Neural Architectural Search for Dynamic Inference

Context:

This project aims to explore Neural Architecture Search (NAS) for big/little dynamic inference networks. Current dynamic inference solutions typically cascade two separate models or implement early exit strategies.

This approach limits adaptability, as the models are trained independently and do not adjust their architecture jointly based on the data.

This project proposes a framework where the big and little models are co-trained in a NAS-supernet-like manner, allowing their dimensions and architecture to evolve dynamically during training.

The goal is to discover dynamic models optimized for edge deployment, capable of reducing inference cost while maintaining accuracy.

Objectives:

- Data and Dataset Preparation: Benchmark on standard vision datasets (e.g., CIFAR-10, ImageNet, or other edge-relevant datasets). Preprocess data to support dynamic inference evaluation.

- NAS Exploration: Investigate and implement different NAS strategies, including supernet training, evolutionary NAS, and differentiable NAS, for discovering coupled big/little models.

- Dynamic Model Training: Co-train big and little components in a unified framework, exploring dimension adaptation (width, depth, or block-level scaling) during training.

- Edge Deployment: Deploy the discovered architectures on edge hardware (e.g., Jetson Nano, Raspberry Pi, or mobile devices) and evaluate latency, memory, and energy consumption.

- Evaluation: Benchmark accuracy, inference cost, and dynamic behavior. Compare against standard cascaded or early-exit baselines.

Methodology:

- Dataset Preparation:
    - Select vision datasets suitable for edge deployment.
    - Perform preprocessing and data augmentation.
    - Partition datasets to support NAS validation and testing.

- NAS Framework Implementation:
    - Build a supernet representing the search space of coupled big/little models.
    - Implement differentiable NAS or evolutionary NAS techniques to search for optimal architectures.
    - Include data-aware components, allowing routing or scaling decisions to depend on input characteristics.

- Dynamic Co-training:
    - Train big and little models jointly, allowing their parameters and dimensions to evolve during training.

- Edge Deployment and Benchmarking:
    - Deploy selected architectures on edge hardware.
    - Measure latency, memory footprint, and energy efficiency.
    - Compare performance to static and cascaded baselines.

- Analysis and Documentation:
    - Analyze trade-offs between accuracy, efficiency, and adaptability.
    - Document methodology, results, and insights for future work.

Expected Outcomes

1. A framework for NAS-driven dynamic big/little models with joint training and input-adaptive scaling.
2. Benchmarked models for vision tasks showing accuracy vs. inference cost trade-offs.
3. Edge-deployable models with measured latency, energy, and memory efficiency.
4. Evaluation of different NAS strategies in the context of dynamic models.

Project Timeline

Step 1: Literature review and selection of benchmark datasets.

Step 2: Implement NAS frameworks and define the big/little supernet search space.

Step 3: Train and evaluate dynamic coupled architectures on vision datasets.

Step 4: Deploy models on edge devices, benchmark efficiency, and compare baselines.

Step 5: Documentation and final presentation of results.

Team Roles

Data Engineer: Prepares datasets, handles preprocessing and augmentation.

Machine Learning Engineer: Designs NAS framework, co-training strategy, and dynamic inference policies.

Edge Software Developer: Implements model deployment on edge devices, measures latency and energy metrics.

Evaluation Lead: Designs benchmarking protocols, evaluates accuracy vs. efficiency trade-offs, and documents results.

Open-Source Requirements:

All software developed as part of this project must adhere to open-source principles. Students are expected to provide extensive documentation of their work, writing maintainable, well-structured and publicly accessible code.

The project encourages contributions to existing open-source frameworks when applicable and aims to provide reusable software components for future research and development in this domain.

Contact:

Francesco Daghero

fdag@mmmi.sdu.dk

DECO Lab