EEL3090 Embedded Systems Project

Neural Architecture Search for LLM execution on mobile/resource-constrained devices

Team Members:

Govind Mali B21EE021

Anshul Tomar B21EE083

Overview of Neural Architecture Search (NAS):

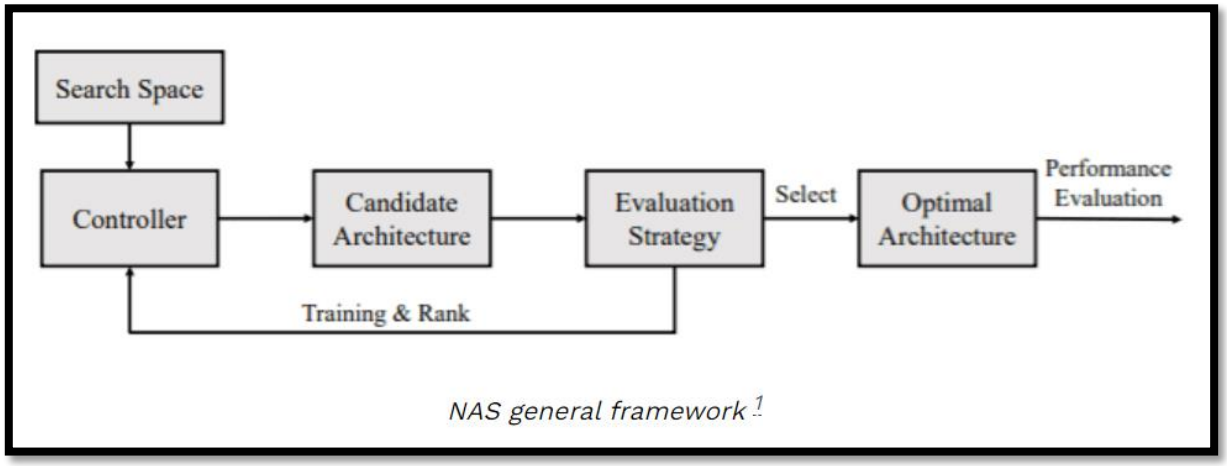
Neural Architecture Search (NAS) revolutionizes the process of designing neural network architectures by automating the exploration of various network topologies to achieve optimal performance for a given task and hardware. Instead of relying on human intuition and trial-and-error, NAS employs computational methods to efficiently navigate the vast search space of possible architectures.

The NAS process typically involves three key components:

a) the search space, the set of possible architectures that the NAS algorithm explores.

b) the search strategy, NAS algorithms employ different strategies to navigate the search space and discover promising architectures.

c) the performance evaluation technique, evaluating the performance of candidate architectures is crucial for guiding the search process



NAS general framework [1]

Implementation:

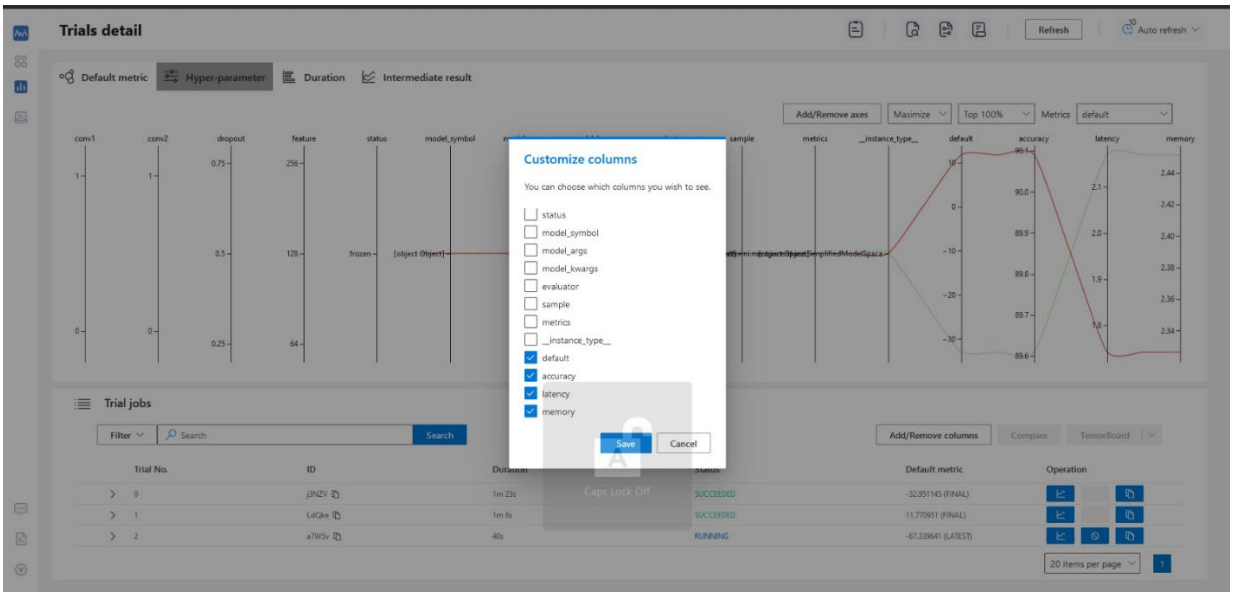
1. In this section, we detail the implementation process of Neural Architecture Search (NAS) for optimizing models using the MNIST Fashion dataset. Due to the time-intensive nature of training Large Language Models (LLMs) and the lack of a predefined search space for LLMs, we focused on the MNIST Fashion dataset as a proxy task. We utilized the Neural Network Intelligence (NNI) toolkit developed by Microsoft to streamline the NAS process.
2. We employed three NAS strategies: **Random sampling** explores diverse architectures without bias, **Gaussian exploration** guides the search towards promising regions using insights from past trials, and **Genetic Evolution** mimics natural selection to iteratively improve architectures through selection, crossover, and mutation operations.
3. We evaluated candidate architectures for hardware optimization using **latency, memory usage, and accuracy metrics**. Latency measured the inference time on target hardware, memory usage gauged RAM consumption during inference, and accuracy assessed classification performance on the MNIST Fashion dataset.
4. We used NNI documentation and APIs for orchestration, hyperparameter tuning, and result visualization. By evaluating architectures based on latency, memory usage, and accuracy, and utilizing diverse search strategies, we identified hardware-optimized models tailored to specific resource constraints.

ReadMe: Steps to setup Project

1. Our project consists of a Python notebook where we implemented Neural Architecture Search (NAS) for the MNIST Fashion dataset.
2. **Prerequisites:** Ensure that PyTorch and NNI are installed on your system. If not installed, follow these steps:

```
pip install nni
pip install torch torchvision
```

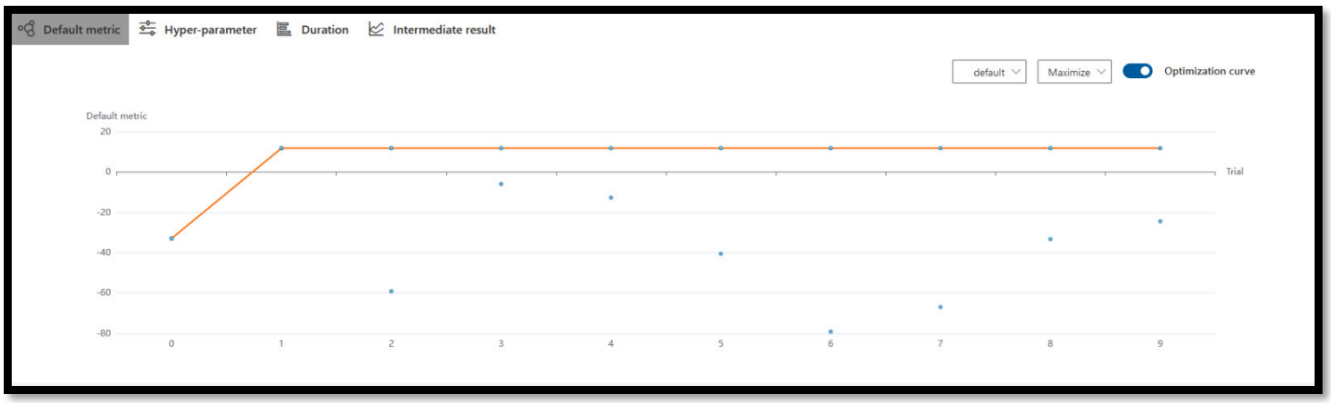
3. Execute the code cells in the notebook sequentially. Each cell contains code for a specific part of the NAS implementation.
4. **Visualizing Results**: To visualize the results of different strategies, click on the respective URL provided in the output of the cell. For example, if the output displays experiment1.run(port:8085), go to localhost:8085 in your web browser. You will find a web interface of NNI where you can monitor the progress of the NAS experiment and visualize metrics.



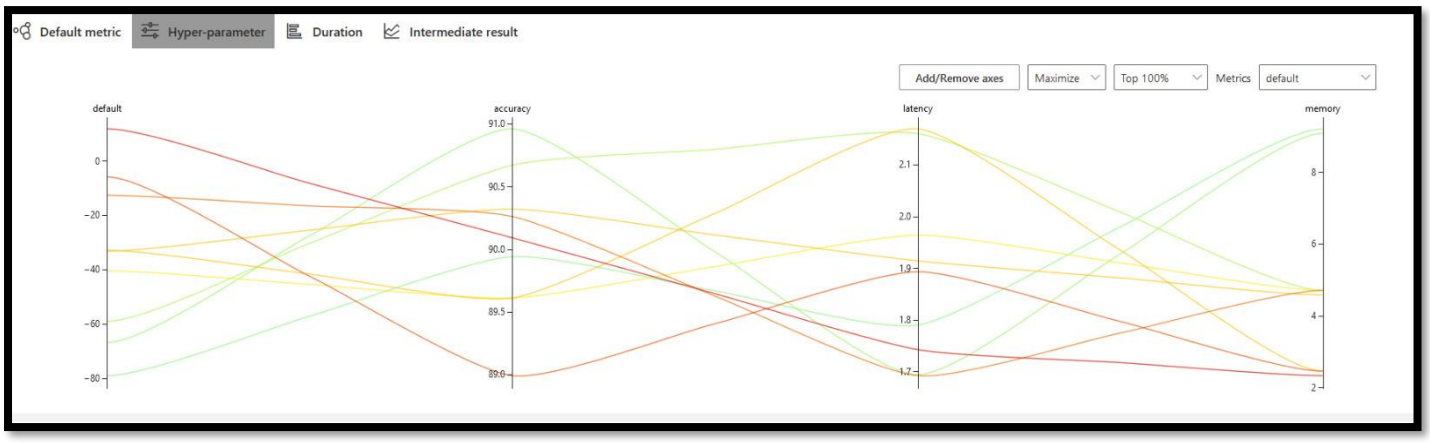
5. Exploring Optimized Models: After the completion of each experiment, the next experiment will run automatically. You can observe the optimized model's parameters in the NNI interface. These optimized parameters can be utilized to create hardware-specific models in future implementations.

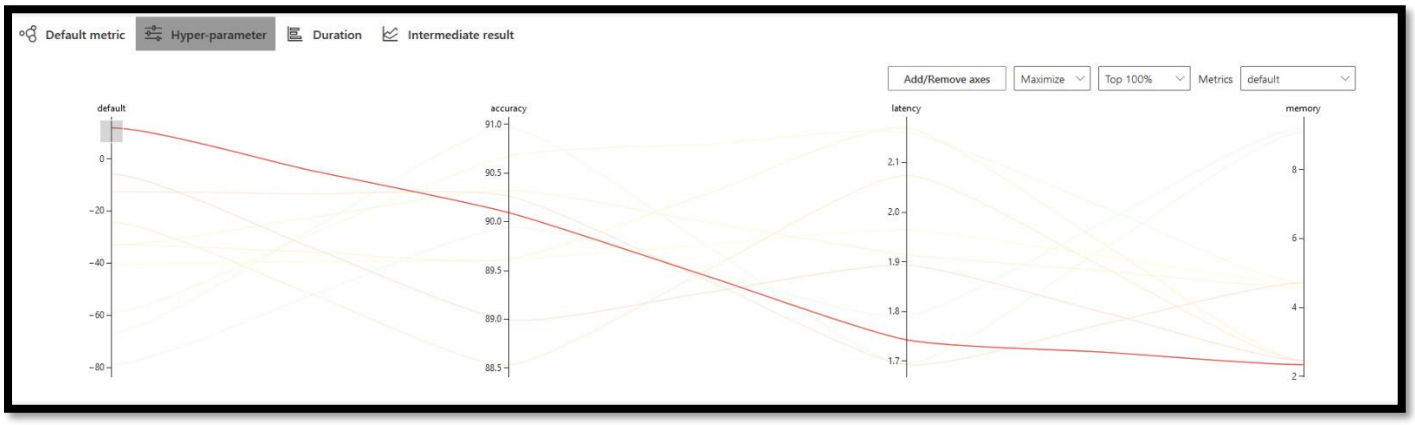
Results:

Strategy : Random Search



Optimization curve vs trial



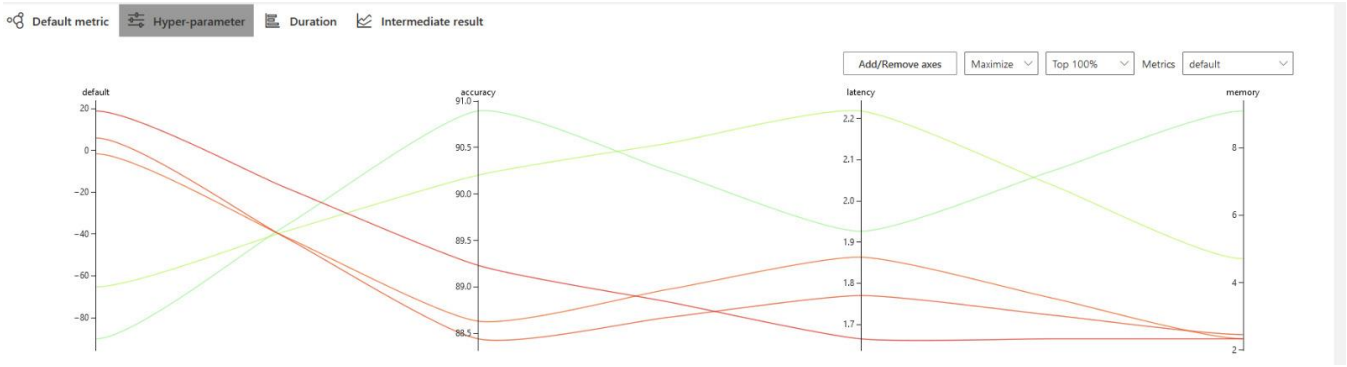


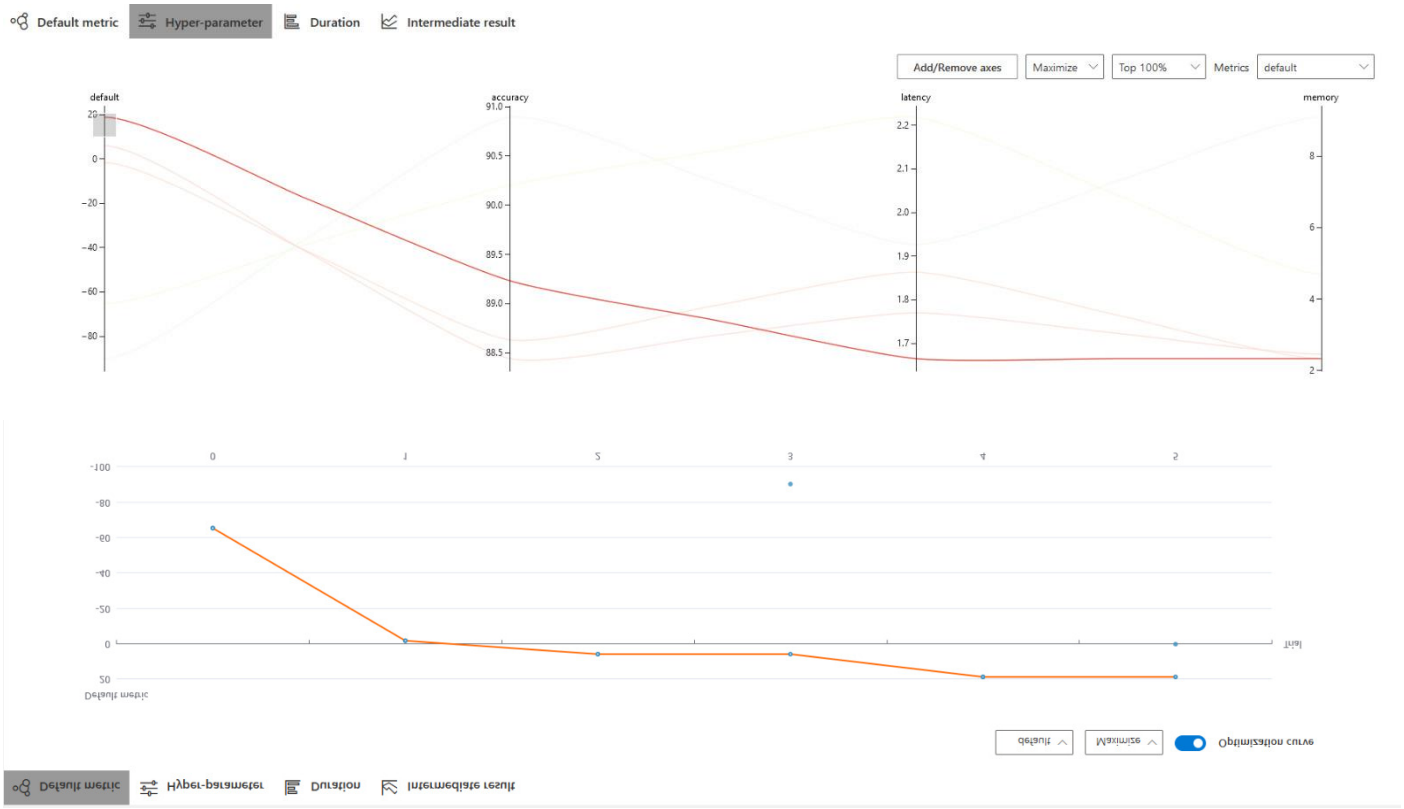
Metrics curve with each trail represented by a curve (best curve highlighted)

Best Model : accuracy : 98.6 ,Latency : 1.75sec .memory : 2.1Mb

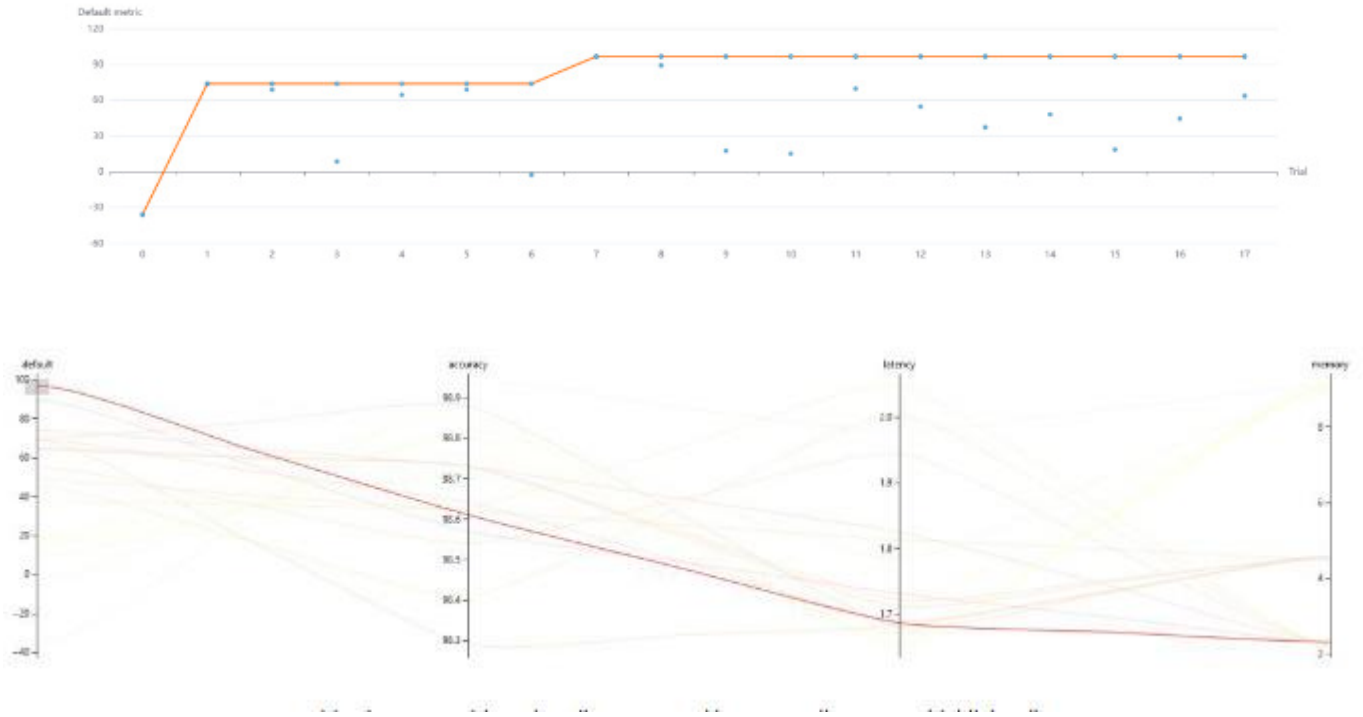
Parameter of best model: conv1:1 conv2:0 dropout:0.25 feature:64

Strategy: Regularized Evolution (Genetic Algorithm)





Strategy: TPE(Gaussian)



Best Model : accuracy : 98.6 ,Latency : 1.69sec .memory : 2.1Mb

Conclusion: NAS methods could be used to achieve hardware aware models for a specific hardware architecture by writing custom metrics or evaluation procedure but its time consuming