Handin 3 - Semi-supervised deep learning

Rasmus Freund - 201700273

December 7, 2023

Short answer questions

- Where are we ensuring that the finetuning does not affect the feature extractor?
  - The `Net` class has been created with separate components for feature extraction and classification; which has additionally been split into two layers: `pretrainer` and `generalizer`. When finetuning (`optim.SGD(network.generalizer.parameters(),lr=0.01)`) we only update the `generalizer` parameters, not those of the feature extractor.
- How dose the code work that gets s samples per class for the pre-training dataset
  - The `get_subsampled_dataset` function iterates over each class in the dataset. For each class, it finds the indices where the target label matches the current class and randomly selects k indices from that list which are then appended to the subset.
- What is the `forward_call` parameter responsible for in the `train()` method (located in `network_training.py`)?
  - It performs a forward pass through the network, calculating the loss.
- Describe how the `augment()` method works (located `augmentations.py`).
  - The function iterates through the batch of images, randomly selects another image from the batch (`merge_indices[i]=np.random.choice(...)`). The current image (i) is excluded from the pool, so we're never getting an augment where an image has been augmented with itself. Augmentation is then applied (either mixup or collage), resulting in a new image and an interpolation value.
- If we pre-train and finetune on the same dataset, is there any reason to do the finetuning step?
  - Typically, no. However, there might be specific cases where this could be beneficial. One example could be transfer learning where we could pre-train the model as a simple classification network. After that, the fine-tuning head could then be utilized for more specific tasks. In this task, I would expect a more significant improvement on the EMNIST dataset due to its' complexity, whereas I doubt much can be improved between pre-training and finetuning on the MNIST data.

Predictions

- Will the collage and mixup data augmentations help achieve higher finetune accuracies? Which do you expect will be more effective?
  - I doubt that data augmentation is going to improve accuracy in the case of these data sets. Changing the information in the images through augmentation is also going to change the true "meaning" of those images, since we're dealing with numbers and letters. Of the two types of augmentation, I suspect that mixup might perform better, since the entire structure of each number/letter will be kept in the augmented data.
- What relationship do you expect between the number of samples in the pre-training dataset and the finetuning accuracy? Does this change with data augmentations?
  - Generally, more samples in the pre-training should improve the accuracy of the fine-tuning head. However, given that the data is relatively simple and present clear patterns, a plateau is likely to be reached fairly quickly in terms of amount of training samples being used. Beyond this plateau, increasing the amount of samples will probably not lead to significant increases in accuracy. While data augmentation is, under most circumstances, a good way to increase accuracy when training on sparse data, I suspect it won't have a great impact in this network, mainly due to the reasons listed in the answer to the previous question.

Discussion

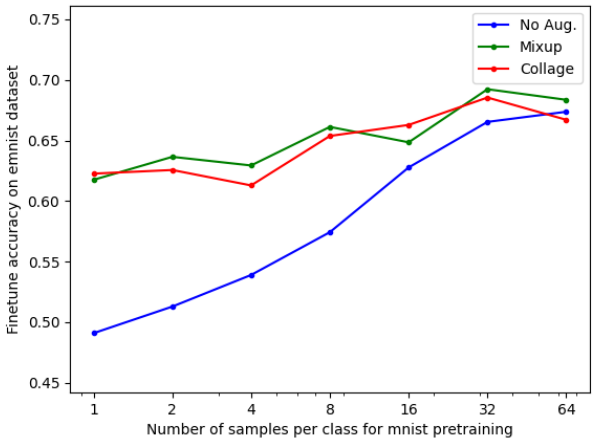

Figure 1: Finetune accuracy of the network on the EMNIST dataset based on number of samples per class when pre-training on the MNIST dataset. Default settings for the network were used when creating this plot.

- How does the number of samples per class affect training performance? Does this get affected by the augmentations?
  - When applying no augmentation, increasing the amount of samples per class always increases accuracy of the model, as seen in 1. However, the difference in accuracy between 32 and 64 samples is relatively minor, indicating that the model is unlikely to get much better with additional samples. Using augmentations improves the performance of the model, especially at low number of samples per class. At a single sample per class, the accuracy is increased from about 50% to about 62% for both types of augmentation. This difference decreases as the number of samples increases, to the point where there is little to no difference between the three options (no augmentation, collage, or mixup). The difference in accuracy based on number of samples is less dramatic on the models that are utilizing augmentation. At 1 sample per class both the mixup and collage models sit at about 62% accuracy, whereas the maximum accuracy, reached at 32 samples per class, is approximately 68%. The performance of augmentation models will be discussed in further detail below.

- Which augmentation performs better? Why?
  - The data reveals that both mixup and collage augmentations provide a substantial boost in accuracy when pre-training with a limited number of samples per class on the MNIST dataset. This improvement is especially notable at the lower end, where having only one sample per class, both augmentation strategies increase the model's accuracy on the EMNIST dataset by around 12 percentage points compared to the non-augmented approach. This improvement indicates that both strategies effectively combat overfitting by generating more diverse training examples from limited data.

    Interestingly, the collage augmentation, initially matching the mixup augmentation, indicates that despite the potential complexity and disjointed nature of the images it creates, it is equally effective as mixup at lower sample sizes. The collage's method of recombining different image parts likely encourages the model to focus on more localized features and to learn from a variety of patterns, which can be beneficial when each class has very few samples.

    The near-identical performance of mixup and collage augmentations throughout the range of sample sizes is quite interesting, as one might expect the more "natural" image blends produced by mixup to consistently outperform the more artificial constructs of collage. However, the data shows that for this task, both augmentation strategies are equally viable in enhancing the model's capacity to generalize from MNIST to EMNIST. This could be due to both methods providing a form of regularization that prevents overfitting, albeit in different ways.

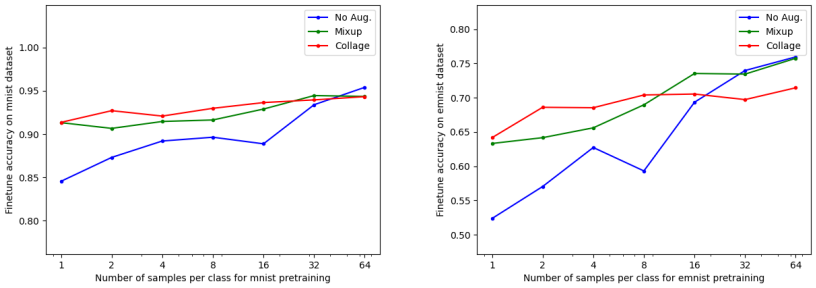

Figure 2: Left: network performance when pre-trained and finetuned on MNIST dataset; Right: network performance when pre-trained and finetuned on EMNIST dataset

- Does finetuning and pre-training on the same dataset obtain better performance than pre-training on one dataset and finetuning on another? Why?

  – As seen on the left plot of 2, pre-training and finetuning on the MNIST dataset leads to a high accuracy of approximately 95% regardless of the augmentation technique used or not used. The right plot shows network performance when pre-training and finetuning on EMNIST. As perhaps one might expect, the accuracy is generally lower than for MNIST due to the inhently more complex nature of the EMNIST dataset.

    Since the network pre-trained on MNIST and finetuned on EMNIST has worse performance than either of the networks presented here, we can conclude that, while some features are certainly transferable between the two sets, it appears that some features of the EMNIST data set must be learned through pre-training.

    In conclusion, the network performs best when pre-training and fine-tuning on MNIST due to its simplicity and the network's ability to specialize in the specific features of the digits. Performance drops when working with EMNIST due to its complexity and further drops when transferring from MNIST to EMNIST due to the additional adaptation required by the network to handle a new set of characters.

If would've like to include a section on pre-training on EMNIST and finetuning on MNIST to compare, but unfortunately neither time or character limit allows for this.