EE-559 Deep learning – Practice 4, 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_4.pdf and Practice_4.ipynb documents.

Text data manipulation and preprocessing

Q: Why the lemmatization of word "explores" is still explores?

A: The reason "explores" is not lemmatized to "explore" is that the default lemmatize() method in WordNetLemmatizer assumes words are nouns unless a part-of-speech (POS) tag is explicitly provided (see the documentation). Additionally, if the word cannot be found in WordNet with the given POS tag, it will return the word unchanged. In this case, since "explores" is not recognized as a noun in WordNet, it is returned as is. However, if you specify the correct POS tag (lemmatizer.lemmatize("explores", pos="v")), "explores" will be considered as a verb and the lemmatization will return "explore".

Model training and evaluation

Q: Why do we use torch.no_grad()?

A: "torch.no_grad() is the context-manager that disables gradient calculation. Disabling gradient calculation is useful for inference when you are sure that you will not be calling Tensor.backward(). It will reduce memory consumption for computations that would otherwise have requires_grad=True." [Pytorch documentation]

This also enables you to have higher computation speed and use larger validation batch sizes [source]. For more information, refer to the following discussions:

1. `https://datascience.stackexchange.com/questions/32651/what-is-the-use-of-torch-no-grad-in-pytorch`,
2. `https://discuss.pytorch.org/t/model-eval-vs-with-torch-no-grad/19615/2`.

It is recommended to use both torch.no_grad() and model.eval() whenever you are not updating the weights of the model. "model.eval() ensures certain modules which behave differently in training vs inference (e.g. Dropout and BatchNorm) are defined appropriately during the forward pass in inference. As such, if your model contains such modules it is essential to enable this. For the reasons above it is good practice to use both (torch.no_grad() and model.eval()) during inference." [source]

Q: What is an epoch?

A: An epoch in the context of training neural networks refers to a complete iteration over the entire training dataset. Usually, an epoch consists of multiple batches. During an epoch, the neural network's weights are updated multiple times using smaller subsets of data, the batches. More information about epochs is available here.

You can also find more about when, why, and the advantages of using batches of data here:

1. `https://medium.com/analytics-vidhya/when-and-why-are-batches-used-in-machine-learning-acda4eb00763`,
2. `https://datascience.stackexchange.com/questions/16807/why-mini-batch-size-is-better-than-one-single-batch-with-all-training-data`,
3. `https://stats.stackexchange.com/questions/49528/batch-gradient-descent-versus-stochastic-gradient-descent/68326#68326`.

Q: Why is my test accuracy higher than train accuracy?
A: This can occur due to several reasons. For example, an "unfortunate" data split or data augmentation (if used excessively) can lead to a training set being harder for the model to learn than the test set. You could experiment with different dataset splits and augmentation and observe the behavior of the model. If the dataset is provided already split into training and testing, it might happen to have a testing set that is easier to predict than the training set. Another possible reason could be due to data contamination, where instances from the training set are also present in the test set, which should be avoided.

Discussions on this topic are available here:

1. `https://stackoverflow.com/questions/51464591/test-accuracy-is-greater-than-train-accuracy-what-to-do`,
2. `https://datascience.stackexchange.com/questions/107834/why-is-my-test-accuracy-higher-than-train-accuracy-sklearn`.

Additionally, note that in the implementation of the practice and marked exercises, the performance on the training set is measured during each batch and then averaged at the end of the epoch. However, with further gradient steps, the model improves, and the accuracy in the first batch of the epoch can be much worse than in the last batch. Therefore, to see the performance of the fully trained model on the train set, you can evaluate the trained model on the training set with .

Q: Why do we need a validation set?
A:Please, refer to the recording of Lecture 3, 12:09 for details about the training, testing and validation sets.

A validation set is needed to assess the performance of a model during training and to tune hyperparameters effectively. The validation set helps us evaluate how well the model generalizes to unseen data and allows us to make adjustments to improve its performance before deploying it on the test set or real-world data. Tuning the hyperparameters or changing your model with respect to the performance on the test is erroneous and will compromise the honesty of the comparison with the other models.

To learn more about the difference between test and validation datasets you can check out this post.

Detailed discussions are also available here:

1. `https://datascience.stackexchange.com/questions/18339/why-use-both-validation-set-and-test-set`,

2. `https://www.reddit.com/r/learnmachinelearning/comments/vogqe5/is_validation_dataset_really_a_must/?rdt=38186`.

Q: Why do we take the gradient with respect to the image in exercises 4.7?
A: You take the gradient of the image because you are trying to attack the pre-trained model by changing the input image. Therefore, the model remains unchanged while only the image is modified. You can see this as a gradient ascent optimisation, where we find the perturbation which would maximise the loss of the fixed model by generating a perturbed image. The gradient of the image is an indication of how changes in the image affect the loss, hence we optimise for the "best" noise, rather than the "best" model.

Q: How does a PyTorch model compute the correct derivatives when `loss.backward()` is called ?

A: PyTorch builds a dynamic computation graph during the forward pass, where each tensor operation is recorded along with its gradients. When `loss.backward()` is called, PyTorch traverses this graph in reverse using the chain rule (backpropagation) to compute derivatives. The `autograd` engine in PyTorch handles this process efficiently, including support for non-trivial cases like in-place operations and detached subgraphs.

Image preprocessing

Q: How are the mean and standard deviation values obtained for image normalization for ImageNet-trained models in Pytorch, such as ResNet?
A: For ImageNet, a pass on the dataset has been made and the mean and standard deviation values have been computed per channel. [source]

Q: Why do we first resize the image to 256 and then center crop to 224 when we prepare the images for the ResNet model?
A: "transforms.Resize() resizes the smallest edge to the given value. Thus, if the image is not square (height != width) Resize(224) would fail to give you an image of size (224, 224)." [source] Moreover, this process eliminates the border, while preserving essential details in the center. A visualization of the effect of these operations is available here. Additional information about image transformations can be found in the Pytorch documentation.

Adversarial attacks

Q: How can I perform a black-box attack if I do not have access to the model and its gradients?
A: Common approaches to craft black-box adversarial attacks rely on gradient estimation methods, substitute models and transferability property, or search-based techniques. Surveys on (black-box) adversarial attacks on computer vision models and defences are available here:

1. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9614158`
2. `https://arxiv.org/pdf/1912.01667.pdf`
3. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9984916`

4. `https://www.sciencedirect.com/science/article/pii/B9780128221099000242`

Q: Can I create an adversarial example by adding noise to just one pixel?
A: In this paper, the authors proposed a black-box attack to generate adversarial examples by perturbing only one pixel.