HPML Algorithm Checkpoint 1

Introduction

For my first algorithm I wanted to implement I initially wanted to implement a transformer based MoE model using auxiliary loss free load balancing that would do text classification but after working on that problem for a bit I decided it was too boring, there's nothing exciting for me about classifying text. So, I moved onto classifying images and for this first implementation converted Alexnet into an MoE model and used the CIFAR-10 dataset to train it.

Data Loading

The dataloading mechanism loads CIFAR-10 by first defining separate transformation pipelines for training/validation and testing, where images are resized to 227 x 227, converted to tensors, and normalized using the dataset's specific means and standard deviations. For the training and validation sets, it loads the CIFAR-10 training data twice—once with data augmentation (if enabled) and once without—and then splits the dataset into training and validation subsets based on a defined fraction, using a fixed random seed for reproducibility and SubsetRandomSampler to randomly select indices. Finally, it creates DataLoaders for each subset so that during model training and evaluation, data is efficiently loaded in batches, while a separate DataLoader is set up for the test set.

AlexNet

The regular AlexNet model is structured as a series of five convolutional blocks folloId by three fully connected layers. Each convolutional block starts with a convolutional layer that uses progressively smaller kernel sizes and appropriate padding to extract spatial features, immediately folloId by batch normalization to stabilize the activations and a ReLU nonlinearity to introduce non-linearities. Max pooling is applied in blocks 1, 2, and 5 to downsample the feature maps and reduce spatial dimensions. After these convolutional layers, the output is reshaped (flattened) into a one-dimensional vector, which is then fed into two fully connected layers that use dropout for regularization and ReLU activations to refine the high-level features, before the final linear layer outputs the class scores for classification.

Mixture Of Experts AlexNet

In MoE AlexNet, once the convolutional and fully connected layers extract a high-level feature representation, the standard classifier is replaced by a Mixture-of-Experts block. Here, a gating network processes the feature vector and outputs a set of affinity scores—one per expert—after which an expert-specific bias is added before selecting the top expert (via argmax) to generate the final prediction. This design allows each expert—a separate linear classifier—to specialize on different subsets of the classification task, thereby dramatically increasing model capacity without a proportional increase in computational cost, since only one expert is active per input. To prevent the gating network from converging to the same few experts (i.e. routing collapse), an auxiliary-loss-free load balancing strategy is employed: rather than adding an explicit loss term that can introduce interfering gradients, each expert has a dynamically updated bias term that is adjusted at the end of each batch—decreased if the expert is overselected and increased if underselected—ensuring a balanced utilization of all experts.

HyperParameters

I set num_classes to 10, indicating that our model is designed to classify input images into 10 distinct categories (such as in CIFAR-10). I train the model for 20 epochs, meaning the entire training set is passed through the network 20 times, and a batch_size of 64 is used so that 64 examples are processed simultaneously per iteration. The learning_rate is set to 0.005, which determines the step size for parameter updates during training. The AlexNetMoE model (my MoE version of AlexNet) is instantiated with these 10 classes and moved to the appropriate device. For the loss function, I use CrossEntropyLoss, which is Ill-suited for multi-class classification tasks. The optimizer is stochastic gradient descent (SGD) with a learning rate of 0.005, light_decay of 0.005 (acting as a regularizer to prevent overfitting), and momentum of 0.9 to help smooth updates and accelerate convergence. Finally, total_step is defined as the length of the training DataLoader, representing the total number of iterations per epoch. And for my MoE model I used 3 experts.

Evaluation

I evaluated the performance of the model using around 20% of the images in the CIFAR10 dataset which is 10000 images. Where the AlexNet MoE model has a 78.12% accuracy and the regular Alexnet model has around an 82.27% accuracy. So AlexNet MoE had around a 4% drop in accuracy, possibly due to the small dataset I used, due to splitting the data across 3 experts the experts did not get to see enough data to be properly trained, to fix this I plan on switching to CIPHAR100 or Imagenet for training for future checkpoints. However, the MoE model did achieve a semi-decent speedup of 20% over the regular AlexNet model on inference. I used torch functions to parallelize the model achieving meager speedups show in this table and graph on inference averaged over 100 runs:

	MoE 1 thread	MoE 4 threads	MoE 8 threads
Runtime	.3671 Seconds	.3284 Seconds	.3198 Seconds
Speedup	1	1.117	1.147