# Multicore Programming Project

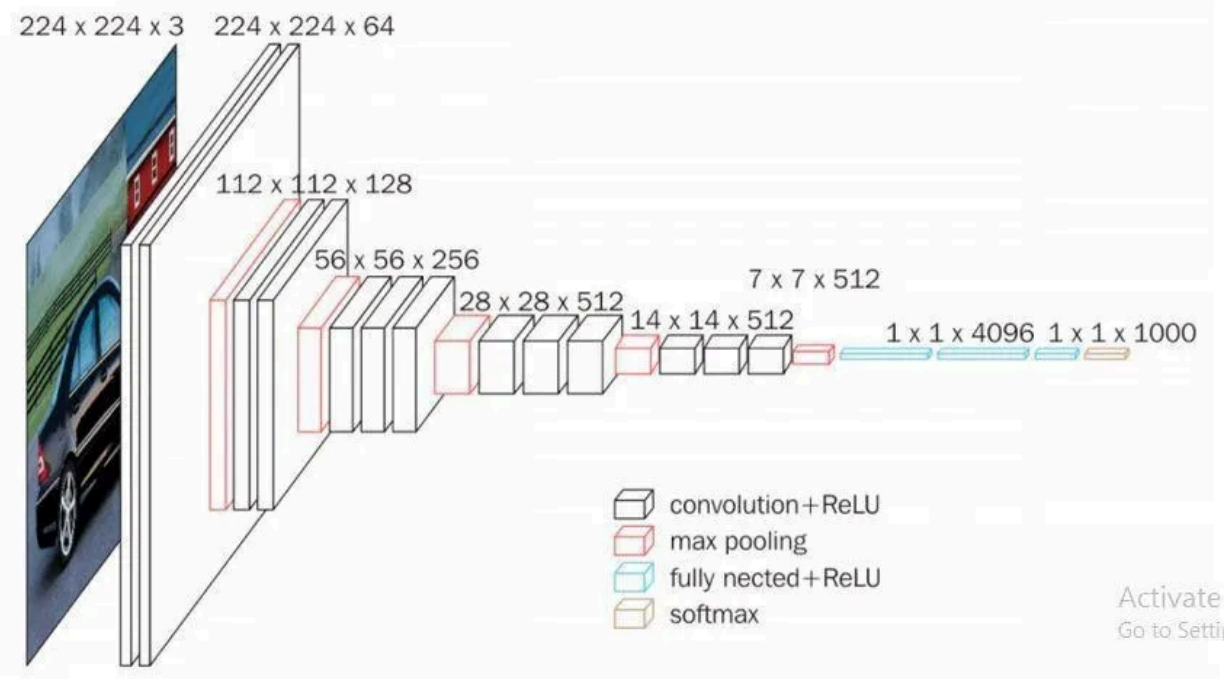
## Amirkabir University of Technology

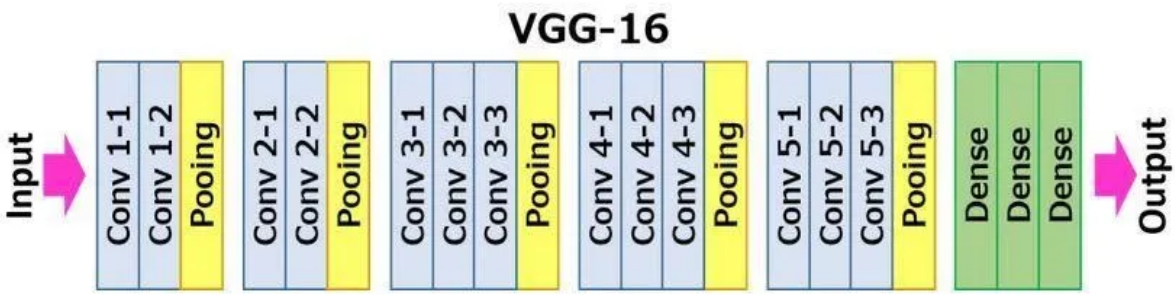
## Winter 2024

## Convolutional Neural Network

Convolutional neural network (CNN) is a regularized type of feed-forward neural network that learns feature engineering by itself via filters optimization. The main idea is that since features are mostly localized in an image, unlike a fully connected network, we only connect adjacent neurons (the ones that are in a convolution filter) and not all neurons. For example, for *each* neuron in a fully-connected layer, 10,000 weights would be required for processing a 100x100 image. However, applying cascaded *convolution* (or cross-correlation) kernels only 25 neurons are required to process 5x5-sized tiles. The result is then passed to a fully connected network to analyze the extracted features.

VGG16 is an object classification network with 16 layers.






Here you can see a project implementing VGG16 using CUDA and OpenMP, the CUDA implementation using cuBLAS and cuDNN. In this project you will, *in groups of 2*, analyze the OpenMP program and use the CUDA project as a base and reimplement the network using simple CUDA kernels.

# Phase 1

Fork simple-vgg16 and analyze it using the various profiles and analyzer programs you know. As we used the Intel VTune profiler in the lab, we will explain the steps you need to take with that program but getting the needed metrics and analysis using any other profiler is acceptable. Here are the steps you should take:

1. Check for data races using Intel Advisor.
2. Using VTune, run performance snapshot analysis.
3. Take note of the memory bound and IPC metrics. What do they tell you about the program's performance?
4. Run hotspot analysis. Take note of the CPU utilization graph and the CPU utilization for the most time consuming area of the code. What do they tell you about the program's performance?
5. Analyze memory access. What kind of memory is the program mostly bound to? Is the DRAM bounded by its bandwidth? What is the LLC miss rate? What do all the metrics in this page tell you about the program's performance?
6. (Bonus) Using the information you got from profiling, improve the code and make it faster.

## Deliverables

You should upload a zip file named GROUPID1_MP_P1.zip containing:

- Screenshots from the profiling and analysis programs as detailed before
- Report containing analysis and profiling of your program and your explanations of the results as detailed before and (Bonus) explanation of how you improved the code
- (Bonus) Your improved code

The deadline for Phase 1 is 1403/3/19.

# Phase 2

Fork the simple-vgg16-cu project and reimplement all functions and data structures that use cuDNN and cuBLAS with simple CUDA kernels and data structures. At the end of Phase 2 you should deliver:

- Reimplantation of the network using simple CUDA kernels
- Thorough explanation of the convolution and matrix multiplication algorithms and how you parallelized them
- Profiling and analysis result of your program using NVIDIA Nsight Compute and NVIDIA Nsight Systems (or NVIDIA Visual Profiler if your GPU doesn't support the new profilers)

Your network's inference time should be within 30% of the base implementation of simple-vgg16-cu.

The minimum required analysis steps for your CUDA program is as follows:

- Using NVIDIA Nsight Compute, profile your kernels.
- In the summary page, take note of all kernel call's duration, compute throughput and memory throughput. Which kernel took the longest time to execute?
- Go to the details page and do the following steps for each of your kernels.
- Look at the GPU Speed Of Light Throughput tab. What is the compute and memory throughput of your kernel? Is it memory or compute bound?
- Look at the roofline graph of your kernel. Where in the graph does your kernel reside? Is there any room for improvement? Does the graph agree with your analysis of compute or memory bound in the last step?
- Look at the memory workload analysis tab. Take note of the memory chart. What is the ratio of shared memory to global memory usage? What are your cache hit-rates? Does the tab give you any warning regarding uncoalesced memory accesses?
- Look at the occupancy tab. What are the theoretical and achieved occupancy rates? Explain any possible discrepancy between the two.

## Deliverables

You should upload a zip file named GROUPID1_MP_P2.zip containing:

- Screenshots from the profiling and analysis programs as detailed before
- Report containing analysis and profiling of your program and your explanations of the results as detailed before and explanation of how your kernels work
- Your CUDA program

The deadline for Phase 2 is 1402/04/09.