# Deep Metric Learning Using Hierarchical Loss Function

Sethu Hareesh Kolluru (hareesh@stanford.edu)
Department of Computer Science, Stanford University (video link:`https://youtu.be/4n3gaqxHMg0`)

## Introduction

**Background and Motivation:** Deep metric learning techniques aim at learning semantic distance measures and embeddings such that similar input images are mapped to nearby points on a manifold in embedding space and dissimilar images are mapped apart from each other.
However, most of the work in this field treated all classes as equally important as in a "flat" structure. But in fact, in most datasets, the classes can be arranged in a hierarchical structure where a form of general-to-specific category ordering often exists between classes. So the motivation for this work is can the hierarchical structure of these classes be effectively used to learn a better embedding and improve the similarity measure.
**Problem Statement:** To build and study neural architectures and learn an embedding that can encode the hierarchical category level semantic information of images(or products) such that nearest neighbours in embedding space represents the products that are visually similar as well as in the category too.

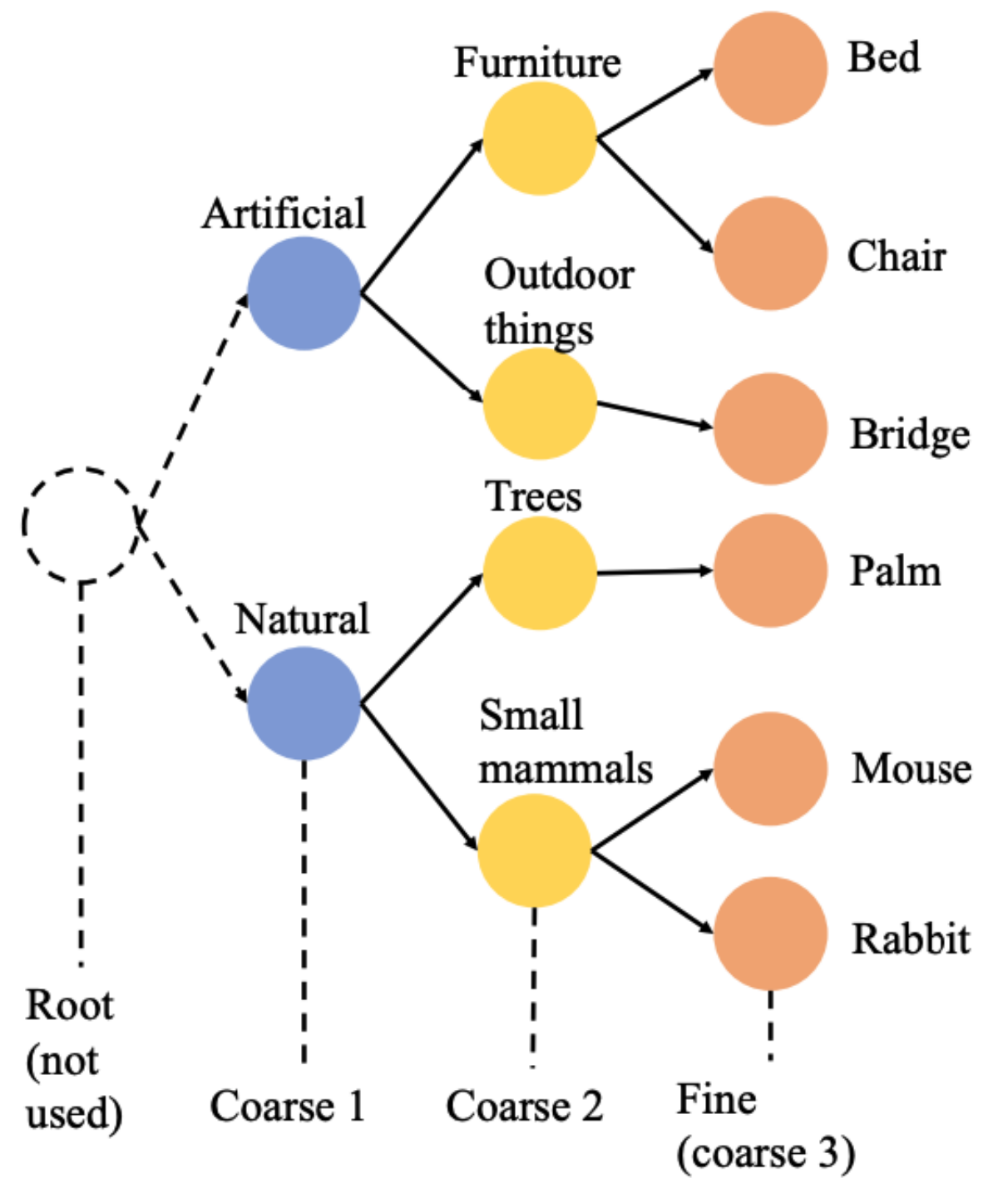
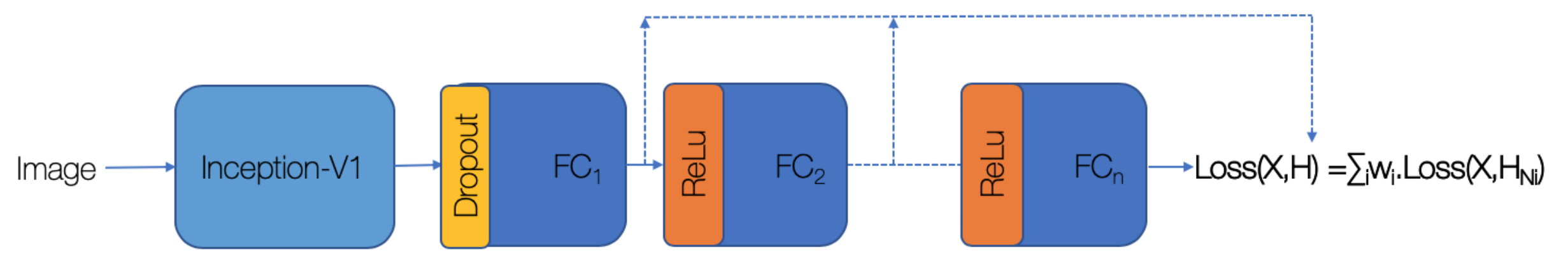


**Figure 1:** A sample hierarchical label tree[1]

## Approach

### General Framework:

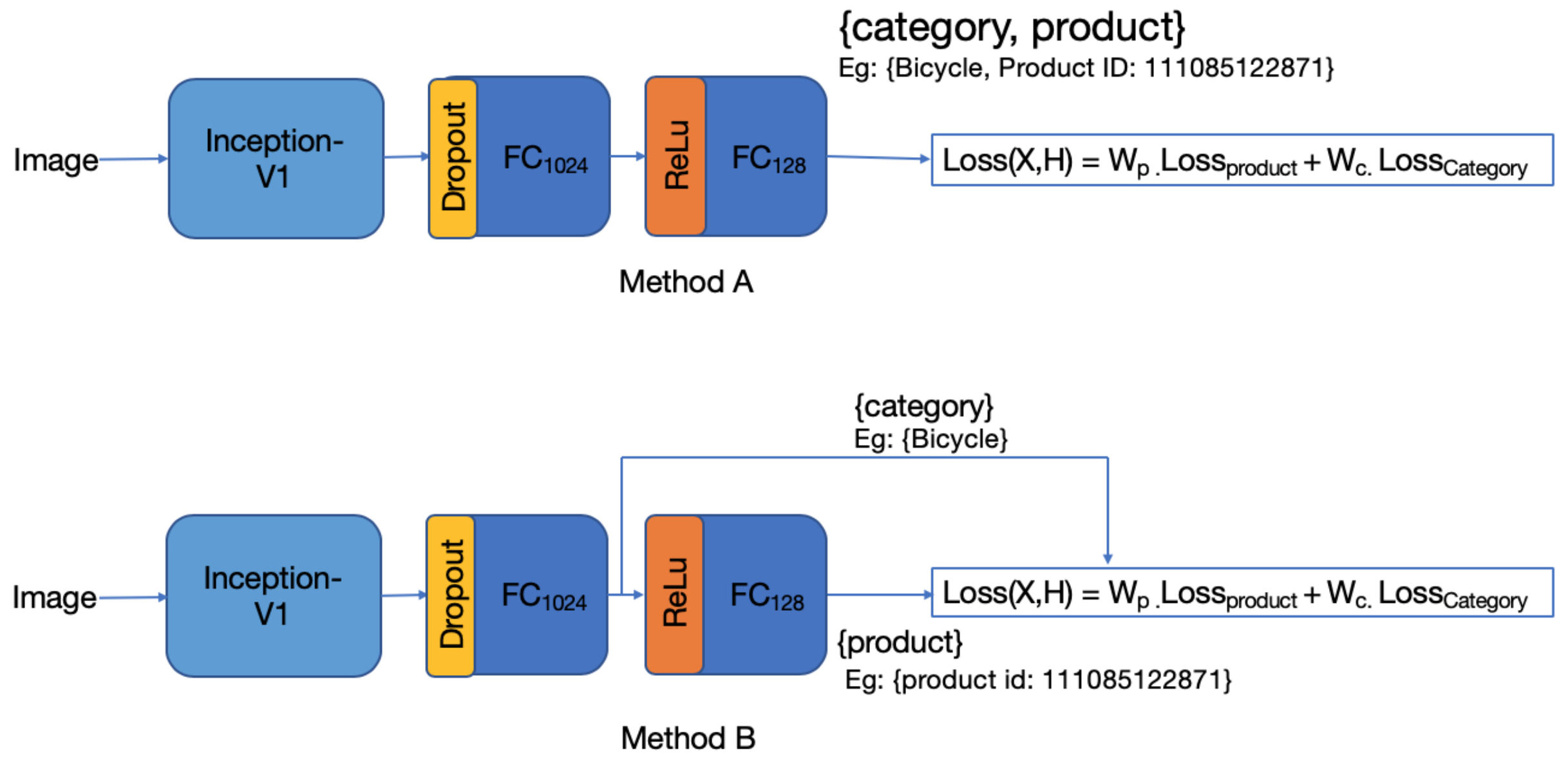
An $n$-node taxonomy, $N_1, N_2, N_3, ...N_n$, associated with the given image is used as a class label to train a network with a flexibility to select multiple embedding either from a series of fully connected (FC) layers or just using the last FC layer.



### Lifted Structure Loss[2]:

$$L_{i,j} \quad = \quad log\left(\sum_{(i,k)\epsilon N} e^{(\alpha-D_{i,k})} + \sum_{(i,l)\epsilon N} e^{(\alpha-D_{j,l})}\right) + D_{i,j}$$

**Network Architecture:** To encode a two-level hierarchical label tree, we explore the following methods in this work.


{category, product}
Eg: {Bicycle, Product ID: 111085122871}
Image
Inception-V1
Dropout
FC$_{1024}$
ReLu
FC$_{128}$
Loss(X,H) = W$_p$ .Loss$_{product}$ + W$_c$. Loss$_{Category}$
Method A
{category}
Eg: {Bicycle}
Image
Inception-V1
Dropout
FC$_{1024}$
ReLu
FC$_{128}$
Loss(X,H) = W$_p$ .Loss$_{product}$ + W$_c$. Loss$_{Category}$
{product}
Eg: {product id: 111085122871}
Method B

## Experiments

**Setup:**

- Implemented in TensorFlow.
- Adam Optimizer with $10^{-5}$ learning rate.
- Parameter for Baseline set as in [2]
- Product Loss Weight, $w_p = 0.8$
- Category Loss Weight, $w_c = 0.2$
- Product Margin, $\alpha_p = 1$
- Category Margin, $\alpha_c = \{5, 1, 0.2\}$

**Datasets:**

- Stanford Online Products (SOP)
  Products: total/train/test = 22634/11318/11316
- Deep Fashion: In-Shop Retrieval
  Products: total/train/test = 11735/5812/5923

**Evaluation Metrics:**

- Retrieval: Recall@K(K=1,4,8,10)
- Clustering: F1, NMI

## Results and Discussion

**Retrieval: Stanford Online Products**

| | Recall@1 | | | Recall@10 | | |
|---|---|---|---|---|---|---|
| Method | Product@1 | Category@1 | Delta | Product@10 | Category@10 | Delta |
| Baseline | 0.66 | 0.86 | 0.2 | 0.83 | 0.98 | 0.15 |
| Method A | 0.57 | 0.83 | 0.26 | 0.77 | 0.95 | 0.18 |
| Method B | 0.57 | 0.86 | 0.29 | 0.76 | 0.96 | 0.2 |

**Retrieval: Deep Fashion -In shop**

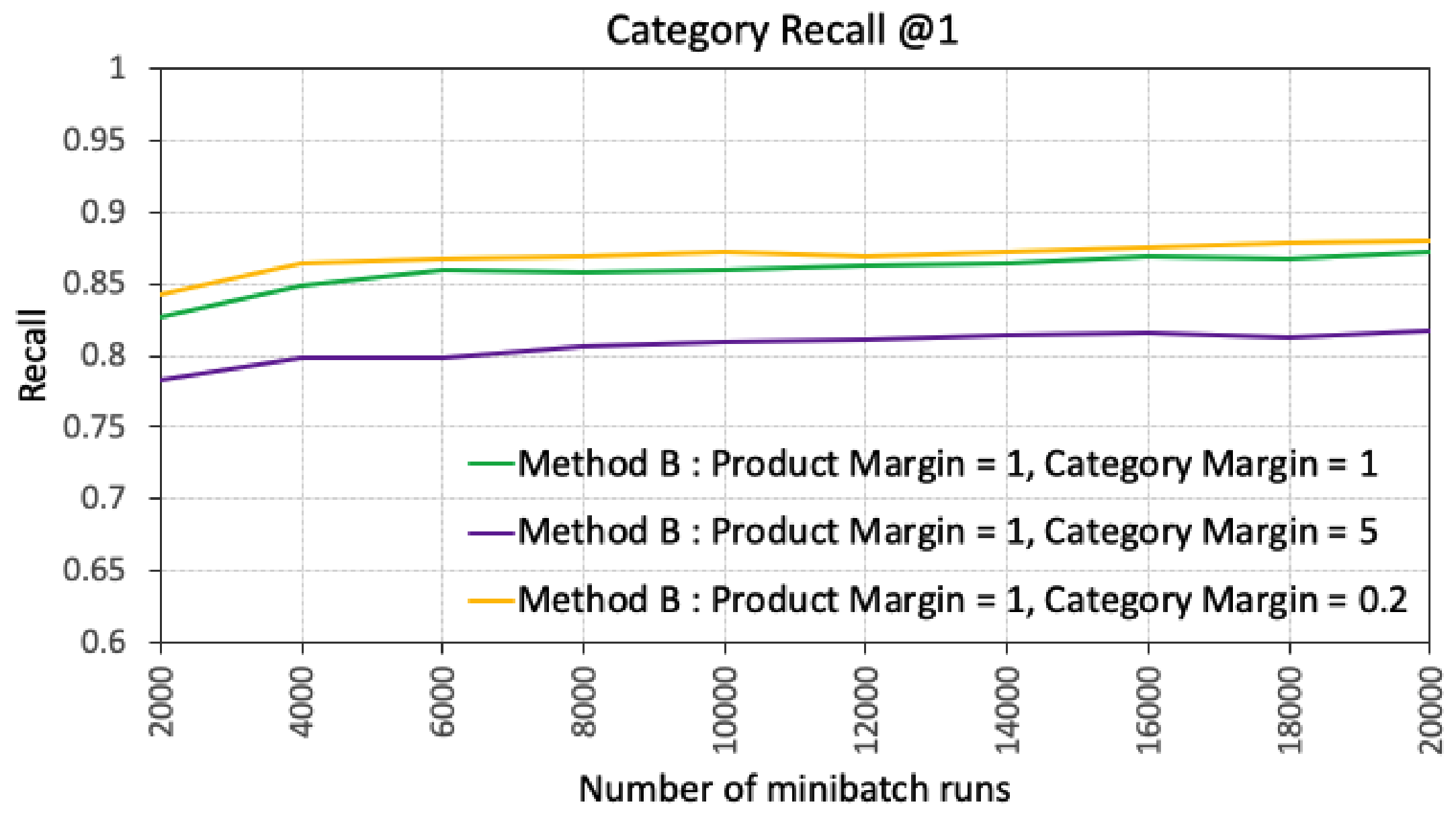
| | Recall@1 | | | Recall@10 | | |
|---|---|---|---|---|---|---|
| Method | Product@1 | Category@1 | Delta | Product@10 | Category@10 | Delta |
| Baseline | 0.85 | 0.92 | 0.07 | 0.97 | 0.98 | 0.01 |
| Method A | 0.72 | 0.80 | 0.08 | 0.93 | 0.93 | 0 |
| Method B | 0.68 | 0.87 | 0.19 | 0.89 | 0.95 | 0.06 |

**Clustering: Stanford Online Products**

| | F1 | | NMI | |
|---|---|---|---|---|
| Method | Product | Category | Product | Category |
| Baseline | 0.33 | 0.30 | 0.89 | 0.34 |
| Method A | 0.26 | 0.63 | 0.88 | 0.63 |
| Method B | 0.26 | 0.64 | 0.87 | 0.64 |

**Effect of Category Margin:** Setting product margin to 1 and category margin to 0.2 encodes the fact that the network should try to get the category correct within stricter margin confinements of 0.2 leaving a relative easier goal for product margin to work with. This enables the network to learn effectively as getting categories nearer in embedding space deals with learning relatively coarse features whereas getting products within strict margin requires learning fine-grained features and hence results in a better category recall.

**Analysis:** The difference between category recall and product recall represents the percentage of the queries, where the nearest neighbor is not from the same product but from the same category as the query image. Both retrieval and clustering results show the improvement in products from the same category clustering together in both approaches - Method B and Method A.



## Conclusion

A neural architecture that can effectively encode prior knowledge of the target data and output multiple predictions based on multiple embeddings on a concatenated set of layers, each corresponding to a level in the hierarchical structure of the data is implemented to showcase that such information can be adopted to boost performance.

## References

[1] Xinqi Zhu and Michael Bain. B-cnn: branch convolutional neural network for hierarchical classification. *arXiv preprint arXiv:1709.09890*, 2017.

[2] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016.

[3] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–285, 2018.