A Simple, Effective Way to Improve Neural Net Classification: Ensembling Unit Activations with a Sparse Oblique Decision Tree

Arman Zharmagambetov and **Miguel Á. Carreira-Perpiñán**
Dept. Computer Science & Engineering, UC Merced

Poster Number: 1142

1 Overview and Motivation

- Deep learning has become highly successful in many applications involving complex inputs such as images, audio or text. They compute features that capture important properties of the input and these features can be invariant to certain transformations (translation, rotation, etc.).
- As a result, we have now a proliferation of deep net architectures of ever increasing complexity, containing millions of parameters.
- At the same time, the cost of training such models (computing time, memory size, energy consumption and human expertise) has escalated dramatically. This also leads to diminishing returns: large increases in size within a family quickly translate into tiny reductions in error.
- A different, proven way to construct accurate classifiers is via ensemble learning and we propose a new simple ensembling mechanism that is specially designed for neural nets.

2 Proposed Idea

Assume we have a dataset of input instances and their labels (in K classes); and several **deep nets** trained, somehow, on that dataset. Then we first construct an **ensemble feature vector** by picking a subset of features (output of intermediate layers) from each net and concatenating them; and then we train a **sparse oblique tree classifier** with TAO on a dataset where the inputs are the ensemble feature vectors and the output is their ground-truth label. This procedure is generic and admits multiple variations: we may have just one deep net and ensemble features from its internal layers (called **within net** in our experiments); or multiple nets of different types and pick features from a different layer in each (we call this **across nets**).

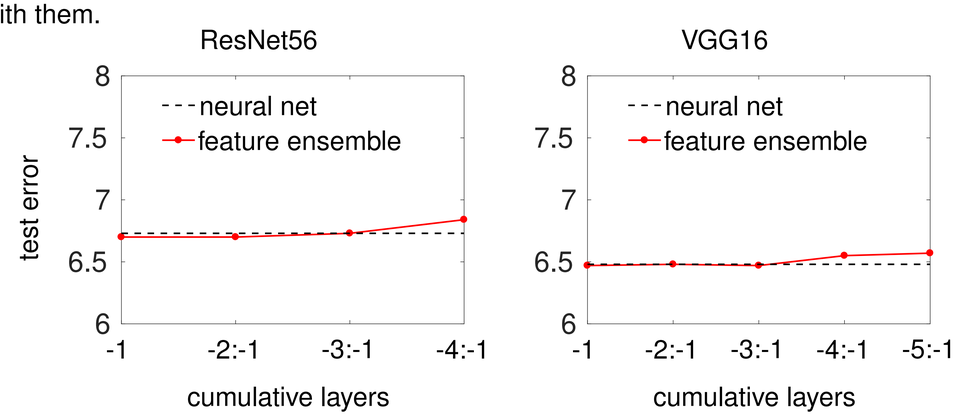
input training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$; initial tree $\mathbf{T}(\cdot;\Theta)$ of depth Δ
and with parameters $\Theta = \{\boldsymbol{\theta}_i\}$, where $\boldsymbol{\theta}_i$ each node parameters
$\mathcal{N}_0, \dots, \mathcal{N}_\Delta \leftarrow$ nodes at depth $0, \dots, \Delta$, respectively
repeat
  for $d = 0$ **to** Δ
    parfor $i \in \mathcal{N}_d$
      if i is a leaf **then**
        $\boldsymbol{\theta}_i \leftarrow$ train a classifier on the training point that reach leaf i
      else
        compute the "best" child for each training points that reach node i
          and set it as a pseudolabel (call this modified training set $\mathcal{R}_i$)
        $\boldsymbol{\theta}_i \leftarrow$ train a linear binary classifier on $\mathcal{R}_i$
until stop
return T

Why TAO trees? they produce trees and forests with high accuracy; they are very fast (at training and inference); and they do feature selection (useful when input dimension is high).

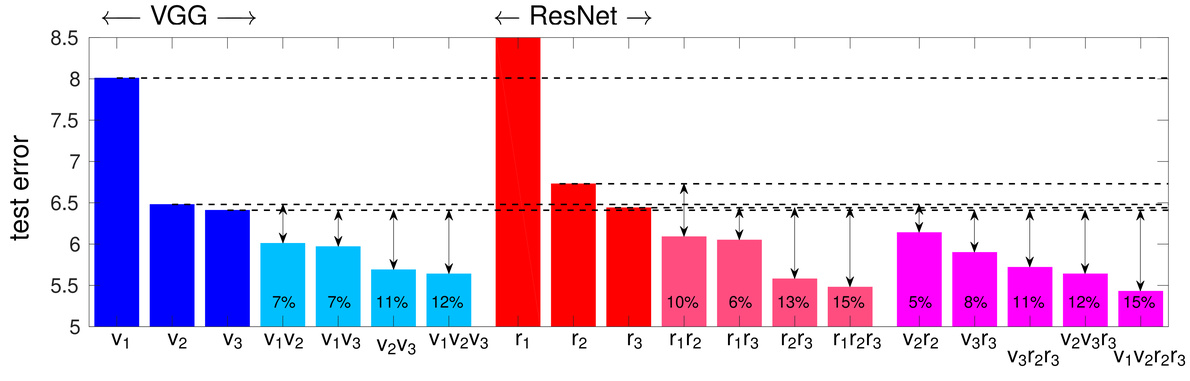
We provide generic pseudocode for the TAO algorithm in the left figure. TAO repeatedly alternates between optimizing over a subset of nodes and fixing the remaining ones. The optimization itself is done by training a binary classifier in the decision nodes and a K class classifier (const, linear, etc.) in the leaves.

3 Experiments

- **Within net** : below figures show our results of ensembling the last i layers (excluding softmax layer) of the same neural net on CIFAR-10. So, "$-i$:-1" contains the concatenated features from the last i layers. We clearly observe that, as we add intermediate features to the penultimate-layer features, the test error decreases very little then increases slightly (likely because the tree is overfitting). This is a **negative** result, but it suggests that the features in the penultimate layer are sufficient for optimal classification, and that the features in previous layers are correlated or redundant with them.



- **Across nets** : below figure shows our results of ensembling penultimate layers (before the softmax output) of different nets on CIFAR-10: v_1, v_2, v_3 stand for VGG11 / 13 / 16; and r_1, r_2, r_3 for ResNet20 / 56 / 110, respectively. We clearly see that any combination improves over all its members. The improvement is remarkable in that we achieve large gains with few members (2 to 4) in the ensemble: the relative improvement over the best member net is between 5% and 15%. This holds when combining nets of different size, architecture, or both. The more members (of different type), the lower the error.



Work supported by NSF award IIS–2007147