Implementing Parallel Pipeline Training For a **Multilayer Perceptron**

CSC 201 Project - Team Copium

Team Members

23114046
Kartik Goyal

23114016
Gauransh Garg

23114092
Shubham Kataria

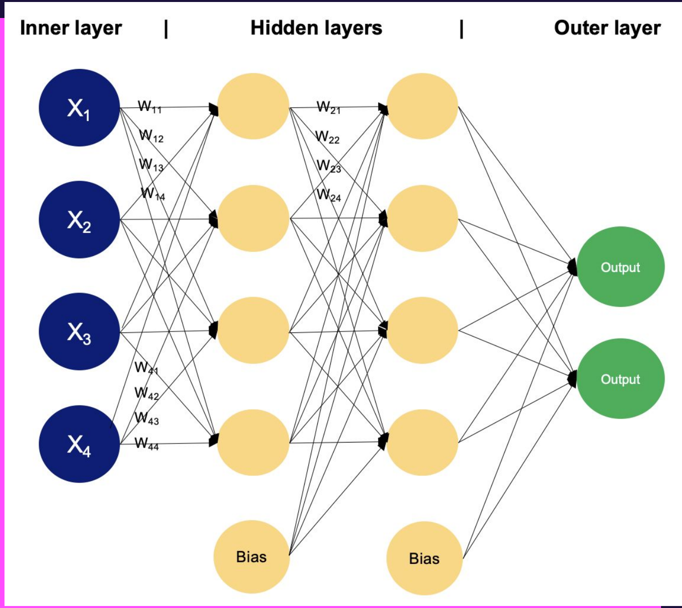
23114063
Parth Baranwal

22114017
Aviral Vishwakarma

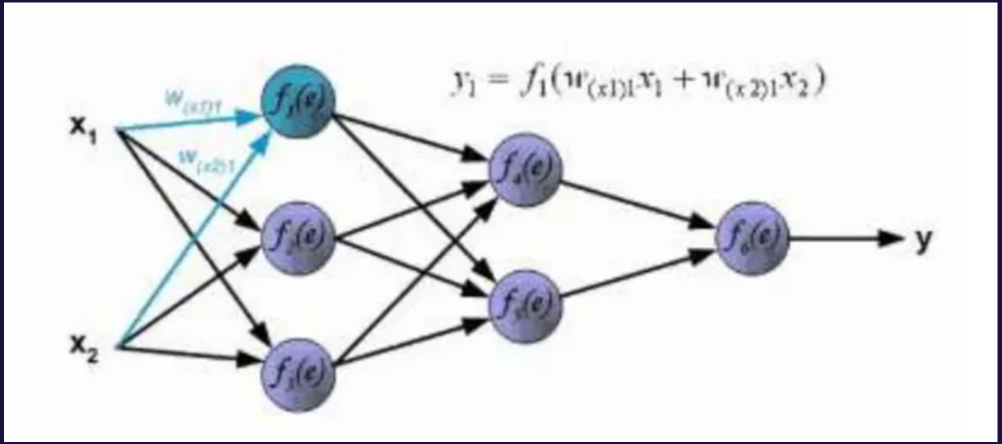
WHAT IS AN *MLP?*

A Multilayer Perceptron (MLP) is a type of artificial neural network used in supervised learning.

- It is composed of multiple layers of neurons connected in a feed-forward manner.
- It consists of an Input Layer, Hidden Layers and an Output Layer. Each neuron in one layer is fully connected to the neurons in the next one.
- Training is done through backpropagation and gradient-descent to minimize the errors.



Backpropagation



Optimize last layer weights w_{kl}

$$L_n = \frac{1}{2}\left(y_n - f(x_n)\right)^2$$

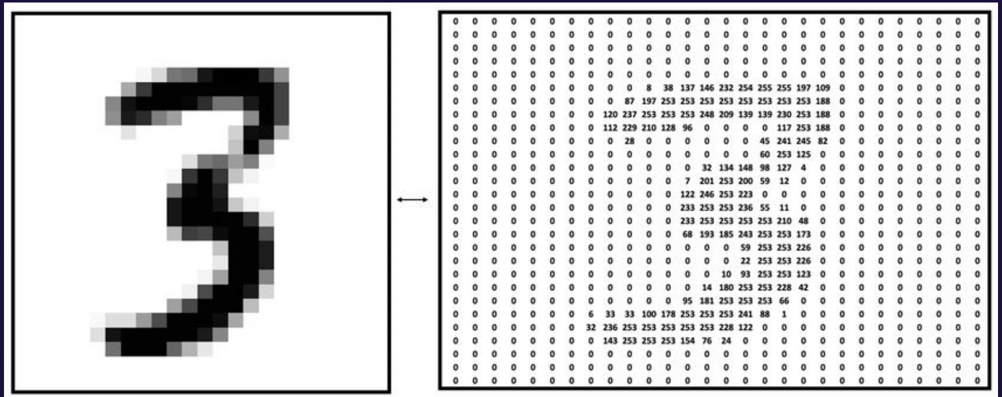
$$\frac{\partial R}{\partial w_{kl}} = \frac{1}{N}\sum_n \left[\frac{\partial L_n}{\partial a_{l,n}}\right]\left[\frac{\partial a_{l,n}}{\partial w_{kl}}\right]$$

Calculus chain rule

- Consists of 2 Stages : forward pass and backward pass
- Forward Pass : Consists of predicting the output using the existing weights.
- Backward Pass : Modifying the weights, analysing the errors and learning the model.

MNIST Data Set

- Modified National Institute of Standards and Technology (MNIST) database
- Collection of handwritten digits used for training various image processing systems
- Pre-processed and normalized images
- Centered digits in fixed-size images
- Balanced distribution of digits (roughly equal numbers of each digit)



In this project, we are considering 3 hidden layers : of 512, 256 and 128 nodes.

Introduction to Parallel Pipelining

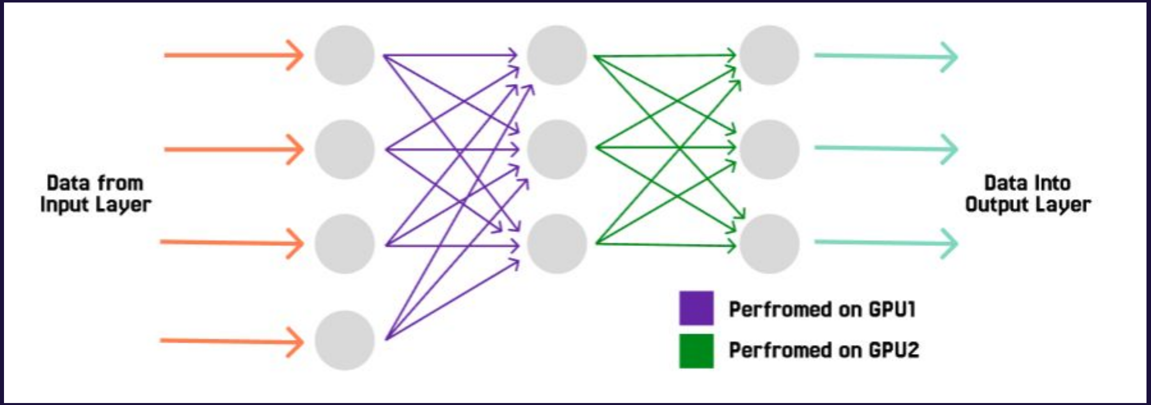
01 Definition

Technique to divide tasks into subtasks that can be processed simultaneously in stages similar to that for an assembly line.

02 Advantages

Increases Computational efficiency by overlapping operation, leading to faster training.

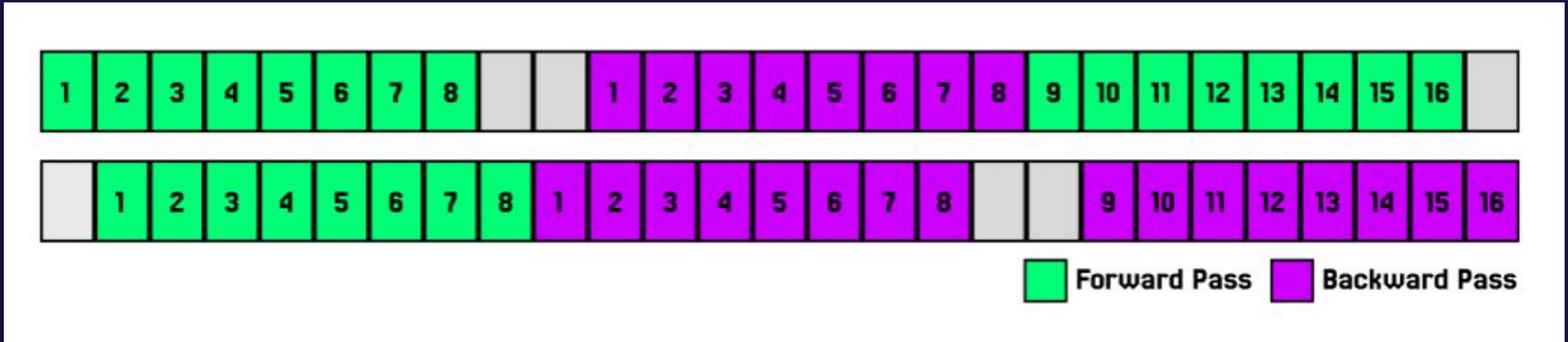
Implementing Pipelines



- In case of multiple GPUs, we can implement each stage of passing data into separate GPUs, encouraging parallelly pipelined processing.
- Each of the GPUs will be communicating with each other using NCCL framework.

- This data is done for a single pass, however parallelism could be achieved on a bigger level by implementing the forward and backward pass operations of different stages on separate GPUs.

Multistage Implementation



- Each forward and backward pass is divided into 2 substages and plugged into the 2 GPUs.
- This way, we are expected to achieve a 2x speedup but there's also an overhead of inter-GPU communication.
- Hence the final speedup lies within a stage of 1 to 2.

Conclusion

- A moderate speedup is achieved on implementing the given model using a 2 GPU pipeline.
- There exists a speed vs accuracy competition in doing the task using serial and pipelined architecture.

```
=== Performance Results ===
Regular Training Time: 76.37 seconds
Parallel Training Time: 53.78 seconds
Speedup: 1.42x
Improvement: 42.00%

gauransh@system:~/experiment/MLP_Pipeline$
```

- A sequential implementation would yield a more accurate output but would take very high amount of time, hence recommended for usage when hidden layers are less.
- On the contrary, a pipelined implementation would take much less time but would compromise on the accuracy, hence recommended for bigger networks for faster training.

THANKS!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon** and infographics & images by **Freepik**

Please keep this slide for attribution

References

- PyTorch Documentation
- Research Paper: Huang, Y., et al. (2019). GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism.
- Lecture: Carnegie Mellon University. (n.d.). Lecture 25: Parallel Deep Learning - Model & Pipeline Parallelism.