SCALABLE MACHINE LEARNING

THROUGH COLLABORATIVE PROCESSING NETWORK

GROUP 12

By
Priyal Dharmendra Patel
Mouli Naidu Lukalapu
Harsh Harnish Patel
Kishore Murugan

INTRODUCTION

Our project introduces a distributed system that optimizes machine learning workflows across multiple hosts. By assigning specific tasks like image preprocessing and model management and loading on different nodes connected to host, all the functionalities were implemented using Python's socket library for communication, efficiency and speed.

KEY FEATURES:

- **DYNAMIC HOST INTEGRATION:** Seamlessly integrates new hosts using socket programming and integrating network functionalities like FTP, remote execution of commands and automatically adjusting task distribution.
- **EFFICIENT PARALLEL PROCESSING:** Combines multiprocessing and socket programming to accelerate improve processing processing and improve scalability.

PROBLEM STATEMENT

Why execution of task on multiple devices?

Simple, time efficient for large scale operations like data preprocessing and data model loading.

Why we need a centralized node?

Without a coordinated system, managing machine learning tasks across multiple hosts leads to inefficiencies and conflicts, similar to a traffic jam. This results in resource competition, communication errors, and challenges in scalability when adding or removing hosts. Our project develops a distributed system that has server acts like traffic control, efficiently managing tasks, ensuring clear communication, and facilitating smooth scalability.

HOW NETWORK CAN HELP ACHIEVING FASTER EXECUTION?

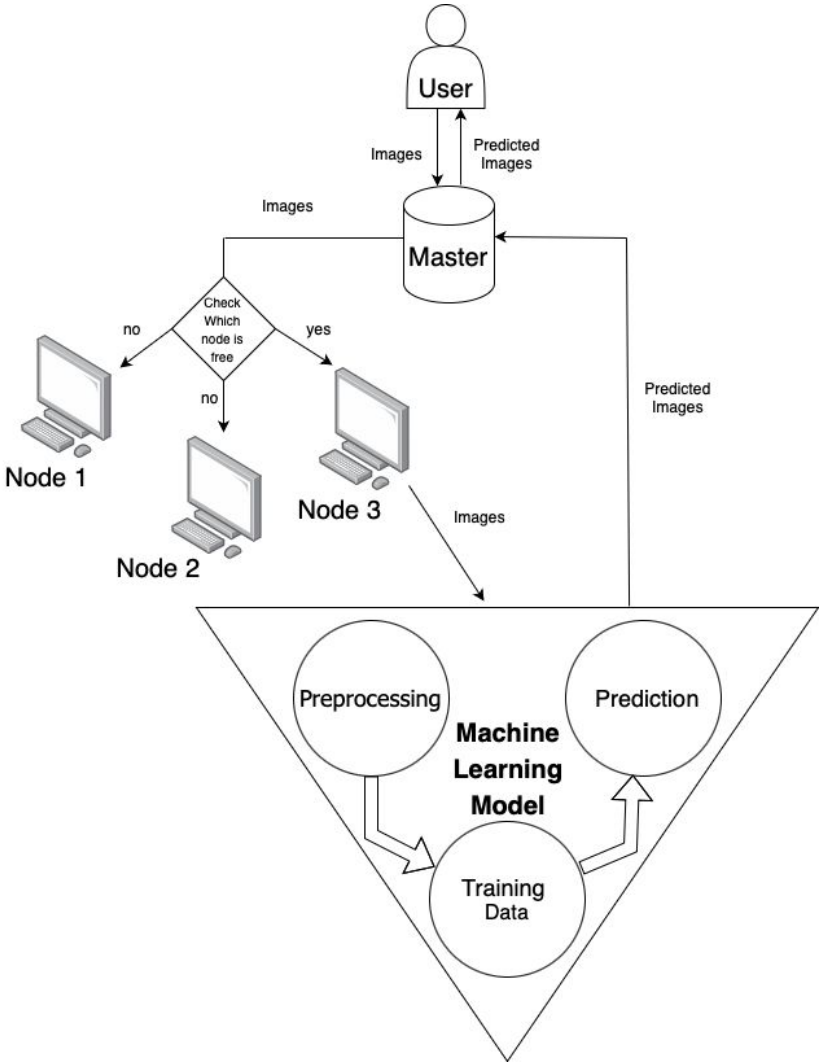
CONCURRENT PROCESSING: Distributing tasks across the network allows multiple nodes to work simultaneously, reducing the overall time for execution.

RESOURCE EFFICIENCY: The network allocates tasks based on the capability of each node, ensuring optimal use of resources and preventing bottlenecks.

DYNAMIC ADAPTABILITY: As nodes join or leave, the network automatically adjusts, maintaining efficient operation without manual reconfiguration.

DATA TRANSFER OPTIMIZATION: Efficient protocols and compression techniques minimize data transfer times, speeding up workflow execution.

FLOW DIAGRAM



FUNCTIONS OF MASTER NODE

SEND IMAGE RECOGNIZER FILES TO NEW NODE

How are we keeping track of devices joining for the first?

SEND REQUIREMENTS: Directs nodes to install Python libraries as needed for the connected node.

CHECK AVAILABILITY: Verifies whether a node is available to receive and process user-submitted images for prediction.

RECEIVE AND DISPATCH PREDICTED IMAGES: Receives the predicted images from nodes and forwards them to the user.

FUNCTIONS OF NODE

REGISTER PROCESS: Registers itself as an image recognizer.

SEND AVAILABILITY: Communicates its availability to the master node.

PROCESS AND RESPOND: Receives images from the master, processes them, and sends the predicted images back to the master.

SHUTDOWN COMMAND: Uses a shutdown command to turn off the node server independently.

FUNCTIONS OF USER

SEND IMAGES: Transmits images to the master node.(but the user never knows who it is sending)

RECEIVE PREDICTED IMAGES: Receives the processed images from the master node.

HOW WE IMPLEMENTED ?

SOCKET PROGRAMMING: Utilized for enabling real-time communication between the master node and other nodes. This ensures efficient management of tasks and data transfer within the network.

FILE TRANSFER: Implemented socket programming for file transfer between nodes. This capability is crucial for sending and receiving images and other data required for processing.

IMAGE PREDICTION: Developed algorithm for image recognition and prediction using OpenCV. The resultant data model are deployed at various nodes to process the incoming images and recognize the persons, which are then sent back to the master node.

FUTURE WORK

SECURITY: Enhance security protocols to safeguard data and communications.

STATIC IP: Implement static IPs for improved network stability.

FAILOVER MECHANISM: Develop failover processes for automatic master node selection in the event of failures.

LOGS: Improve logging for better troubleshooting and monitoring.

USER INTERFACE: Create a user-friendly interface to simplify system interactions.

MODEL UPDATES: Set up a system for regular updates and sharing of machine learning models across nodes.

COVERING WIDE RANGE OF TESTING SCENARIOS

CONTRIBUTION

Priyal

Node and User Setup Logic, File Transfer

Mouli

Architecture, Remote Command Execution

Kishore

Image Recognition Model, Documentation, Keeping network connection intact

Harsh

End to End image transfer across nodes, Testing various scenarios

THANK YOU