Warm Up Exercises 1, 2

Rishitosh Kumar Singh
Arizona State University
rksing18@asu.edu

Abstract

This report is part of the assignment that is due on 21st Jan 2024. This report consists of two practice problems and by completing these problems I got to know how datasets and dataloaders work in Pytorch, and how to train and test a neural network with Pytorch.

1. Warm Up Exercise 1

1.1. Problem

In this exercise, we were provided with two datasets. The objective was to proficiently load the data using PyTorch's dataloaders and dataset APIs, and subsequently, successfully generate matplotlib plots by writing the images.

1.2. Solution

To load the provided dataset, custom PyTorch dataset classes were crafted. These classes offer flexibility to handle datasets of various types and formats. The torchvision.io.read_image function is employed for reading PNG/JPG images, while the pydicom library is utilized for reading DICOM data.

Transformations

- When PNG/JPG images were loaded, a resize transformation was applied to change the images to a size of 256x256. Figure 1a displays a batch of 4 images from the PNG/JPG dataset.
- After loading DICOM data, a sequence of transformations was employed. At first, the image type is converted from uint16 to uint8, because PyTorch resize function does not support uint16 dtype. Subsequently, the tensor is converted to a PIL image, and the image is resized. Following the dataset loading, all the images are plotted in a matplotlib plot. Figure 1b displays a batch of 4 images from the DICOM dataset.

2. Warm Up Exercise 2

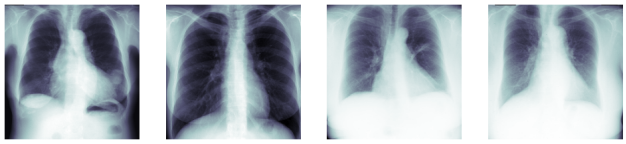
In this introductory practice exercise, chest X-rays were provided, and the task involved training/fine-tuning a model to determine the orientation of these X-rays.

2.1. Dataset

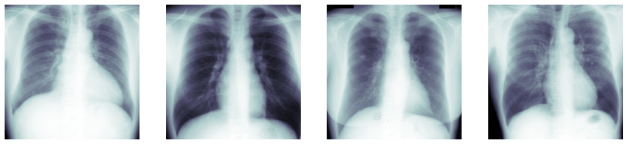
The provided dataset consists of RGB X-ray images with 948 training samples depicting various orientations and an additional 40 samples for testing. All images are of size 128x128 and are oriented either as up (0), right (1), down (2), or left (4). Due to the absence of a dedicated validation set in the given dataset, I have randomly partitioned the training samples into training and validation sets in 8:2 ratio.

2.2. Model Used

For this task, I utilized ResNet18 as the backbone model with pre-trained weights from ImageNet. Since ResNet18 was originally trained on ImageNet, its output tensor has a size of (n, 1000). However, in the given problem, we are specifically dealing with 4 classes (up, right, down, left). Therefore, I introduced a fully connected layer followed by a log-softmax layer to obtain probabilities for each class. Refer to Figure 2 for a graphical representation of the model architecture.



(a) PNG/JPG dataset



(b) DICOM dataset

Figure 1. XRay images batch

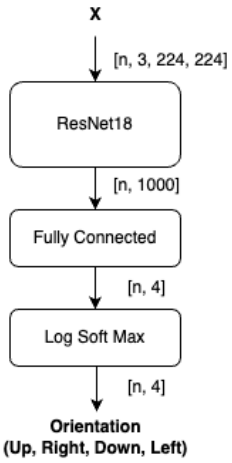


Figure 2. Graph of model used

2.3. Training

The model is trained using the categorical cross-entropy loss function with stochastic gradient descent as the optimizer. As the Log-Softmax layer is already included in the model, the torch.nn.NLLLoss function is employed instead of a torch.nn.CrossEntropyLoss. Four experiments were conducted with learning rates of 0.001, 0.003, 0.01, and 0.03, and the results were averaged. Refer to Figure 3 for a demonstration of the learning curve from one of the experiments.

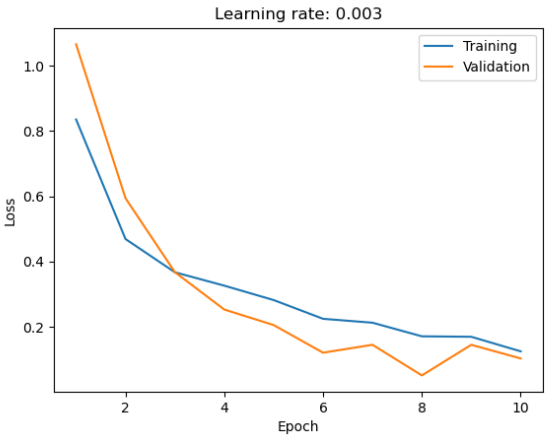


Figure 3. Learning curve

2.4. Results

After fine-tuning the model with the ResNet18 backbone network, the model was transitioned to testing and evaluated on the provided test samples. On average, it achieved an accuracy of 96.87%, and in some experiments, 100% accuracy was also attained. Figure 4 showcases some of the test samples along with their model predictions and target classes.

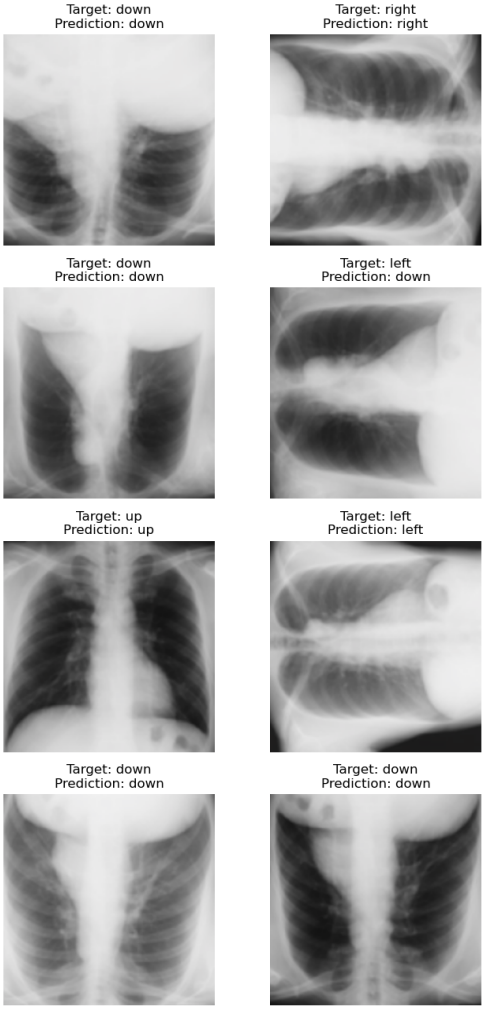


Figure 4. Predictions on few samples of test data