# Operating Systems(DS)

## Implementation of Neural Network using OS Concepts

Deadline: Sunday, 3rd Dec, 2023

- This project can be done in a group of two/three students only (preferably 3 students). Teamwork is an important learning exercise, therefore individual projects are strongly discouraged.
- Zero marks will be awarded to the students involved in plagiarism.
- All the submissions will be done on Google classroom.
- You have to submit .cpp files in Zip Folder named after your roll no (20I-XXXX.zip). Naming convention has to be followed strictly.
- Be prepared for viva or anything else after the submission of project.

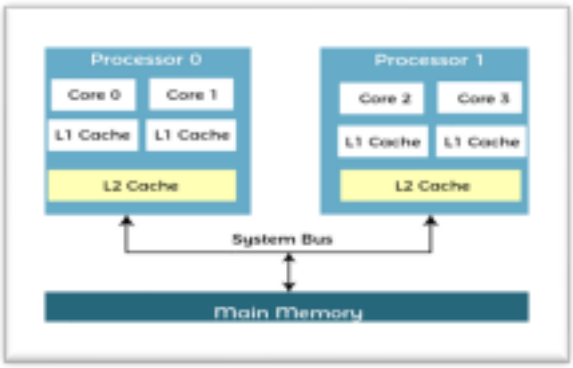
## Problem Statement

You have to design an operating system that implements a neural network architecture using separate processes and threads on a multi-core processor. The system should use inter-process communication through pipes for exchanging information such as weights and biases between processes. Each layer of the neural network should be represented as a separate process, and each neuron within a layer should be treated as a separate thread. During backpropagation, the error signal should be propagated backward through the layers of the network, and the system should update the weights and biases based on the calculated gradients, while utilizing the processing power of multiple cores.

### Detailed Description

Neural networks are a popular machine learning technique that mimics the structure and function of the human brain to solve complex problems. A neural network consists of interconnected layers of neurons, where each neuron takes input from multiple sources, processes it, and produces an output signal.

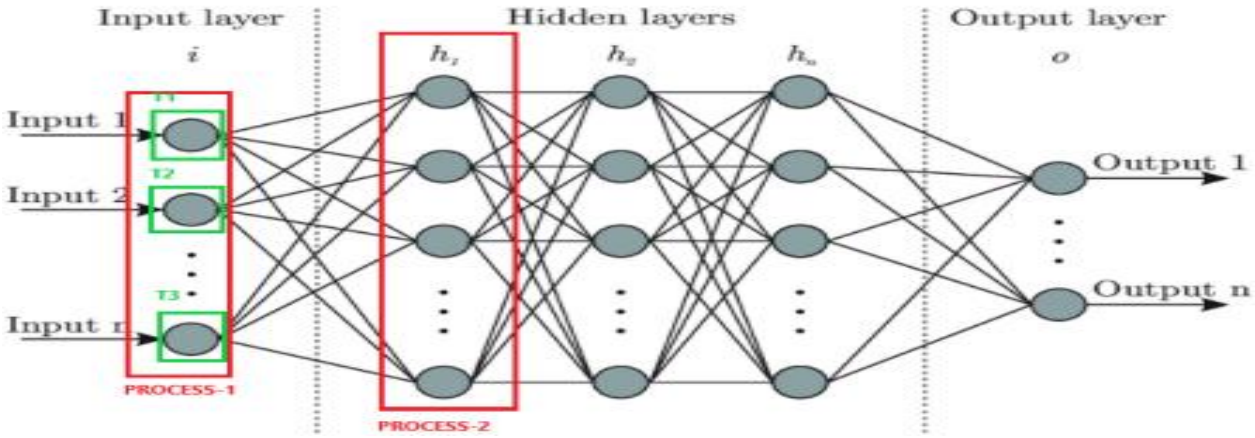
In a multi-core processor, there are several processing units (cores) that can execute multiple threads simultaneously, thereby increasing the system's overall processing power. To take advantage of this capability, the proposed operating system should utilize multi-core processing to parallelize the computation of the neural network.




Each layer of the neural network should be represented as a separate process, and each neuron within a layer should be treated as a separate thread. These threads can be assigned to different cores of the processor to enable parallel processing of the inputs.

The operating system should provide a mechanism for inter-process communication (IPC) through pipes to exchange information between processes. Weights and biases, which are the parameters that the neural network learns during training, should be exchanged through these pipes.

During training, the system should use a batch-based approach, where the input data is divided into smaller batches and each batch is processed by a separate layer in the network. Each layer should receive its input batch from the previous layer through a pipe and then apply the weights and biases to generate the output. The output should be passed to the next layer as input through another pipe.




The system should use locking mechanisms to prevent multiple threads from accessing the same resource at the same time. By distributing the computation across multiple cores, the system can achieve faster processing of large datasets.

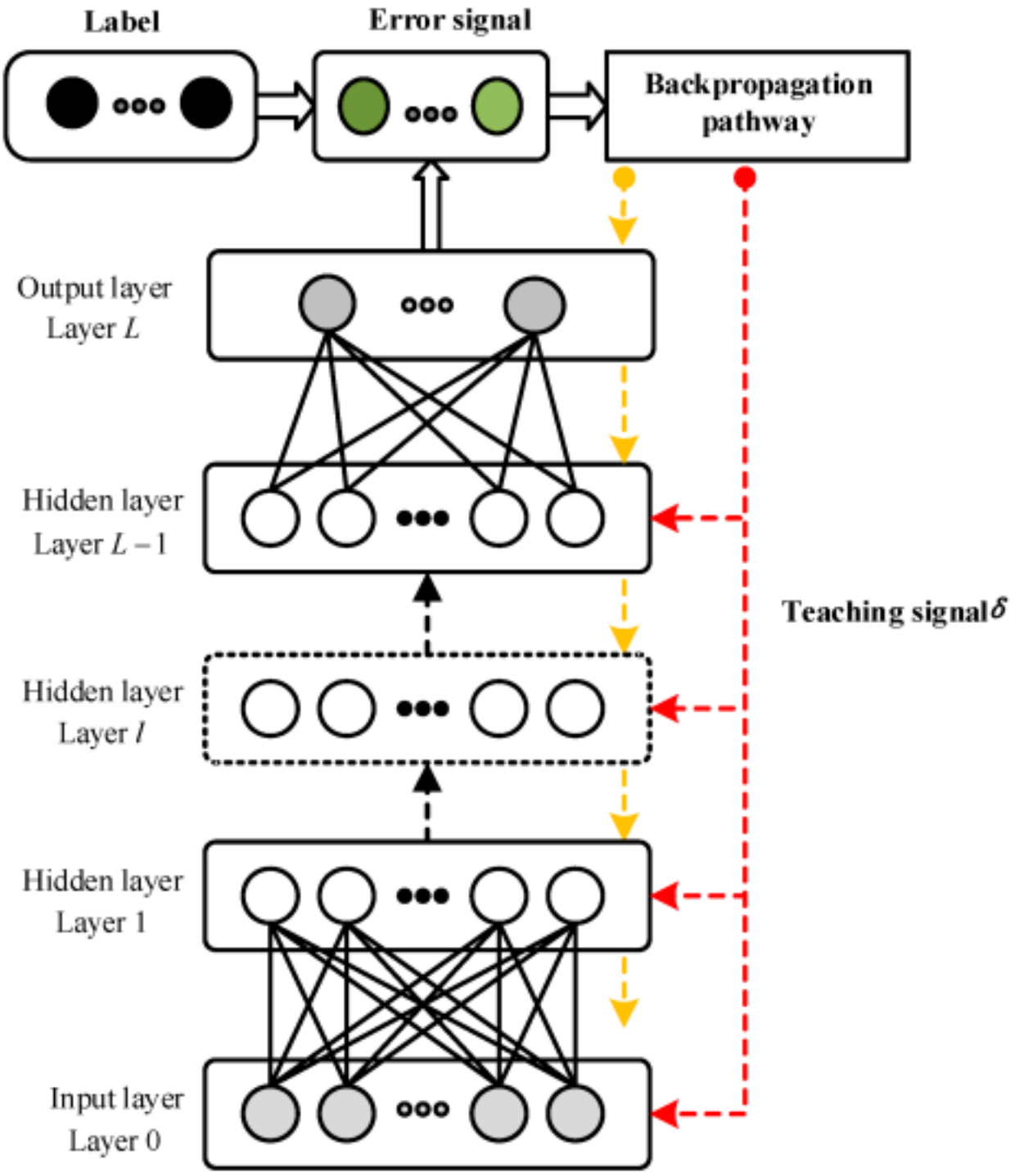
During backpropagation, the error signal should be propagated backward through the layers of the network, and the system should update the weights and biases based on the calculated gradients. You could also use shared memory for backward propagation of weights and bias. To utilize the processing power of multiple cores during backpropagation, the system should divide the computation into smaller tasks and assign each task to a different core.

Furthermore, backpropagation includes:

- **Process Synchronization:** The system should ensure that only one process accesses the shared resources such as weights and biases at any given time to avoid race conditions. This can be achieved using synchronization primitives such as semaphores or locks.
- **Memory Management:** The operating system should provide a mechanism for allocating and deallocating memory for the neural network processes and threads. The system should also ensure that each process and thread has access to its own memory space to avoid conflicts.
- **Inter-Process Communication:** The operating system should provide a mechanism for inter process communication (IPC) through pipes or other forms of communication to exchange information such as weights and biases between processes during backpropagation.
- **Thread Management:** The system should manage threads efficiently to allow parallel processing of inputs and reduce the overall training time of the neural network.
- **Process Scheduling:** The operating system should schedule processes and threads efficiently to ensure that the neural network training process runs smoothly and efficiently.

**Instruction:**

**You can use only C++ and system calls and libraries studied in OS labs (e.g. fork(), wait(), pipes (Named/unnamed), pthread library, mutexes, semaphores etc.) to implement the above-mentioned scenario. The solution should be implemented in Linux (preferably Ubuntu) as demo will be on machines where Ubuntu is installed. Even if you use any other Linux/Mac distro or any other OS, make sure your program runs as expected in Ubuntu environment.**


Label
Error signal
Backpropagation pathway
Output layer
Layer $L$
Hidden layer
Layer $L-1$
Hidden layer
Layer $l$
Hidden layer
Layer 1
Input layer
Layer 0
Teaching signal $\delta$

**GOOD LUCK ☺**