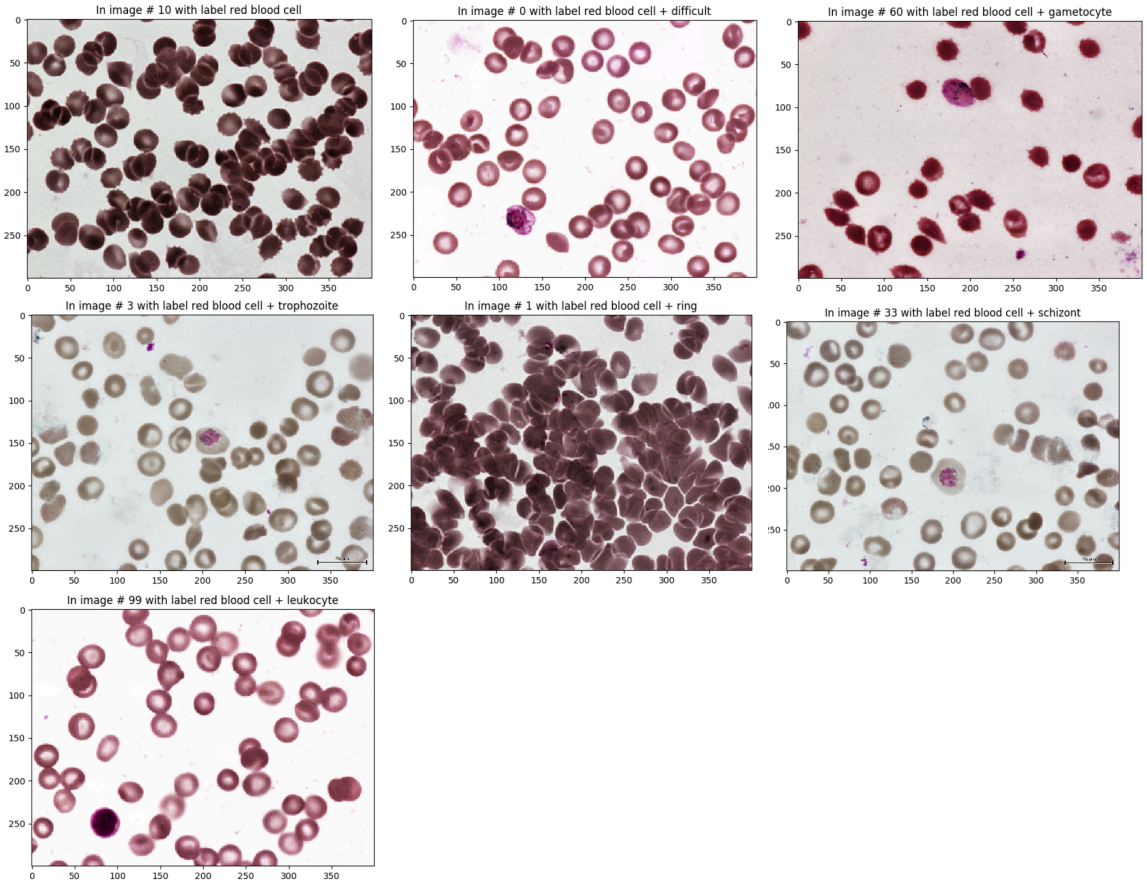
Exam2 Report

Igor

Pre - Day 1:

I tried to read these images and have a general view of them. I found all these images have a label “red blood cell”. Here I print 7 images with each type of labels. I also resize them from the original size 1600x1200 to 400x300 for computational efficiency.



These images gave me a hint that the color may not affect the target labels. Besides, the backgound colors of different images are also different. I’ve got an idea that creating a threshold (by cv2) of these images and generating the equivalent images with grayscale and zero backgroud color.

After loaded the image data, I focused on the model part. Referred to the code Conv_Mnist_gpu.py by Prof. Amir, I constructed my first model with CNN.

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 16, kernel_size=5, padding=2),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.layer2 = nn.Sequential(
            nn.Conv2d(16, 32, kernel_size=5, padding=2),
```

```
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.fc1 = nn.Linear(75 * 100 * 32, 256)
        self.fc2 = nn.Linear(256, 7)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(out.size(0), -1)
        out = F.relu(self.fc1(out))
        out = self.fc2(out)
        out = torch.sigmoid(out)
        return out_
```

In my test data, this model performs really bad, which can only predict [1, 0, 0, 0, 0, 0, 0] for each input. I left its performance later and submitted it to check my prediction function work or not.
Then I saved this model as pkl format and loaded it in my prediction function. In my prediction function, I also pasted the class of model (as shown above) for loading the model parameters.

In Day1 result, I've got an error with the "use .pt format".

Day2:
I changed all the format ".pkl" to ".pt". Both can work in my check_predict_function.py.
Question: what's the difference between them?

In Day1, I added `out = torch.sigmoid(out)` to get the integer output like [1, 0, 0, 0, 0, 0, 0]. However, the outcome becomes bad if I add a sigmoid on the output layer. It is also strange to add an activation on the output layer.

```
def forward(self, x):
    out = self.layer1(x)
    out = self.layer2(out)
    out = out.view(out.size(0), -1)
    out = F.relu(self.fc1(out))
    out = self.fc2(out)
    out = torch.sigmoid(out)
    return out
```

Thus, I use a simple way to get the integer output. If the output is positive, the prediction is 1. Otherwise, 0.

```
return (y_pred>0).float()
```

Then, the result in test data is a little better than Day1 result.

In Day2 result, I've got a loss of 5.649, which ranks 6th in the dailyboard.

Day3:
I've tried grayscale with threshold, which makes the background value = 0. However, the result is not so good as before. Thus, I recover the changes.

I tried to make some data augmentation by `torchvision.transforms`. The improvement is not obvious, but I still keep the augmentation.

Then I tried to modify the architecture of the network. I changed the structure from “conv-BN-pooling” to “conv-conv-BN-pooling-Dropout”. After that, the loss was lowered nearly 1.0.

In Day3 result, loss is 4.768.

Day4:
I’ve still worked on the architecture of the network. Increasing the depth of the network, both conv layers and fully connected layers. Besides, I lowered the learning rate to 1e-4.

In Day4 result, loss is 4.649. Not an obvious improvement.

Day6:
Due to changing the structure of my network did not improve so much in Day4, I turn to the pretrained network. Here I chose the VGG16 as my pretrained model. In my test set, VGG16’s performance is quite close to my model. By the way, I reduced my previous model to only 30MB while the VGG16 model is very large, over 500MB. To validate the performance of VGG16, I still submit this model in Day6.

In Day6 result, loss is 4.392.

Day7:
From the improvement in previous day, I still tuned the VGG16 model. I tried to add more data augmented transforms and run the model with longer epochs.

In Day7 result, loss is 4.25.