# Signal Processing System Design Laboratory (EE69205)

**LAB REPORT**

by


## YASH ANAND

(Roll No.- **23EE65R22**)




**SIGNAL PROCESSING AND MACHINE LEARNING**

**Department of ELECTRICAL ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR**

**Kharagpur - 721302, India**


September, 2023

# || Experiment 2 ||
# || Parallelization for execution speedup ||

## 2.1 2D and nD Convolution

**AIM:**
The 2D and nD Convolution using nested loop and parallel loops.Code profiling for execution complexity comparison.

**FUNCTION USED:**

- **with concurrent.futures.ThreadPoolExecutor() as executor:** This statement creates a ThreadPoolExecutor object that provides a simple way to use threads for concurrent execution of tasks. It's part of the concurrent.futures module, which provides a high-level interface for asynchronously executing functions.
- **np.empty():** The np.empty() function is used in the function linear_convolution() to create an empty array of the specified output size. This array will be used to store the convolved signal.
- **filter_array = np.random.randint(0, 3, size=filter_size):** This creates a random filter array of size filter_size
- **perform_multidimensional_convolution(signal, filter_array, dim):** This calls the perform_multidimensional_convolution() function to perform the convolution.

**OBSERVATION:**

★ The execution time of the nested loop increases as the convolution dimension increases. This is because the nested loop has to iterate over more pixels in the output array for each dimension.

★ The execution time of the parallel loop is initially higher than the nested loop, but it decreases as the convolution dimension increases. This is because the parallel loop has to start up the threads and distribute the work among them. However, once the threads are started up, they can work in parallel, which can significantly reduce the execution time.

★ The parallel loop is faster than the nested loop for convolution dimensions of 10 or higher. This is because the overhead of starting up the threads is outweighed by the time saved by working in parallel.

**Here is a table that summarizes the time complexity of Nested and Parallel Loops in nD Convolution:**

| Method | Time Complexity | |
|---|---|---|
| | Nested Loop | Parallel Loop |
| 1D Convolution | $O(n^2)$ | $O(n^2)$ |
| 2D Convolution | $O(n^4)$ | $O(n^2)$ |
| 3D Convolution | $O(n^6)$ | $O(n^3)$ |
| $n$D Convolution | $O(n^{2d})$ | $O(n^d)$ |

TABLE 1: Time Complexity of Nested and Parallel Loops in nD Convolution

**OUTPUT:**

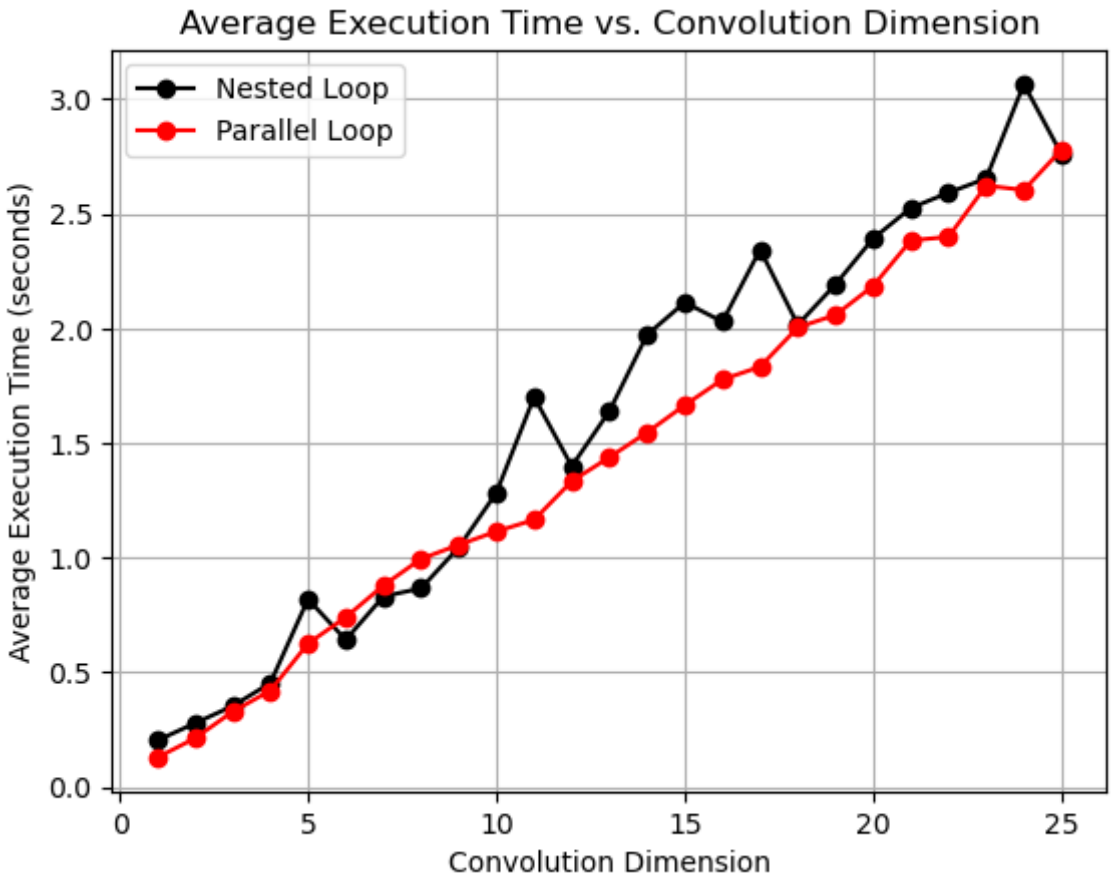


FIGURE 1: **T**ime Complexity of Nested and Parallel Loops in nD Convolution

## 2.2 2D and nD DFT

**AIM:**
To compare the execution times of nested and parallelized Discrete Fourier Transform (DFT) calculations for different dimensions.

**FUNCTION USED:**

- **np.fft.fftn()**: This function is used to perform the multidimensional Fast Fourier Transform (FFT) on an array. The FFT is a mathematical operation that converts a signal from the time domain to the frequency domain. This means that it breaks down the signal into its constituent frequencies.
- **import concurrent.futures**:The import concurrent.futures function imports the concurrent.futures module from the Python standard library. This module provides a high-level interface for running asynchronous tasks.The concurrent.futures module can be used to perform tasks in parallel, which can significantly improve the execution time of long-running tasks.

**OBSERVATION:**

★ The execution time of the nested loop increases as the DFT matrix dimension increases. This is because the nested loop has to iterate over more pixels in the output array for each dimension.

★ The execution time of the parallel loop is initially higher than the nested loop, but it decreases as the DFT matrix dimension increases. This is because the parallel loop has to start up the threads and distribute the work among them. However, once the threads are started up, they can work in parallel, which can significantly reduce the execution time.

★ The parallel loop is faster than the nested loop for DFT matrix dimensions of 10 or higher. This is because the overhead of starting up the threads is outweighed by the time saved by working in parallel.

**Here is a table that summarizes the time complexity of 2D and nD DFT**

| Method | Time Complexity | Complexity Expression |
|---|---|---|
| 1D Nested Loop | $O(d^2)$ | Derived from nested loops over 1D signal |
| 1D Parallel Loop | $O(d^2)$ | Derived from parallelized loops over 1D signal |
| 2D Nested Loop | $O(d^4)$ | Derived from nested loops over 2D signal |
| 2D Parallel Loop | $O(d^4)$ | Derived from parallelized loops over 2D signal |
| nD Nested Loop | $O(d^{2n})$ | Derived from nested loops over nD signal |
| nD Parallel Loop | $O(d^{2n})$ | Derived from parallelized loops over nD signal |

TABLE 2: Time Complexity of DFT on different dimensions

**OUTPUT:**

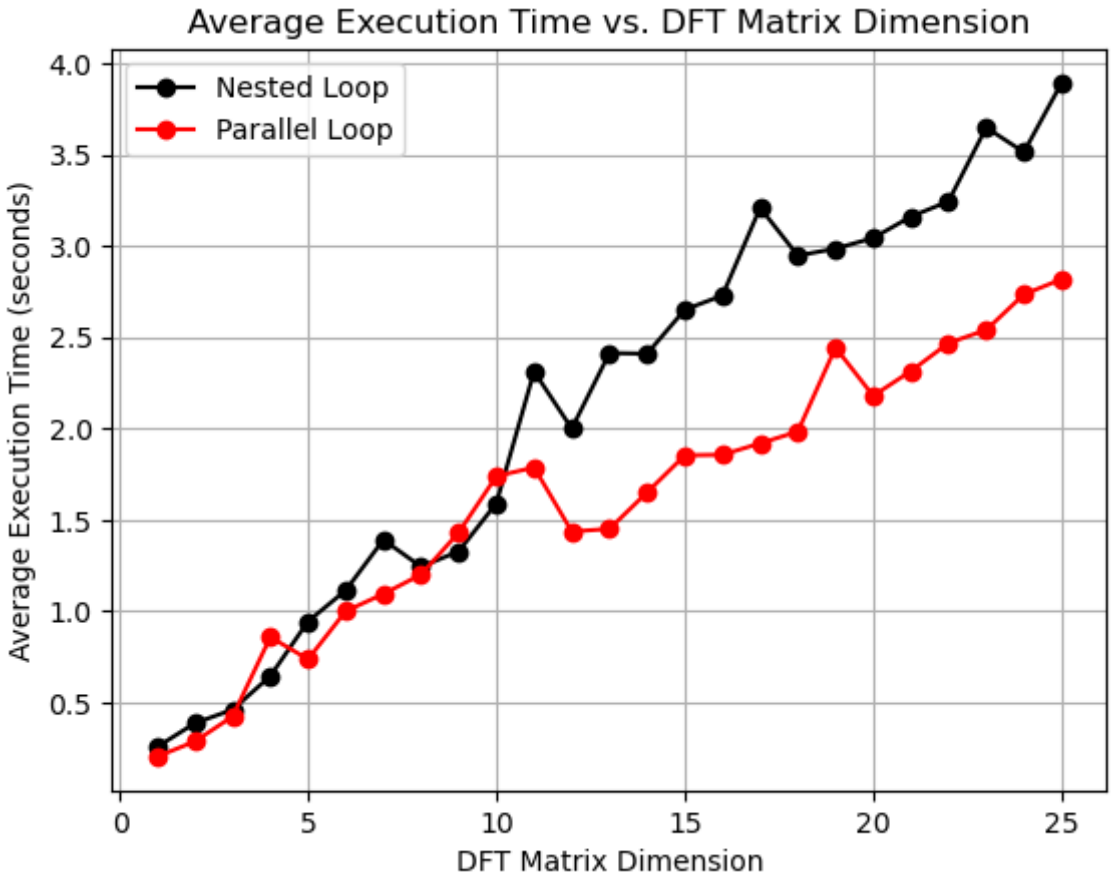


FIGURE 2: **Time complexity comparison of DFT on different dimensions.**