Neural architecture search with target hardware control

Firsov Sergey
Supervisor: Bakhteev Oleg

Moscow Institute of Physics and Technology

2024

Problem

- **Neural Architecture Search:** Automate the design of neural network architectures to solve ML tasks
- **Goal:** Identify architectures that optimize task performance while balancing computational cost and resource efficiency.
- **Challenges:**
  - **Vast Search Space:** The number of possible architectures grows exponentially with model complexity.
  - **Trade-offs:** Balancing accuracy, model complexity, and hardware efficiency.
  - **Resource Constraints:** Models must perform well within predefined latency, memory, or energy limits.

Existing Methods

Exhaustive search for optimal architectures using:

- reinforcement learning
- evolutionary algorithms
- gradient-based methods

Differentiable Architecture Search (DARTS):

- formulates NAS as a continuous optimization problem, enabling gradient-based search.
- reduces computational cost by relaxing the discrete search space into a differentiable one.

Problems with Existing Methods:

- **accuracy vs. complexity:** often focus solely on accuracy, neglecting the importance of model complexity control.
- **hardware constraints ignored:** lack of mechanisms to account for real-world hardware constraints like latency or energy consumption.
- **limited flexibility:** insufficient control over the trade-off between accuracy and resource efficiency.

Architecture of solution

Step 1: Complexity-Aware Architecture Representation

- Architectural parameters $\boldsymbol{\alpha}$ depend on complexity parameter λ:
$$\boldsymbol{\alpha} : \lambda \to \mathcal{A}.$$
- Generates multiple architectures with varying complexities in a single optimization process.
- Ensures flexibility in architecture design across resource constraints.

Step 2: Simplex-Based Complexity Control

- Replace scalar complexity parameter (λ) with a simplex representation:
  $$\boldsymbol{S} \in \Delta^{k-1}.$$
  This makes complexity management more flexible for different types of operations.
- Δ^{k-1}: $(k-1)$-dimensional simplex, where k is the number of operations.
- Sampling via Gumbel-Softmax enables differentiable selection of operations:
  $$\boldsymbol{S} \sim \mathrm{GumbelSoftmax}(\boldsymbol{\theta}).$$

Step 3: Hardware-Aware Latency Regularization

- **Latency-Aware Optimization:** Introduce a term in the loss function to account for operation latency on target hardware.

  $$\mathsf{Loss} = L + \kappa \cdot \mathsf{Latency}(\boldsymbol{\alpha}),$$

  where κ is a regularization coefficient.
- **Purpose:**
  - Ensure the resulting architecture meets real-world hardware constraints, such as inference time limits.
  - Optimize architectures not just for accuracy, but for deployment feasibility on specific devices.

Final Loss Function

- **Optimization problem:** minimize functional

$$\mathbb{E}_{\boldsymbol{S}\sim\text{GumbelSoftmax}}\Big[L(\boldsymbol{w}^*(\boldsymbol{S}),\boldsymbol{\alpha}(\boldsymbol{S}))+\lambda\cdot\text{Reg}(\boldsymbol{\alpha}(\boldsymbol{S}))+\kappa\cdot\text{Latency}(\boldsymbol{\alpha}(\boldsymbol{S}))\Big].$$

- **Key Insights:**
  - by using complexity dependence architecture parameters we generate multiple architectures in a single optimization process,
  - by replacing the scalar complexity parameter with the simplex representation, we achieve finer control over architectural complexity, enabling flexible optimization across a range of hardware and resource constraints,
  - by using latency lookup table, we can control hardware constraints.

Planned Experiments

- Validate the proposed method on:
  - Image classification benchmarks (e.g.,fmnist, CIFAR-10).
  - Hardware-specific performance metrics.
- Compare against:
  - Baseline methods (DARTS, ProxylessNAS, FBNet).
  - Metrics: Accuracy, latency, and complexity.
- Evaluate trade-offs between accuracy and latency.

Results

- Placeholder for experimental results.
- Future work: Analyze results and provide insights.

References I