HW 06: k-NN Accelerator

Due: December 21st 23:59pm

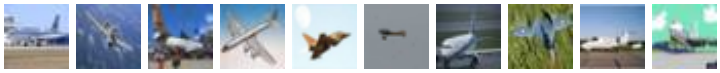
Goal

In this assignment, you will implement k-nearest neighbors algorithm (k-NN) in C and Verilog on PicoRV32. The implementation details are the same as the previous assignment. But this time, the accelerator also acts as a master to the memory. You will need to read images out of the memory and process them.

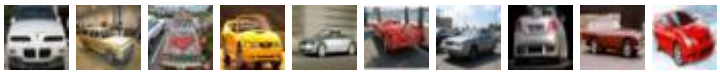
Files Description

data_batch.bin: contains 10000 images from CIFAR-10 dataset. Each image comes with a label from 0 to 9, indicating one of the 10 classes. We only need the first 1000 images for this assignment. Here are the 10 classes in the dataset. For each class, we also list 10 random images for your information:

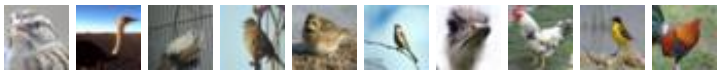
airplane



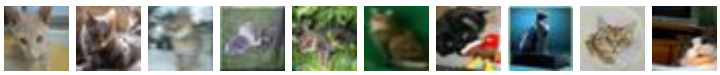
automobile



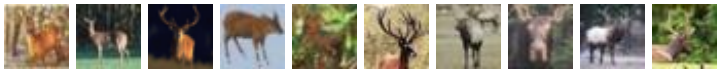
bird



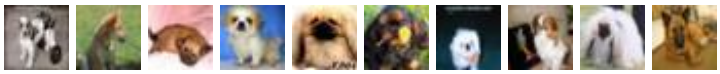
cat



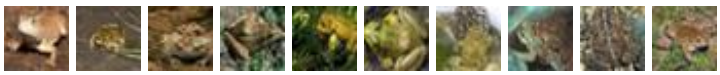
deer



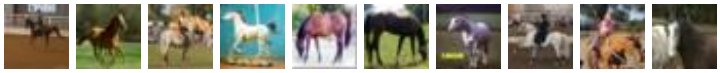
dog



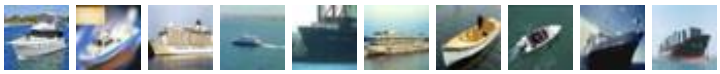
frog



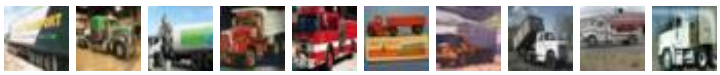
horse



ship



truck



golden.txt: The golden results (for k=5) to verify your designs.

imagegen.v: A tool to generate a bmp file for image #ID (0~999) for visualization. Usage:

```
make image ID=0
```

Problem Description

k-NN: http://cs231n.github.io/classification/#nn

The k-NN algorithm could be summarized as follow:

1. Use the first 50 images as the test images, and the rest 950 images as the training images;
2. For each test image, compute its distances to all the training images;
3. Find the *k* closest images (with the least distance);
4. Have top *k* closest images vote on the labels; Break tie by choosing the smallest class ID.

Memory Interface

The native memory interface of PicoRV32 is a simple valid-ready interface that can run one memory transfer at a time. Our accelerators and CPU could access the memory at the same time (similar to a **dual-port** memory). Furthermore, the Read-Cycle timing is just like the SRAM we used in **the FIFO Homework**.

The 1000 images are stored in the memory from 0x00010000 in row-major order. An image row contains a label following by 3072 pixels (32x32x3). Each label or pixel is a 32-bit value in the memory. Refer to CIFAR-10 website for further details.

Working Items

1. Study the software version of k-NN and complete it for performance comparison. Compute the square of L2 (Euclidean) distance, i.e., $(L2)^2$. (Avoiding the complicated square root operation.)
2. Implement the hardware version of k-NN with k=5; Use either PCPI or MMAP method.
3. You don't need to put everything into the hardware part. Instead, you can partition the algorithm into several steps (e.g. distance computation, sorting, etc.) And let the software control when and which step to execute.
4. Change the distance metric and the value of k. Try to achieve a better accuracy than the default setting. (We only consider a small subset of the images, so you may expect for a relatively low accuracy.)

Questions & Discussion

1. How do you partition your algorithm?
2. Profile both software and hardware versions. Which part do you think is the bottleneck of your (software/hardware) design?
3. Briefly describe the experiments you did and their results.