**Assignment 1 report**

**Book and shoes image binary classifier**

1. Preparing dataset: How you made each dataset and how many images each dataset has

The training dataset has 730 photos, including 373 for **book** and 357 for **shoes**. The test dataset has 40 photos, including 20 for **book** and 20 for **shoes**.

The training data was collected from Google Image Search by relevant search terms. Search terms for **book**: books, book, best seller book, a single book, the best book, book cover … Search terms for **shoes:** shoes, pair of shoes, nike shoes, adidas shoes, shoes for man … Almost 800 pictures were collected. After that, the dataset was cleaned manually to filter the bad and or irrelevant pictures. The final dataset has 730 photos.

The test dataset was phone-taken photos of me and my friend’s books and shoes. The photos were from many different books and shoes and intentionally taken with many different orientations because I want my model to be capable of deal with this complexity and truly find the hidden patterns in **book** and **shoes**, which is practical in real world problems.

1. Preprocess: The type of preprocessing you conducted and why.

First, all photos were resized to (40,40) to bring all of them to the same size.

For testing data, I did CenterCrop because the main content of the photo is at the center. For training data, I did RandomCrop to represent different parts of the photos and reveal different patterns in them more clearly. All photo then has size (32,32) to be compatible with the model. (this size also makes the model perform well without heavy overfitting or underfitting). I additionally did RandomFlipHorizontal for train dataset to further improve the variety.

The photos were then converted to tensor for computational purpose, and Normalized with mean 0.5 and std 0.2 to bring the intensity of each pixel to the common range

1. The structure of your model and why you chose that structure:

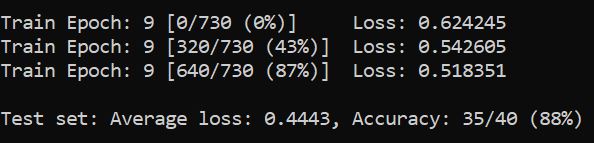
I start build up my model from the model in lab 2, which already has an acceptable performance.

**Main structure:** 3 convolutional (conv) + maxpool layers followed by 2 fully-connected (FC) layers. Adding another convolutional or fully-connected layer lead to a heavy overfitting, and remove a layer reduce the performance. All but the last layer have RELU activations to add nonlinearity.

**Detail: The conv layers** have 32, 64, 128 output channels respectively, all have kernel size of 3, stride 1 (because I do not want to lose many valuable features), zero-padding of width 1 (to preserve the size of feature maps). All **maxpool layers** have kernel 2 and stride 2 to reduce the feature map size by 4 while keeping important information. The output was then flattened out. Flattened output was passed to a RELU and fed to the **FC** layers. The first hidden layer has 32 nodes and the output layer has 2 nodes. The output was passed through a log softmax (log for numeric stability) to yield the log probability and find the most probable class.

**Regularization:** To reduce overfitting, dropout was used after the CNN and before each FC layer. Batch normalization was used after the hidden FC layers. I mainly put regularization after conv layers and before output layers to preserve the information in these important layers.

1. Your final accuracy and a technical explanation for why the accuracy is the way it is



The result was quite precise, thanks to: datasets of reasonable quantity and quality, carefully-chosen structure to avoid overfitting/underfitting, adequate regularization (dropout, batch normalization, L2-regularization), carefully fine-tuned hyperparameters, especially the annealed learning rate to make the solution converge and the early-stopping at around 10 epoches to prevent overfitting.

The result was not perfect, due to some inherent problem: there are many different orientations and positions, especially for books (it can be open or close, lie or stand on the table, …), the natural different of phone-taken and Google photos, the complexity of the model leads to overfitting …