Edge Machine Learning for Resource-constrained IoT Devices

Yichen Ruan, Haocheng Fang, Jinhang Zuo
Carnegie Mellon University

Introduction

Edge computing aims to fully utilize the computing power of resource-constrained devices that are physically close to end users. The deployment of machine learning algorithms on the edge architecture is appealing for applications like real-time anomaly detection and offline identity verification, which require low latency, strong privacy protection and security guarantee.

However, existing distributed machine learning algorithms typically fail to work due to the resource-constrained nature of edge IoT devices. In this project, we propose a tree-based classification model for multiple resource-constrained IoT devices. It utilizes prior knowledge of taxonomy to build a large classification tree with hierarchy, then prunes the tree based on the number of available devices, while minimizing accuracy loss.

Our classification model has two main advantages. First, it has very low requirements on the computing power of each node. It can combine numerous resource-constrained devices to obtain a classification model with high accuracy that is otherwise not possible on such devices. Second, in our tree-based model, each parent node only propagates the request to one of its child node. Therefore, it allows for high levels of parallelization and increased throughput when working on large number of images.

Method

Our idea is to **group classes into some super-classes**. Thus instead of building one single classification model over the entire dataset, each individual edge node works on a small group that contains only a subset of the whole data.

For example, suppose we have 3 devices, and we want to classify human images into 4 categories: women, girls, men and boys. We can create two super-classes: females vs. males, or adults vs. children.

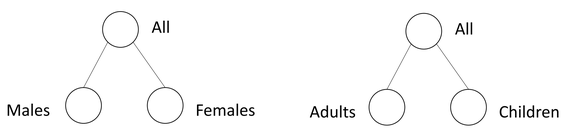

Figure 1. Possible tree structures for 4 classes: women, girls, men, boys

Generally, suppose we want to classify **L** classes with **K** devices, our objective is to find an optimal tree structure that has the highest average accuracy. Where the average accuracy (score) for a subtree rooted at T is defined as:

$$\text{score}(T) = \frac{1}{|T|} \sum_{C \in T.children} |C| \text{score}(C)$$

Without some constraints on the topology of the tree structure, the searching space is exponential w.r.t. **L**. To improve the training efficiency, we propose to use the **prior knowledge from taxonomy**. The reasons are as follows:

1) Taxonomy is a science that groups classes with similar properties, which is usually what machine learning algorithms do;
2) There are many existing algorithms specifically designed for some groups in the taxonomy, which can be reused.

Method (cont.)

We are working on a subset of the Cifar-100 dataset, which has a total of 50 classes, and can be represented with a taxonomy tree as follows.

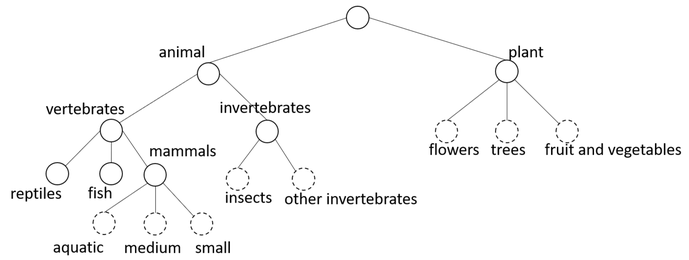

Figure 2. Taxonomy tree for the classes of interest. Dashed nodes are dropped by the algorithm in the experiment.

The original taxonomy tree contains 16 nodes. Each node corresponds to a classifier deployed on a resource-constrained device. In the case when the number of available devices **K** is less than the nodes of the taxonomy tree, we propose to prune the tree with the algorithm below.

Algorithm 1: Taxonomy based Algorithm

function Build-Tree-Taxonomy $(K, data, G)$;
Input : Number of edge nodes K; n observations $data$; Pre-built taxonomy tree G
Output : A tree structure T with at most K nodes.
$T = G$
while $|G| > k$ **do**
  For each new pair (l_1, l_2) with the same parent, try merge them.
  Compute the change of the average score. Store the result into some global table.
  Find in the table the pair (l_1^*, l_2^*) with the highest merging score.
  Merge (l_1^*, l_2^*), insert the new node l_{new} under $p = l_1^*.parent$.
  if $l_1^*.parent$ *has only one child* **then**
    Replace p with l_{new}
  end
end
return T

We generally do not set limit for the types and complexities of individual classification models. It is possible to use totally different models for different nodes. However, for efficiency consideration, it is wise to constraint the choice to a small set of models.

Figure 3. Edge Machine Learning System of 8 Raspberry Pis

Experiment & Result

I. Training

We train the model with Algorithm 1 using the taxonomy tree in Figure 2. The final tree is shown in Figure 2 (solid nodes). For each node, we use Keras to train a CNN model. During the execution of the algorithm, the nodes are merged in the following order:

invertebrates->aquatic+small->trees+fruit&veges->plant->mammals

The training result for the final tree is shown in Table 1.

Table 1. Training result for the final tree structure.

	All	Animal	Plant	Vertebrates	Invertebrates	Reptiles	Fish	Mammals
# sample	25000	17500	7500	12500	5000	2500	2500	7500
Accuracy	0.87	0.76	0.95	0.80	0.91	0.82	0.87	0.94

II. Testing

For testing, we deploy the pretrained models onto 8 Raspberry Pis. Figure 3 shows the deployment of them. All devices are connected to a server, which allows communication between them. As shown in Figure 2, each parent node tells its child nodes which part of data they need to classify. In each node (Pi), we use Keras to load pretrained model and do classification.

Table 2 shows the running time of each Pi for testing. Note that they run in parallel and the total testing completion time is 95 seconds, which is much faster than using a single Pi with a complicated model to classify all the data.

Table 2. Running time for testing.

	All	Animal	Plant	Vertebrates	Invertebrates	Reptiles	Fish	Mammals
Running time (s)	90	61	38	52	8	2	47	5

Discussion

Conclusion:

1. Proposed a tree-based hierarchical classification model; designed a pruning algorithm for resource-constrained edge computing.
2. Implemented our algorithm on a system comprised of 8 Raspberry Pis; achieved fast image classification.

Future Work:

1. Improve the accuracy of individual classifiers with sophisticated models.
2. Inspect the influence of the prior taxonomy structure on the final tree topology as well as the overall performance.
3. Extend the optimization for inhomogeneous edge devices.