MPCS 51087
Problem Set 3
Machine Learning for Image Classification
Final Specification

Winter 2024

due: Saturday March 9 @9am

1 Model Updates

For the final specification you will improve the performance of your GPU naive implementation by using CUDA's implementation of BLAS (CuBLAS). You will then provide a comprehensive summary of your best performance (not accuracy) results for a given problem specification using Model 2, similar to Milestone 2.

Since weights and batch inputs are chosen randomly, two correct codes will in general not give the exact same accuracy for otherwise identical parameter values. This has complicated drawing conclusions about whether big differences in accuracy between submissions indicate bugs. To minimize this we want to first make sure other aspects of the benchmark simulations identical across submissions. Even if you can get higher accuracy with a different parameter choice, the final benchmarks will be done as follows for everyone:

- Normalize the pixel weights to range from 0 to 1.
- Use single precision.
- Use the Kaiming initialization scheme, as in Milestone 2 (a few of you didn't): $w = \mathcal{N}(0, 2/n)$ – i.e. a normal distribution centered at zero with variance $2/n$. If you aren't using C++ and want to sample from a Guassian yourself, Box Muller is a very easy technique, giving two independent numbers from $\mathcal{N}(0, 1)$ using just $U(0, 1)$ (rand) and triginometric functions:

```
U1 = rand;
U2 = rand;
Z1 = sqrt( -2*log(U1) ) * cos(2*pi*U2);
Z2 = sqrt( -2*log(U1) ) * sin(2*pi*U2);
```

- Use $\alpha = .1$
- Use a batch size $nb = 200$
- Run 50 epochs

Furthermore, the higher accuracy parameter choices are often worse rather than better, in the sense that they overfit the data – ie are biased toward the particular choice of training set. To explore this we'll change the workflow a bit, dividing the 60K training images into a set of 50K training inputs and 10K *validation* inputs. After each epoch using the (now) 50K training images, you will calculate the loss based on the validation set, but only use the 50K training images to update the weights (ie the validation set is not used

in the weight calculation). You are required to show a plot of this loss curve – epoch on the x axis, loss on the y-axis. The rule of thumb is that when the loss stops decreasing, you have a good unbiasied fit to the data. To make this process uniform, everyone will use the first 50K for training and the last 10K for validation.

Finally, you should have a sense of how much randomness effects your accuracy. There is no formal requirement for this, but you should run your simulation multiple times (changing the prng seed) and get a sense of how much the accuracy fluctuates from execution to execution. Knowing this helps to identify when differences in accuracy between two codes likely signal a bug, or are within the normal range of random fluctuations. In your write up, report briefly and informally on this sensitivity analysis.

Finally, finalize your performance table with the data in the table below.

Version	**Processor**	**Accuracy**	**Grind rate**	**Training Time**	**TPB/cores**
GPU native					
GPU CuBLAS					
CPU native					
CPU BLAS					

Table 1: Fill out Table 1 with your best performance for the model problem. Be sure to only use the parameters specified above for these runs: learning rate $\alpha = 0.1$, batch size $nb = 200$, $epochs = 50$, using $50K$ training samples and $10K$ validations samples.

In Summary, your report will include:

- Your full name and CNET ID
- A discussion of any shortcomings in your codes, anything you couldn't finish, any anamolous behavior. You can also include a short summary of what you did as a reminder. Also please mention your observations on the sensitivity of the code to random inputs, as mentioned above, and finall the particular implementation of BLAS used.
- Loss curves for each of the four benchmark problems
- Table 1, filled out with as much as you were able to complete.

Finally, as always please include a Makefile and README with instructions on how to build/run your code, and remove extraneous files from your repo.