AdaptiveNet: Post-deployment Neural Architecture Adaptation for Diverse Edge Environments

Hao Wen¹, Yuanchun Li¹, Zunshuai Zhang³, Shiqi Jiang³, Xiaozhou Ye⁴, Ye Ouyang⁴, Yaqin Zhang¹, Yunxin Liu¹

¹Institute for AI Industry Research, Tsinghua University
²Shanghai University ³Microsoft Research ⁴AsiaInfo Technologies (China), Inc.

2023-10-7
AI is Transforming the World, with Cloud + Edge

ChatGPT

Cloud AI
multi-domain, multi-task, general-purpose services

Edge AI
Domain-specific, real-time, privacy-sensitive applications
Environment Diversity is a Main Challenge in Edge AI

- **Device diversity is a main challenge**
  a) hardware diversity
  b) Intra-device diversity (backend number, software version, temperature)
  c) data distribution diversity

- DNNs are expected to meet certain constant latency requirements.

**Challenge: Generate models for diverse edge environments.**
Conventional: Pre-deployment Model Generation

- **Most popular techniques**: Neural Architecture Search (NAS), Model Pruning, etc.

- **Limitations**:
  1. **Requires collecting privacy information** about computational resources, runtime conditions, data distribution, etc.
  2. **High maintenance cost**. Less practical in many edge/mobile scenarios where the model execution environments may be very diverse and dynamic.
Conventional: Pre-deployment On-cloud Model Generation

3. Modeling the edge environment may be difficult.
   - The cloud-based model generation relies on accuracy and latency predictors.
   - The unified accuracy predictor may not perform well for edge devices with data distribution shifts.

Performance of accuracy predictor on non-iid edge data. The edge data is simulated with Dirichlet distributions with (a) $\alpha=0.005$ and (b) $\alpha=0.1$. The sample ratios of top-50 classes are shown in (c) and (d).
Solution: Post-deployment Neural Architecture Adaptation

Benefits:

• Directly evaluate the a given DNN *without accuracy predictor*, which is more precise.

• A plug-and-play process, *reduces the computation overhead* of the cloud.

• Protects user privacy.

Related work in mobile community: on-device model scaling (NestDNN, LegoDNN, etc.):

*Limited model space; Still relying on performance predictors.*
Challenges

Generating the model search space for edge devices is difficult.

- The search space should be **large** and **flexible** enough.
- Should contain **high-quality candidate models** for edge devices.

The model performance evaluation process can be time-consuming at the edge.

- **Limited computing resources** and tight deadline of model initialization.
- The edge environment is **dynamic**.
Method
AdaptiveNet: System Design

The architecture overview of AdaptiveNet
1. Given an arbitrary pre-trained DNN, We discover the repeating basic blocks \((B_0^{(0)} \sim B_n^{(0)})\) in the DNN.
2. We convert the given pre-trained DNN into a **supernet** by adding *merged blocks* \((B_i^{(1)}, B_i^{(2)})\) and *pruned blocks* \((B_i^{(-1)})\). The **supernet** encompasses a large search space of **subnets**.
Cloud Stage: Distillation-based Training

**Branch distillation phase:**

- Adopt feature-based knowledge distillation (Pre-trained model as the teacher).
- In each iteration, randomly sample a subnet from the supernet and use the pre-trained model as the teacher model to train the new branches in the subnet.
Cloud Stage: Distillation-based Training

Further tuning phase:
• Further train the supernet using labelled data.
• In each iteration, randomly sample a subnet and forward a batch of samples, compute the Cross-Entropy loss and update the parameters of the new branches.
• **Edge Stage** is to obtain the optimal architecture adaptively in the target environment by searching the subnet space.

• *Using a normal search method as in NAS can cost more than 10 hours on edge devices*. Most of the searching time is spent on evaluating the subnets.
Edge Stage: Search Strategy

Taking the example of Genetic Algorithm (GA) - based search strategy:

1. Build \textit{Latency Table} \( T = \{ t_i^j \} \) (\( t_i^j \) is the latency of \( B_i^j \)). Thus, the latency of a chosen subnet is the sum of all its blocks.

2. Generate the \textit{initial candidate subnets} by randomly sampling a group of subnets whose latencies are near the \textit{latency budget}.

3. In each iteration, mutate subnets by replacing branches. \textit{(Make sure the mutated subnets are also near the latency budget)}. 
Edge Stage: Evaluator

- In each iteration, we usually need to evaluate *hundreds of candidate subnets* with the edge data to find the most accurate ones.

- The candidate subnets usually share common prefix substructures, so we can reuse common intermediate features across subnets.

- We introduce a *tree-based feature cache* to schedule the evaluation (Right Figure).
Edge Stage: Dynamic Model Update

- After searching, the subnets achieving the highest accuracy at different levels of latency are saved.
- AdaptiveNet dynamically pages in and pages out alternative blocks when the environment changes.
Evaluation
Evaluation: Experimental Setup

1. Edge devices:
   - Android Smartphone (Xiaomi 12) with Snapdragon 8 Gen 1 CPU and 8 GB memory
   - Jetson Nano with 4 GB memory
   - Edge server with NVIDIA 3090 Ti with 24 GB GPU memory

2. Baselines:
   - LegoDNN [1]: a pruning based, block-grained technique for model scaling
   - Slimmable Networks [2], FlexDNN [3], SkipNet [4]: dynamic neural networks with flexible widths, depths, and layers.

3. Tasks, Models, and Datasets:

<table>
<thead>
<tr>
<th>Task</th>
<th>Model</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object detection</td>
<td>EfficientDet</td>
<td>COCO2017</td>
</tr>
<tr>
<td>Semantic segmentation</td>
<td>FPN</td>
<td>CamVis</td>
</tr>
</tbody>
</table>

[1] Han et al. LegoDNN: Block-Grained Scaling of Deep Neural Networks for Mobile Vision. (MobiCom 2021)
Evaluation: Model Scaling

- AdaptiveNet achieves higher accuracy than baseline approaches at almost every latency budget.

- Increases accuracy by 10.44% and 28.03% on average compared to LegoDNN with 90% and 70% latency budget respectively.

- AdaptiveNet outperforms the baseline models more at a lower latency budget thanks to the merging blocks.
Evaluation: Model Scaling and Training Efficiency

Quality of models generated for detection and segmentation tasks.

- Optimal accuracy achieved with different num of subnets.
- Optimal accuracy achieved with different search time.
- Speed of evaluating a group of subnets.

Training efficiency of on-cloud elastification.
Conclusion and Outlook

• AdaptiveNet is a novel approach for **on-device, post deployment, and environment-aware** model architecture generation.

• It is an end-to-end system equipped with **on-cloud model elastification** and **on-device model adaptation**.

• Future work
  
  • Generalize AdaptiveNet to **pre-trained/foundation models**.
  
  • Design supernets that can adapt to **edge data distributions**.
  
  • Generate subnets that can deal with **domain-specific tasks** directly.

Open sourced: https://github.com/wenh18/AdaptiveNet
Thanks!

AdaptiveNet: Post-deployment Neural Architecture Adaptation for Diverse Edge Environments

Hao Wen¹, Yuanchun Li¹, Zunshuai Zhang³, Shiqi Jiang³, Xiaozhou Ye⁴, Ye Ouyang⁴, Yaqin Zhang¹, Yunxin Liu¹

¹Institute for AI Industry Research, Tsinghua University
²Shanghai University ³Microsoft Research ⁴AsiaInfo Technologies (China), Inc.