EE-559 Deep learning – Practice 3, Students' questions

A question is denoted by **Q**; the corresponding answer is denoted by **A**. The questions-related exercises are marked by their numbers in Practice_3.pdf and Practice_3.ipynb documents.

Working with data

Q: What is a DataLoader, can I access the dataset directly from the DataLoader, why can't I use counter as in lab 2?
A: "PyTorch provides two data primitives: torch.utils.data.DataLoader and torch.utils.data.Dataset that allow you to use pre-loaded datasets as well as your own data. Dataset stores the samples and their corresponding labels, and DataLoader wraps an iterable around the Dataset to enable easy access to the samples." This information is taken from official PyTorch documentation – check it for more details.

DataLoader as an iterable structure. DataLoader outputs batches of samples and targets sampled from the original dataset until the number of outputted samples is equal to the dataset ($\pm$ one batchsize). You can sample from the dataset with different probabilities and with or without repetitions of samples.

Instead of taking Iterable_loader = iter(dataloader), and then taking next(Iterable_loader) every time you want a new batch, you can iterate through the dataloader with a for loop: for data in dataloader. Here data will be a pair of (input, target), where both input and target are batches of a few samples from the dataset.

Q: Why we set shuffle=True for train_loader?
A: Shuffling the training data ensures that the model does not learn the order of the data, which helps in reducing overfitting and improving generalization. This randomization encourages the model to learn robust features rather than memorizing the sequence in which the data is presented.

Q: For plot_class_distribution_for_dataloader(dataloader) function, why we need to consider the case when class_name is not in count_dict ?
A: We should note that the count_dict is empty initially. Since the dictionary structure does not automatically create keys, we must check if the 'key' is present. If it's not, we need to initialize it with a value before incrementing, ensuring all classes are counted correctly.

Q: Do we need to plot the distribution of classes per mini-batch or per training/test sets?
A: Per training / test sets.

Q: What happens if I change the order of the transformation functions?
A: For this question it is important to account for two factors. Firstly, some transformations in torchvision are deterministic (i.e., transforms.Grayscale), whereas others are not (i.e., transforms.RandomHorizontalFlip).
In general, if we assume that the transformations are deterministic (e.g., when you are working in image editing software), some of the transformations are commutative, whereas others

are not. For example, it does not matter if you convert an image to grayscale and then resize it, or vise versa. However, it would matter if you at first rotated an image, and then applied an affine transformation, or vice versa. More information on transformations is available here.

Model architecture and losses

Q: In exercise 3.3, where can I find the example of how to freeze the feature layers?
A: You won't see it directly on Gnoto, but there is a link embedded in the word "here". Just click on it to access the example.

Q: In exercise 3.3, after loading the AlexNet model, do we need to create new linear layers, or should we modify the existing ones?
A: As indicated in the code comments with `model.classifier[?] = nn.Linear(...)`, you should modify the three existing linear layers of the classifier.

Q: How do we define the architecture of the model (e.g., how did we select 128 as the number of hidden units)?
A: Unfortunately, there is no easy answer on how to select a model architecture. Model architecture could be an output of trial-and-error, or a grid search (which is computationally expensive). Often you would be relying on models' complexity used in the prior work of your domain. Besides this, you could be using models pre-trained on a large corpus of data, hence you would be adopting their architecture, changing only few layers for finetuning.

Q: How does the F1 score work for multi-class classification?
A: For multi-class classification, the F1 score is calculated using a one-vs-all approach, where each class is treated as the positive class while the others are considered negative. This transforms the problem into multiple binary classification tasks, each yielding an F1 score. The final result can be aggregated in different ways, such as micro, macro, or weighted averaging. For more details, refer to the scikit-learn documentation.

Q: What is the difference between the forward pass and the model predictions in exercise 3.5?
A: When you do the forward pass, `out = model(data)`, you obtain logits, that is, the raw, unnormalized scores produced by the model. Since we process a batch of 10 images at a time, `out` has a shape of 10×10, meaning each image is associated with a tensor of 10 values (one per class). The model's predictions, however, refer to selecting the class with the highest logit for each image, giving us the final predicted labels for the given batch.

Q: How to create named modules in the model in PyTorch?
A: Named modules refer to layers or components inside a model with explicitly assigned names. In PyTorch, named modules are typically created by defining a custom class that inherits from `torch.nn.Module`. Each submodule is assigned as an attribute of the main module, and PyTorch automatically registers them. For more details, see this link.

Q: I get an error of dimension mismatch in my model, what's the problem?
A: Dimension mismatch errors in PyTorch usually arise from incorrect tensor shapes being

passed between layers. This could be due to incorrect input size, improper reshaping, or mismatched dimensions between layers (e.g., incompatible input-output sizes in linear layers or convolutions). Checking `model.forward` using `print(tensor.shape)` at various points can help diagnose the issue.