**Critical Sections of Sequential CNN**

CNNs are very useful in solving problems that are hard to describe and solve algorithmically. The only problem with them is the amount of time they can require to run. Due to the large number of internal layers and nodes, the sequential code is not scalable to large sample sets. To understand what parts of the sequential code needed to be parallelized, I decided to profile the sequential code using Intel's processor performance analysis tool called VTune. Through this, I was able to find the hotspots that are taking up the most time during the execution and then target those sections in the parallel implementation. Figure 3 below shows the results of profiling the sequential code. A staggering 97.3% of the runtime is spent within the conv\_forward\_valid function. This means that the convolution layer is clearly the bottleneck when it comes to the execution speed. This means that if the runtime of the convolution layer does not decrease, then there won't be able to see much of a speed up. Due to this, the focus of the base parallel implementation is to parallelize the conv\_forward\_valid function.

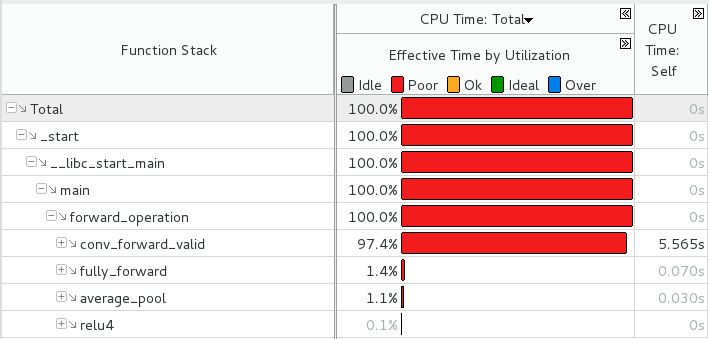


Figure 1: Top-down runtime tree of sequential code

Another important realization that came out of profiling the sequential code is the division of work between the layers. Figure 4 below clearly shows that the second convolutional layer does the most amount of work (68.8%). Based on this, the kernel design needs to be highly work-efficient that will work well with variable input sizes.

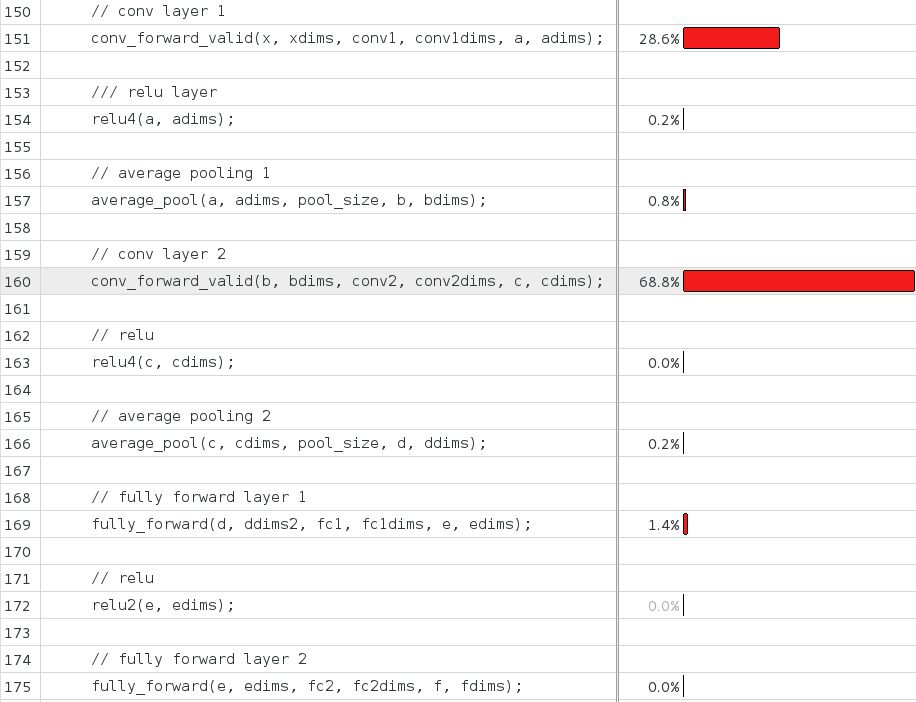


Figure 2: Division of work within the forward\_operation function.