An elegant way to map implementation method to layers in a CNN based on constraints

# Problem statement:

Given a CNN, a set of implementation methods for a convolution layer in hardware, we need to choose the optimal sequence of implementation layers such that total running time (layer wise execution time, inter layer context switching time) is at a minimum.

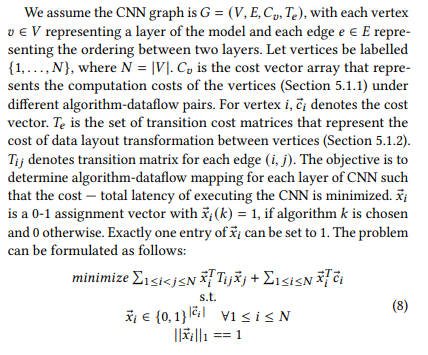


Fig 1: A definition to the same problem statement in the DYNAMAP paper [1]

# An equivalent representation of the problem statement:

Consider an equivalent representation of the same G = (V,E,Cv,Te). Instead of having a small number of V’s and the main complexity of the problem in (E,Cv,Te), Gs = (Vs,Es) where a vertex vij signifies the cost of implementing a layer i using method j and edge eij : vi1j1 -> vi2j2 signifies the cost to context switch from layer i1 implemented using method j1 to layer i2 implemented in method j2.

Ex:

Consider a CNN with 4 layers and 2 ways to implement each layer.

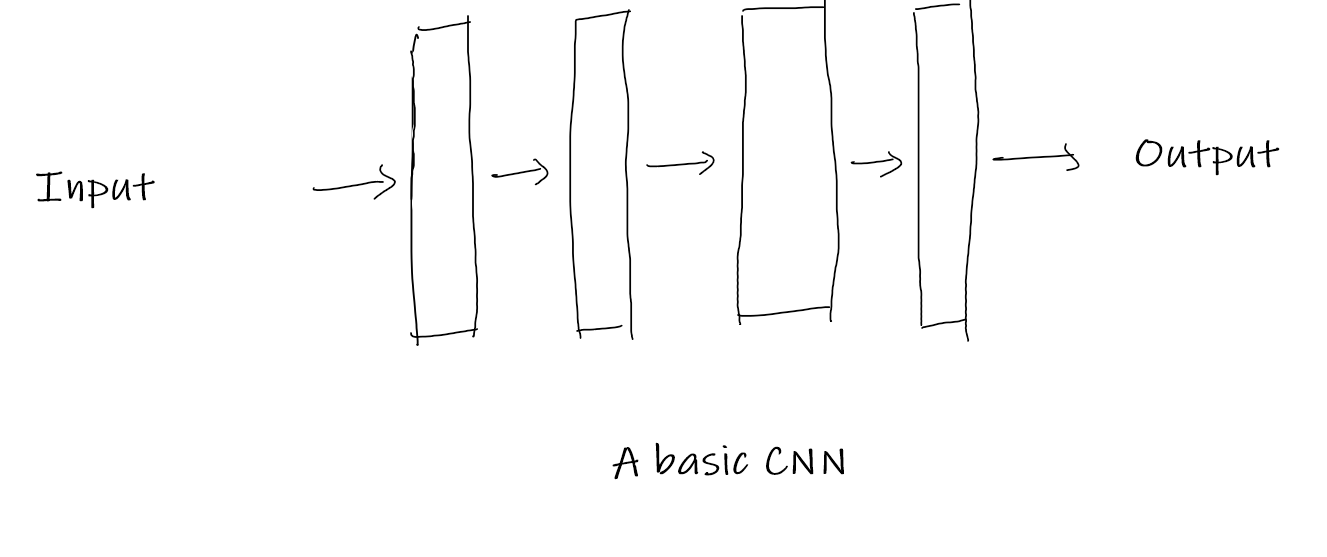
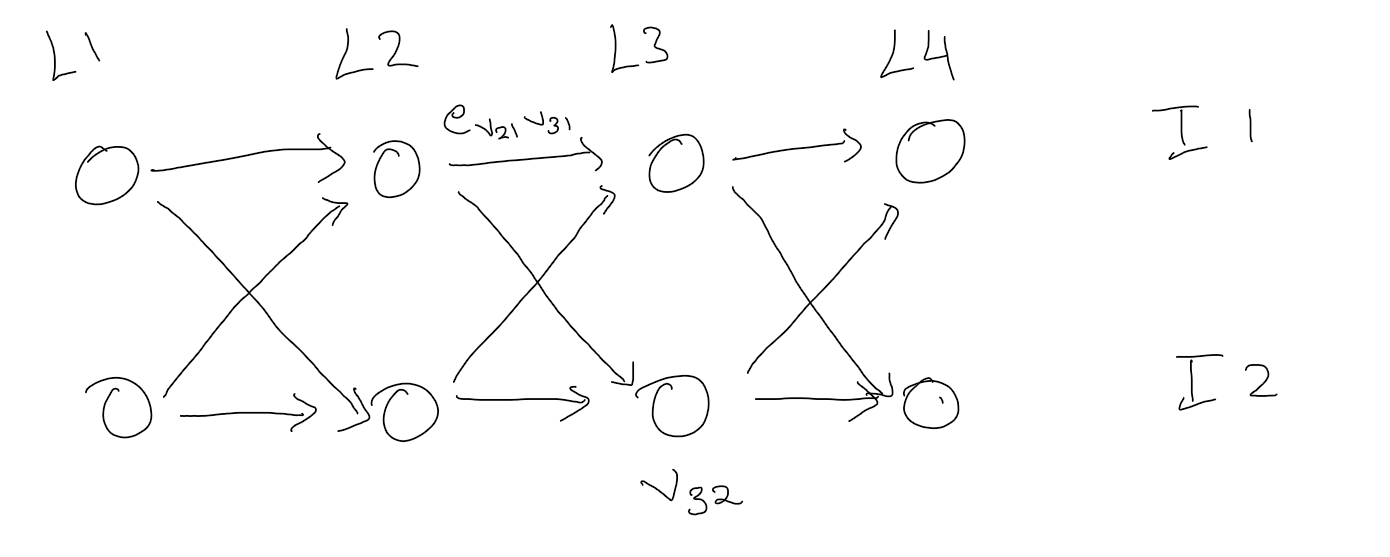


Fig 2: A CNN with 4 layers

Then the graph representing all of the specifications is:

Fig 3: Graph representing all the specifications of the problem statement

Here V32 holds the cost of implementing layer 3 with method 2 and ev21v31 signifies the cost(time) to context switch b/w v21 & v31.

# Some key insights:

1. The graph is a DAG
2. The vertices involved in the shortest path among set of all paths going from a vertex in layer 1 to a vertex in the final layer is the optimal sequence.

# Seamless way to tackle the 2nd key insight:

We can have a source vertex s and a destination vertex t such that s has only outgoing edges to vertices in L1 with cost 0 and t has only incoming edges from Ln with cost 0.

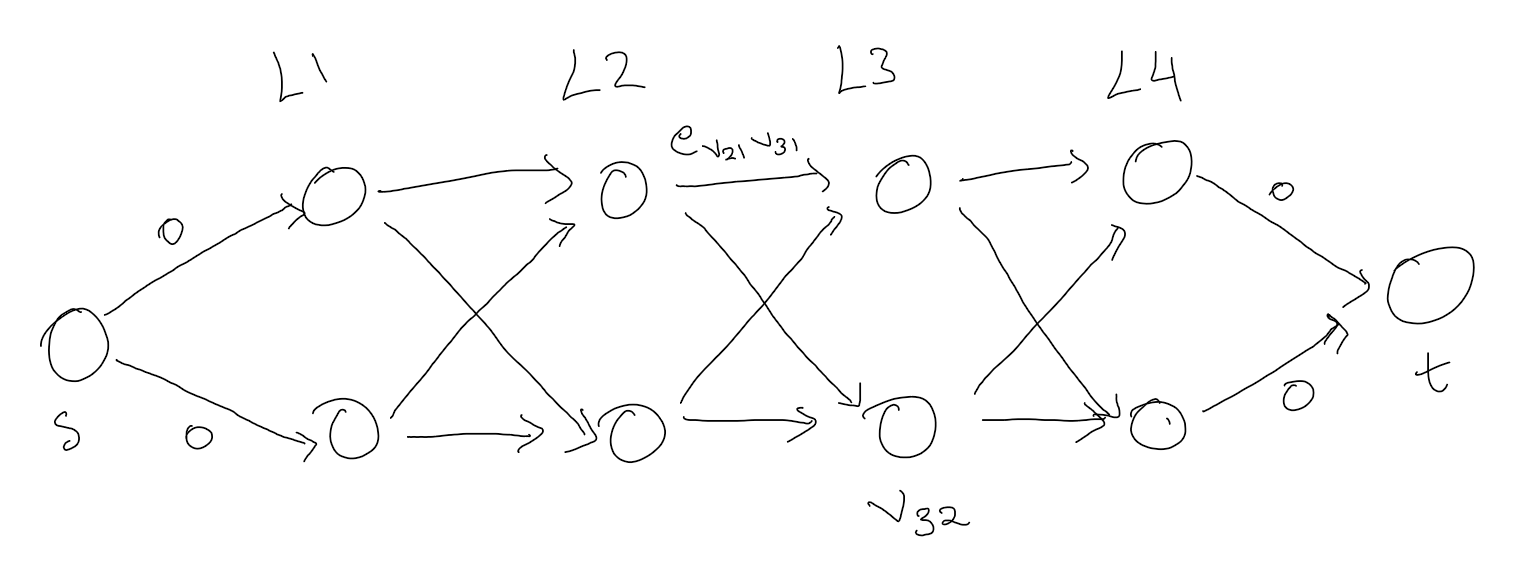


Fig 4: Augmented graph

**Then the 2nd insight just reduces to the shortest path between s and t**.

Since our graph is a DAG(1st insight) with every path from source vertex to a given vertex having same length we can solve this problem linearly (using BFS or dynamic programming).

Hence the running time of our algorithm is O(m+n) where m is the num of edges and n is the number of vertices in the graph.

But m = (N-1)(K^2)+2K

and n = KN where N is the num of layers and K is the number of possible implementation choices for each layer.

**Hence T(N,K) = O(N\*K^2)**

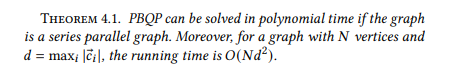


Fig: 5: The same running time is also obtained in the DYNAMAP paper [1]

Reference:

[1] Meng, Y., Kuppannagari, S., Kannan, R. and Prasanna, V., 2021, February. Dynamap: Dynamic algorithm mapping framework for low latency cnn inference. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 183-193).