Multi-label classification using CNN

Anwesha Tomar
MS Data Science
George Washington University
atomar19@gwu.edu

1. Introduction:

In this Exam we used Pytorch to build a CNN model to identify the seven different data cells, namely: "red blood cell", "difficult", "gametocyte", "trophozoite", "ring", "schizont", "leukocyte". My best model on the seventh day resulted in a 10.31149292 loss.

2. Steps and Thought process:

As we are not using Pytorch inbuilt dataset we have to find a way to download, preprocess and load the data onto GPU.

```
RESIZE_TO = 400
x, y, labels = [], [],[]
for path in [f for f in os.listdir(DATA_DIR) if f[-4:] == ".png"]:
   img_file=(cv2.resize(cv2.imread(DATA_DIR + path), (RESIZE_TO, RESIZE_TO)))
   img_file = img_file.transpose((2, 0, 1))
   img = img_file.astype("float") / 255.0
   x.append(torch.from_numpy(np.asarray(img)))
```

I have made use of the data load function used in the previous exam, the only change I made is to transpose the image file so that the array is in the order (channel, width, height).

Next I One- Hot encoded the data:

```
with open(DATA_DIR + path[:-4] + ".txt", "r") as s:
   label = [line.strip() for line in s]
   vals_enc=encod(labels)
   #print(vals_enc)
y.append(torch.from_numpy(np.asarray(vals_enc)))
```

For the first six days, I had the wrong function, it returned only zeros, instead of actually label encoding the values. On the seventh day I was able to recognize and fix this error and this decreased my loss function from 67 to 10.

One-Hot encoding function:

```
def encod(targets):
   #ecoding the target variables
   test=["red blood cell", "difficult", "gametocyte", "trophozoite","ring",
"schizont", "leukocyte"]
   enc=np.zeros(7)
   for i in range(len(test)):
       for j in range(len(targets)):
           print(test[i],targets[j])
           if (test[i]==targets[j]):
               enc[i]=1
   return enc
```

3. Model:

My model has not changed over the seven days, I have four sets of convolution and pooling layers and three linear layers in order to flatten the data. I resized the data to 400 instead of 50 to try and get more accurate results. The layers were calculated based on the input parameters, which is an image of channel 3, and height and width of 400.

```
class CNN(nn.Module):
   def __init__(self):
       super(CNN, self).__init__()
       self.conv1 = nn.Conv2d(3, 18, (3, 3))
       self.pool1 = nn.MaxPool2d((2, 2))
       self.conv2 = nn.Conv2d(18, 36, (3, 3))
       self.pool2 = nn.MaxPool2d((2, 2),
                                 ceil_mode=False)
       self.conv3 = nn.Conv2d(36, 72, (3, 3))
       self.pool3 = nn.MaxPool2d((2, 2))
       self.conv4 = nn.Conv2d(72, 72, (3, 3))
       self.pool4 = nn.MaxPool2d((2, 2))
       self.linear1 = nn.Linear(72 * 23 * 23, 128)
       self.linear2 = nn.Linear(128, 64)
       self.linear3 = nn.Linear(64, 7)
       self.drop1 = nn.Dropout(DROPOUT)
       self.drop2 = nn.Dropout(DROPOUT)
       self.drop3 = nn.Dropout(DROPOUT)
       self.sigmoid = nn.Sigmoid()
       self.act = torch.relu

   def forward(self, x):
       x = self.drop1(self.pool1(self.act(self.conv1(x))))
       x = self.drop1(self.pool2(self.act(self.conv2(x))))
       x = self.drop1(self.pool3(self.act(self.conv3(x))))
       x = self.drop1(self.pool4(self.act(self.conv4(x))))
```

```
        x =
self.linear3(self.drop2(self.act(self.linear2(self.drop2(self.act(self.linear1(
x.view(len(x), -1)))))))))
        return self.sigmoid(x)
```

4. **Hyper parameters:**

```
LR = 0.001
N_EPOCHS = 20
BATCH_SIZE = 30
DROPOUT = 0.05
```

I noticed that a very low learning rate meant that the model did not learn the data at all, therefore I lowered the learning rate from 5e-3 to 0.001. I also used a small number of epochs in order to avoid overfitting of data. A small batch size was used to avoid memory issues, and lastly I used a drop out rate of 0.05 to get an improved loss value.

5. **Training the model:**

```
model = CNN().type('torch.DoubleTensor').to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=LR)
criterion = nn.BCELoss()
```

I chose to use an Adam optimizer and BCE loss function for training the model.

6. **Improvements in the model and Future scope:**

I spent a lot of time trying to understand how to input the data. And after understanding that I made a mistake with the label encoder which meant that my model did not train on real samples for six days, only on the seventh day did my model train on the data.

I did not perform any data augmentation, had I been given more time I would try to make use of the JSON files in order to reduce the class imbalance and get better results.