Hierarchical Classifier

intern@mmlab, Sep.13

Abstract

1). Explore Dataset **2).** Ablation study on Baseline model. **3).** A unified method combining coarse-grained and fine-graied classification task.

Contents

1. Related Work

1.1. Hierarchical Softmax Classifier (Mikolov et al., 2013)

The large number of classes of ImageNet, *i.e.* 22K categories, can be expensive to train, if use basic full softmax classifier. w_O need to travel through all possible classes, *e.g.* 10^7 words in dictinary for NLP task.

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

The idea of hierarchical softmax is **1).** view 20K classes as a tree generated by Huffuman Coding(If just want to reduce training cost. The cost is $O(L(w_O))$). **2).** train a multi-label classifier at each node on the tree.

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^\top v_{w_I} \right)$$

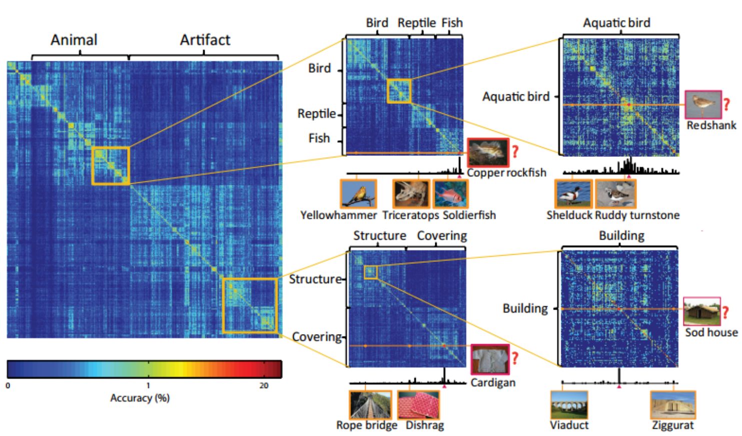
But it is different with our work:

- The learned representation is flatten. The dillema between different granularity still exists.
- If trained on ImageNet 21K (A more pratical case), we can use *Flattened Hit Rate@Top k* or other metric to evaluate it. Given a new species 'Leopard-tiger' it can only say 'It is animial $\succ$ leopard $\succ$ tiger', but it can not say 'It is animal, to be specific, may be leopard or tiger'.

1.2. Multiple Granularity Descriptors (Wang et al., 2015)

1.3. ImageNet (Deng et al., 2010)

- Flattened Classifier: Train on ImageNet7K, equvalent to out flattened baseline. The hierarchical structure of classes is learned.



- Measure Grainuity of Dataset: $h(i,j)$ between categories i and j, as the height of their lowest common ancestor. Grainuity = $\text{mean}_{i,j\in\mathcal{C}}\{h(i,j)\}$.

- Loss for Hierarchical Dataset: 'redshank' is a 'bird', classiy it as 'bird' is less informative, but at least right. Rather than using 0-1 Loss, (Deng et al., 2010) use Hierarchical Loss. $L(x) = \Sigma_{j=1}^{K} C_{i,j} p_j(x)$, where $C_{i,j} = h(i,j)$.

1.4. Structure prediction

2. ImageNet

- ImageNet22K: Whole Dataset. 22K cateories from the Fall 2011 release of ImageNet, only including leaf categories.
- ImageNet10K(Deng et al., 2010): 10184 categories from the Fall 2009 release of ImageNet, including both internal and leaf nodes with more than 200 images in each category.
- ImageNet7K: 7404 leaf categories from ImageNet10K. Only leaf.
- ImageNet1K(ILSCV 2010 – 2012 Classfication Task): 1000 leaf categories randomly sampled from ImageNet7K.

2.1. Observation:

- Approximately Tree Structure

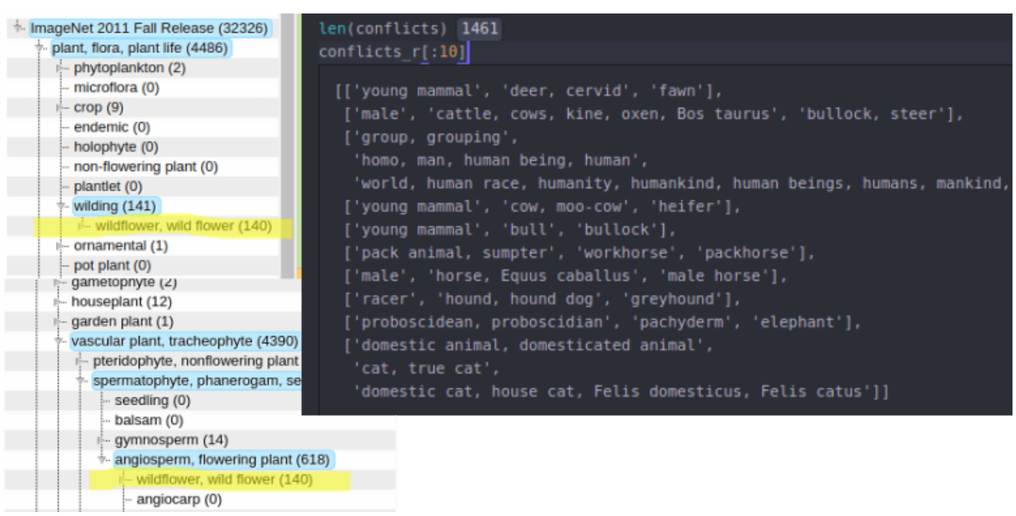


Figure 1: A problem in ImageNet: It is true that fawn is both yong manmal and dear,thus wordnet is a DAG. But the conflict is few only 1461, compared to 82K[2], so can be approximately viewed as a tree.

We will use DFS to force it to become a tree.

- Analyze Hierarchical Stucture of Classes

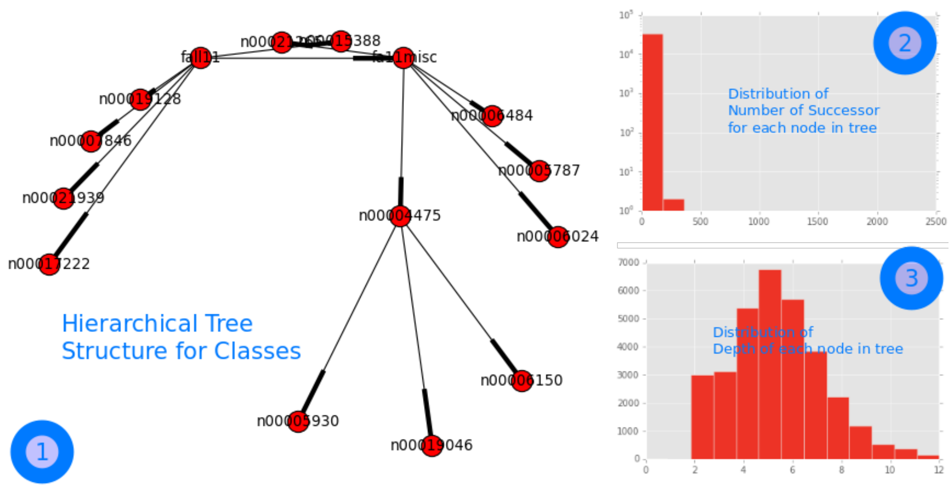


Figure 2: Note that in **2).**, the Y-axis is log-scaled

- Log-tail Distribution of Number of Instance in each Class

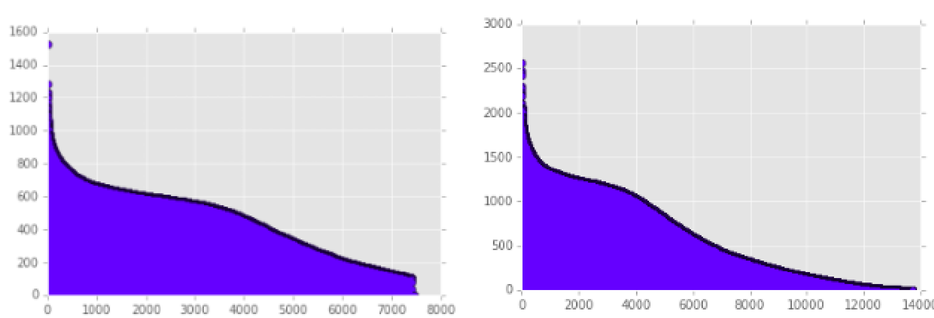


Figure 3: **Left:** Number of instance in each class for classes in leaves of ImageNet10K(At least 200 images per classes), **Right:** For classes in ImageNet22K(Partially downloaded). update.

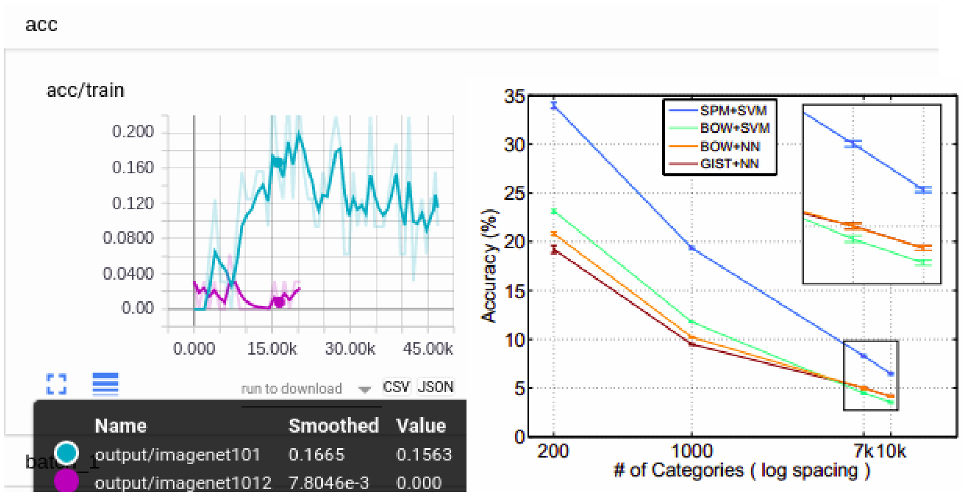
[2] 82K contains all classes from coarse to fine, 22K-ImageNet refer to the most fine-graied classes(leave node) is 22K.

- We will use ImageNet10K for our experiments
  - Categories are over-lapping bwteen internal-node and leaves.
  - Contains ImageNet1K, where pretrained model trained from. But we may ignore this problem.
  - The benefit is number of images in each class is more than 200, although training data still unbalanced.
- No standard train/valid/test split for the task
  - Some people use 50/50, which is a bit crazy, but it is the convention of the PASCAL VOC Challenge, and (Deng et al., 2010) use this split protocol.
  - Some use 90/10, which can be quite different. It is split protocol for ILSVR Competition.

3. Training on ImageNet10k

After removing internal node, ImageNet10k contains 7461 categories. Thus in fact we are using ImageNet7k proposed by (Deng et al., 2010) but we can still give prediction for all classes in ontology tree for ImageNet10k. Train baseline on this dataset:

- Resnet101 → 7x7x1024 → global ave pooling → 1024 → FC → 7461
  - Step 020660 - Loss 9.43 - acc 0.00% (0.37sec/step)
- Resnet101 → 7x7x1024 → conv → 2x2x1024 → Flatten + FC → 7461
  - Step 042420 - Loss: 5.86 acc: 34.38% (0.870 sec/step)
  - Step 045160 - Loss: 5.48 acc: 3.12% (0.697 sec/step)
- Training expense still comes from I/O, rather than normalization computation in softmax layer.
- Due to imbalance image number/classes in one batch, training accuracy is fluctuating. The test accuracy(avarage on each instance) reported in (Deng et al., 2010) is 7%. Our train avaraging accuracy is 16%.



Left: Learning curve of our model, **Right:** Accuracy reported in (Deng et al., 2010)

4. Train on Cifar100

The Structure of Cifar100(20 super classes, 100 fine classes) is similar to our setting, except that instance in each class is balance(100 imgs/class) and batch size $256 > 100$.

Note that for multiple loss experiements, we will reorder the label for each class, the class label for semantically similar class will be similar, *e.g.* mapping $4, 30, 55, 72, 95$ to $0, 1, 2, 3, 4$. Using this class label, mapping from coarse label to fine label can be calculated directly rather than using HashTable.

```
In [17]: cifar100.mapping_human

Out[17]: {'aquatic_mammals': {'beaver', 'dolphin', 'otter', 'seal', 'whale'},
          'fish': {'aquarium_fish', 'flatfish', 'ray', 'shark', 'trout'},
          'flowers': {'orchid', 'poppy', 'rose', 'sunflower', 'tulip'},
          'food containers': {'bottle', 'bowl', 'can', 'cup', 'plate'},
In [18]: cifar100.mapping

Out[18]: {0: {4, 30, 55, 72, 95},
          1: {1, 32, 67, 73, 91},
          2: {54, 62, 70, 82, 92},
          3: {9, 10, 16, 28, 61},
```

- How to train fast:

- Finetune = Initialize from pretrained model + Only train last parameter block
- But observe overfitting if learning rate is large for a long time. High val acc on cifar100 can be 75.72%, but we will first use this model to iterate ideas.

- Combine multiple loss

Given the feature of a sample $\mathbf{x}_i$, whose ground truth is $\mathbf{y}_i = c_i$, *i.e.* $y_i^{c_i}$, we want to know the propabobility of each possible category y_i to form a prediction vector for this sample $\mathbf{P}(\mathbf{y}_i|\mathbf{x}_i) = [P(y_i^{c_0}|\mathbf{x}_i), \cdots, P(y_i^{c_n}|\mathbf{x}_i)]^T$.

If we have semantic prior from wordnet tree, then

$$
\begin{aligned}
P({y_i}^{c_i}|\mathbf{x}_i) &= \Sigma_{c_k \in Par(c_i)} P({y_i}^{c_i}|y_i^{c_k}, \mathbf{x}_i) P(y_i^{c_k}|\mathbf{x}_i) \\
&= P({y_i}^{c_i}|y_i^{Par(c_i)}, \mathbf{x}_i) P(y_i^{Par(c_i)}|\mathbf{x}_i)
\end{aligned}
$$

E.g. , let $\mathbf{y}_i = c_i = tiger$, $Par(c_i) = carnivore$ and $Par(c_j) = herbivore$, then $P({y_i}^{c_i}|y_i^{Par(c_j)}, \mathbf{x}_i) = 0$, since we infer $y^{c_i} \perp y^{Par(c_i)}|\mathbf{x}_i$ from wordnet tree.

Hierachy softmax force wordnet tree into *binary* tree using hierachuy clustering (Morin and Bengio, 2005), due to computation cost consideration. We aim to fully utilize the semantic prior: **1).** use geometry structure between classes to boost performance, **2).** use taxonomy tree to handle unbalanced training examples.

Cross entroy loss is widely used in multi-class classification. Let $\mathbf{f} = [f_0, \cdots, f_n]^T$ be the logits, then cross entroy loss is defined as $L_i = -\log(\frac{e^{f_{y_i}}}{\Sigma_j e^{f_j}})$. Based on it, We design a hierachical loss incorprating semantic prior from wordnet.

To begin with a simple case, we consider cifar100, Ref. to Fig. 4:

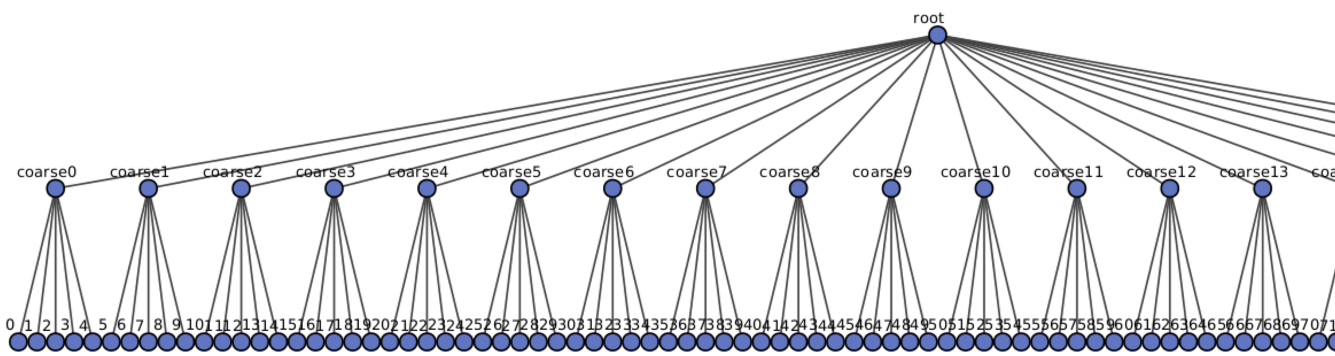


Figure 4: Coarse and Fine Categories of Cifar100

Assume we have Function $\textsc{Par}(node, depth)$, Ref. to Alg. 1 and $\textsc{Ch}(node, depth)$, Ref. to Alg. 2 and we define $\textsc{Find} = \text{lambda } y_i, start, end : \textsc{Ch}(\textsc{Par}(y_i, start), end)$.

$$
\begin{aligned}
L_i^{100} &= -\log\left(\frac{\Sigma_{j \in \textsc{Find}(y_i,2,2)} e^{f_j}}{\Sigma_{j \in \textsc{Find}(y_i,0,2)} e^{f_j}}\right) \\
L_i^{20} &= -\log\left(\frac{\Sigma_{j \in \textsc{Find}(y_i,1,2)} e^{f_j}}{\Sigma_{j \in \textsc{Find}(y_i,0,2)} e^{f_j}}\right) \\
L_i^{group} &= -\log\left(\frac{\Sigma_{j \in \textsc{Find}(y_i,2,2)} e^{f_j}}{\Sigma_{j \in \textsc{Find}(y_i,1,2)} e^{f_j}}\right)
\end{aligned}
$$

Algorithm 1 Find Parents of *node* at *depth*

```
function Par(node,depth)
% Input: node: the ground truth label to query;
%        depth: parent node's depth.
% Output: parent node at depth
    if node.depth ≥ depth then
        return node
    else
        return Par(node.parent,depth)
```

Algorithm 2 Find Child Nodes of *root* at *depth*

```
function Ch(root,depth)
% Input: root: the root node to query;
%        depth: child nodes' depth.
% Output: list of child nodes at depth
    ChildNodes ← list()
    for node in root.childs do
        if node.depth ≥ depth then
            ChildNodes.extend(list([node]))
        else
            ChildNodes.extend(Ch(node, depth))
    return ChildNodes
```


Deng, J., Berg, A. C., Li, K., Fei-Fei, L., 2010. What does classifying more than 10,000 image categories tell us? In: European conference on computer vision. Springer, pp. 71–84.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119.

Morin, F., Bengio, Y., 2005. Hierarchical probabilistic neural network language model. In: Aistats. Vol. 5. pp. 246–252.

Wang, D., Shen, Z., Shao, J., Zhang, W., Xue, X., Zhang, Z., 2015. Multiple granularity descriptors for fine-grained categorization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2399–2406.