Hierarchical Image Classification with Conceptual Hierarchies Generated via Lexical Databases

Tomoaki Yamazaki*[1], Seiya Ito*[1], and Kouzou Ohara[1]

Aoyama Gakuin University, Japan
d5620003@aoyama.jp, {s.ito,ohara}it.aoyama.ac.jp
*Equal Contribution

Abstract. Hierarchical image classification is the task of classifying images using hierarchical information. Conventional methods use hierarchies only during training and classify the fine classes in the hierarchy. In this paper, we focus on the hierarchical information utilizing not only during training but also during inference and evaluation. To verify the effectiveness of hierarchical information, we propose a simple architecture that explicitly utilizes it during training and inference. We also introduce hierarchy-based evaluation metrics. The key idea of designing the architecture is to obtain outputs for all nodes of a given hierarchy and integrate them to predict hierarchical classes. To this end, we explore neural network architectures that do not require a hierarchy tailored to the network. In our evaluation metrics, to analyze the characteristic of the hierarchical image classification model in more detail, superior concepts on the hierarchy of fine classes as well as fine classes are subject to evaluation. Furthermore, we present a new method for generating concept hierarchies via a lexical database. We empirically verify the effectiveness of the method that combines the proposed network architecture with the generated concept hierarchy.

Keywords: image classification · hierarchical classification · lexical database · conceptual hierarchy.

1 Introduction

Thanks to deep learning, image classification models have made great strides over the past decade. In general, an image classification model consists of a feature extractor that extracts features from an image and a classifier that recognizes the class as one of the predefined classes. As image features are important for classification, a variety of convolutional neural network- (CNN) [6, 13] and Transformer-based [3,9] architectures have been proposed for feature extractors. In contrast, simple architectures (e.g., one linear layer) are often used for the classifier. In such a classifier, the classes to be recognized are treated as flat, making it difficult to discriminate among classes with similar image features.

Besides, the similarity of image features does not correspond to semantic similarity. Specifically, even the state-of-the-art method sometimes classifies "couch" as "bed", even though "chair" and "couch" are semantically close.

To prevent such false classification, several methods have been proposed that utilize the hierarchical structure of the classes [5,16]. However, since most methods only use hierarchies for training, it is not guaranteed that predictions are performed based on hierarchical information. Moreover, their concept hierarchies are manually constructed to fit the classification models. From this perspective, two questions arise: 1) Is it appropriate not to use hierarchical information during inference? 2) Is it adequate to adjust the conceptual hierarchy according to the classification model architecture?

To address these questions, we propose a network that is simple yet capable of handling complex hierarchies. The key idea is to obtain outputs for nodes of a hierarchy (hierarchical classes) from image features and integrate them along the hierarchy to predict classes. The proposed integration scheme depends only on the number of nodes in the hierarchy. Unlike previous hierarchical classification methods, we do not need to adjust the hierarchy to the model architecture. Thus, we also propose a method to generate a concept hierarchy for classification via a lexical database based on class labels. Since the concept hierarchy in the lexical database contains a lot of information unrelated to classes, we construct a compact concept hierarchy by eliminating unimportant nodes.

We evaluate the proposed method with respect to how it predicts classes along the hierarchy as well as general classification accuracy. The proposed network is capable of predicting classes other than the fine classes as final predictions. In practical situations, it is sometimes more valuable to predict classes at a coarse level that is not wrong than to fail to classify classes at a fine level. For example, there are cases in which it is sufficient to recognize a "chair" or "couch" as a "seat" without distinguishing between them or to recognize "furniture" without even distinguishing between seats and tables. We define metrics focusing on the concepts to which the predictive and correct answer classes belong, and report on the effectiveness of the proposed method.

To summarize, the contributions of this paper are threefold:

- We tackle the task of classifying images based on a conceptual hierarchy and propose a network that can handle complex hierarchical structures.
- We propose a novel method for constructing compact concept hierarchies for hierarchical image classification from fine-grained class labels by referencing a lexical database.
- We present an evaluation method for image classification based on a conceptual hierarchy and show the results of a detailed analysis of hierarchical image classification.

2 Related Work

2.1 Hierarchical Image Classification

Hierarchical image classification is an image classification task in which hierarchical information related to the classes is given in addition to the image [16]. Hierarchy information is generally defined in a tree structure with classes as nodes and can be utilized to improve image classification accuracy with the same classification model architecture. The leaf nodes of the tree have a high level of detail and are called fine classes, while the other nodes are called course classes. For example, CIFAR-100 [8], a commonly used benchmark for hierarchical classification, has 20 coarse (super) classes, and each class is associated with five fine classes (i.e., the course class "people" has five fine classes: "baby", "boy", "girl", "man", and "woman"), for a total of 100 fine classes. However, many datasets do not contain hierarchical information, so hierarchical information needs to be manually created in order for hierarchical image classification [16].

Hierarchical image classification models predict coarse as well as fine classes for each image. A typical model is based on the idea that features are represented differently depending on the depth of the layers (blocks) of the CNN and predicts the class of each hierarchy using the features of the blocks according to the depth of the hierarchy. Branch Convolutional Neural Network (B-CNN) [16], one of the representative methods, employs VGG-16 [13] as a feature extractor and divides it into three blocks to predict three hierarchical classes. B-CNN uses separate classifiers to identify each level of hierarchy and improves fine class accuracy by auxiliary use of course class classification loss. However, B-CNN is limited in the hierarchical structure it can handle because it is necessary to determine the number of feature extractor blocks according to the hierarchical structure. To seek hierarchy in accordance with model architecture, some hierarchy generation algorithms such as group assignment [7,12] and hierarchy k-means [4] have been proposed. However, these algorithms do not generate a semantically interpretable tree structure like WordNet [10].

In the literature, the concept closest to our approach was proposed by Guo et al. [5]. They presented a network architecture called CNN-RNN that combines a CNN and a Recurrent Neural Network (RNN). In CNN-RNN, the RNN takes the features of the final layer of the CNN as input and predicts all classes in the hierarchy at each time step once. While CNN-RNN can handle arbitrary hierarchies, it ignores hierarchical structures during prediction. In this paper, we explore methods that can handle arbitrary hierarchies in prediction.

2.2 Hierarchy Generation

Generating hierarchy from a lexical database such as WordNet [10] consists of the following steps [14,15]:

1. Select words from the collection.
2. Get one hypernym path from each selected word.

3. Build the tree from hypernym paths.
4. Compress the tree by eliminating a parent that has fewer than a specific number of children.

Following the above procedure, we first select words with the appropriate concepts from the class labels in the dataset. In the second step, because the most lexical database is not the tree structure, [14] adopts a heuristic approach that selects the first sense of hypernyms, but we often struggle to determine whether hypernym is appropriate for the dataset. Since the concepts selected in the second step influence the subsequent process of tree construction and compression, it is difficult to choose one hypernym without looking at the tree carefully. In this paper, we determine the importance of nodes based on the tree and eliminate nodes with less importance while updating the tree.

3 Methodology

This study aims to design a hierarchical classifier that can handle arbitrary hierarchies. If the final prediction is not restricted to fine classes but includes hierarchical classes, there are multiple possible ways to predict classes based on hierarchy. To this end, we explore hierarchical classifier architectures. Note that the proposed architecture consists of an arbitrary image feature extractor and a hierarchical classifier as in general image classification models, so a fixed-dimensional feature vector obtained by a feature extractor is an input to the hierarchical classifier.

In this section, we first explain how to construct a concept hierarchy from a lexical database in Sec. 3.1 and then the weighting of hierarchical classes for hierarchical classification in Sec. 3.2. We then discuss the model designs and loss function for any hierarchy in Sec. 3.3.

3.1 Conceptual Hierarchy Generation

We propose a new hierarchy generation algorithm that determines a parent (hypernym) by its importance while comparing it to a tree in generating process. To determine the parent of a node, the importance of a node n is defined as the number of paths N_n^{root} that go through a node n from the leaves to the root (referred to as "root paths"). To avoid deleting important nodes coincidentally, all nodes are sorted by the number of root paths N_n^{root}. Moreover, each time the tree is significantly updated, we recalculate the number of root paths N_n^{root}. The proposed algorithm is as follows:

1. Select a concept (synset) for each class in the dataset.
2. Build the tree, calculate the number of root paths N_n^{root} for each node n, and initialize the target number of root paths r_t as 1.
3. Keep iterating until a r_t is no longer the path of the root.
   (a) Get target nodes whose root path is equal to r_t and sort the leaves of the tree in descending order of depth.

(b) Compress the tree by eliminating a parent that has fewer than k children recursively and select a parent for each target node with many root paths.
(c) Recalculate the number of root paths N_n^{root} for every node.
(d) Update r_t as $r_t + 1$ if there is no change in the tree through processing the target nodes.

The first step is to manually assign synsets to the dataset if no synsets are assigned. The second step is the initialization that builds the tree from hypernym paths of selected synsets and calculates the root path of all nodes. The root paths of leaves are always one, but in the initialization, the root path of the root is not always the number of classes because some nodes have multiple parents. In the third step, it keeps updating the tree until the target root path r_t reaches the root path of the root. In other words, there is no change in the tree when checking all nodes. In the recursive process, we first obtain target nodes with less important nodes of unprocessed nodes and sort their order by the depth in descending order. Second, we eliminate a parent that has fewer than k children unless the parent is the root. This process is repeated as long as the importance of a parent is the same as that of the target node. At this time, if the node has multiple parents, we select a parent by the number of importance. If parents have the same importance, we select a parent with deeper depth and a smaller synset ID in a lexical database. In image classification, a fine class is not necessarily a tree leaf, as in the case of a chair being considered a piece of furniture. To account for this, if the parent of a fine class is another fine class, it is aligned horizontally and linked to the descendants of the fine class. After updating the tree by processing the target nodes, all root paths are recalculated. If there are no updates, the target number of root paths is incremented so that the root path of the root equals the number of classes eventually.

3.2 Weights for Hierarchical Classes

In a hierarchical structure, all parents inherited from a certain fine class are positive for that fine class (e.g., furniture for a chair). In contrast, siblings of a certain fine class and classes around inherited parents are negative for the fine class (e.g., a desk whose parent is also furniture, but the desk is not a chair).

Based on this idea, we assign weighted positive labels and negative labels to every fine class. Specifically, the positive weights for the parents of the fine class are obtained by a function that takes the distance from the fine class as input. Meanwhile, we compute negative weights for the siblings of a parent, i.e., the children of that parent, by a function with positive weights and the number of children as input. The overall algorithm is shown in Algorithm 1.

In this algorithm, it is necessary to define functions to compute positive and negative weights. Intuitively, the deeper the depth in hierarchical classification, the harder it is to discriminate. For this reason, we define a function that increases the weights with depth:

$$weight_{(+)}(d) = \frac{1}{1 + \log(d+1)} \tag{1}$$

Algorithm 1 Hierarchical class weighting

Input: $\mathcal{N}$: nodes that have one parent node and multiple child nodes, n_r: a root node ($n_r \in \mathcal{N}$), n_f: a fine node ($n_f \in \mathcal{N}$)
Output: $\mathbf{w}^f_{(+)}$, $\mathbf{w}^f_{(-)}$: positive and negative weights

Initialize positive and negative weights except for the root:
$\mathbf{w}^f_{(+)} \leftarrow zeros(|\mathcal{N}| - 1)$, $\mathbf{w}^f_{(-)} \leftarrow zeros(|\mathcal{N}| - 1)$
Initialize target node and depth: $t \leftarrow n_f$, $d \leftarrow 0$
while t is not n_r **do**
  Calculate positive weights: $\mathbf{w}^f_{(+)}(t) \leftarrow weight_{(+)}(d)$
  for $c \in children(parent(t)) \setminus t$ **do**
    Calculate negative weights:
    $\mathbf{w}^f_{(-)}(c) \leftarrow weight_{(-)}(weight_{(+)}(d), N_t^{children})$
  end for
  $t \leftarrow parent(t)$
  $d \leftarrow d + 1$
end while

Designing a function with negative weights, we note that it may interfere with learning if the weights are larger than the positive weights at the same depth or even their parents' positive weights. Therefore, we calculate the negative weights such that they are less than the positive weights with respect to the positive weights at the same depth:

$$weight_{(-)}(w_{(+)}, s) = w_{(+)}/s \tag{2}$$

where s is a specific number.

3.3 Network Architecture

For cases where the final prediction target is not limited to fine-grained classes but includes hierarchical classes, there are three classifiers for predicting classes based on hierarchy. The first is to predict all hierarchical classes flat as in [5]; the second is to follow the hierarchy top-down; and the third is to follow the hierarchy bottom-up. We hereafter describe how to predict classes using hierarchies. Note that the dimension of the final output is the number of nodes excluding the root (i.e., the total number of hierarchical classes) for all networks.

All-at-Once Classifier The first is an all-at-once classifier, which uniformly predicts all hierarchical classes in a given hierarchy. Let x and f^{all} represent the input image and network, respectively. The probability distribution over hierarchical classes $\mathbf{p}$ are calculated as follow:

$$\mathbf{p} = \sigma(f^{all}(x)) \tag{3}$$

where σ is the softmax function.

Algorithm 2 Top-Down classifier

Input: x: an input image, $\mathcal{N}$: nodes, n_r: a root node, f^{td}: a network
Output: $\mathbf{p}$: probability distribution over hierarchical classes
1: $\mathbf{w} \leftarrow f^{td}(x)$
2: Initialize target node and depth: $q \leftarrow queue()$, $q.push(n_r)$
3: **while** q is not empty **do**
4: $\quad t \leftarrow q.pop()$
5: $\quad$ **for** $c \in children(t)$ **do**
6: $\quad\quad \mathbf{w}(c) \leftarrow \mathbf{w}(c) + \mathbf{w}(t) * N_c^{root} \ / \ N_t^{root}$
7: $\quad\quad q.push(c)$
8: $\quad$ **end for**
9: **end while**
10: $\mathbf{p} \leftarrow \sigma(\mathbf{w})$

Top-Down Classifier The second is a top-down classifier, which predicts hierarchical classes by adding the output of the higher level nodes to the lower level nodes. Similar to the all-at-once classifier, the top-down classifier outputs initial scores for all hierarchical classes from image features but updates the scores following Algorithm 2.

Bottom-Up Classifier The third is a bottom-up classifier, which predicts hierarchical classes by adding the output of the lower level nodes to the higher level nodes. This is almost the reverse process of the top-down classifier, as detailed in Algorithm 3. Since a general classifier that predicts only detailed classes can be viewed as a bottom-up classifier, we will show the results of applying Algorithm 3 to a general classifier in our experiments.

Loss Function Using outputs of the hierarchical classifier $\mathbf{p}$, positive weights $\mathbf{w}_{(+)}$, and negative weights $\mathbf{w}_{(-)}$, the loss function is defined as the sum of the positive and negative errors:

$$\mathcal{L} = \mathcal{L}_{(+)} + \mathcal{L}_{(-)} \tag{4}$$

where

$$\mathcal{L}_{(+)} = -\log(\mathbf{p}) * \mathbf{w}_{(+)} \tag{5}$$
$$\mathcal{L}_{(-)} = -\log(1-\mathbf{p}) * \mathbf{w}_{(-)} \tag{6}$$

4 Experiments

4.1 Implementation Details

Dataset We evaluate the proposed method on CIFAR-10 and CIFAR-100 [8]. Both datasets consist of 60,000 images of size 32×32, split into 50,000 for training

Algorithm 3 Bottom-up classifier

Input: x: an input image, $\mathcal{N}$: nodes, n_r: a root node, f^{bu}: a network
Output: $\mathbf{p}$: probability distribution over hierarchical classes

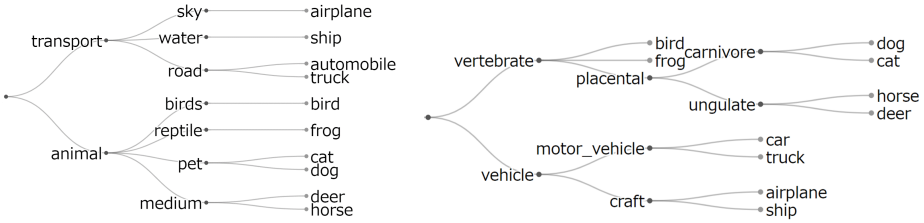
$\mathbf{w} \leftarrow f^{bu}(x)$
Initialize target nodes and depth: $q \leftarrow priority\_queue()$
for n in $\mathcal{N}$ **do**
  if $is\_fine\_class(n)$ **then**
    $q.push((-depth(n), n))$
  end if
end for
while q is not empty **do**
  $(d, t) \leftarrow q.pop()$
  $\mathbf{w}(parent(t)) \leftarrow \mathbf{w}(parent(t)) + \mathbf{w}(t)$
  if t is not n_r **then**
    $q.push((-d + 1, parent(t)))$
  end if
end while
$\mathbf{p} \leftarrow \sigma(\mathbf{w})$

and 10,000 for testing. CIFAR-10 contains 10 classes, including bird, dog, and car, whereas CIFAR-100 has 100 classes, including more fine-grained classes than CIFAR-10. For data augmentation, we apply zero-padding by 4 pixels, random crop of size 32×32, horizontal flip with 50% probability, and random rotation with the range of $[-15°, 15°]$.

For hierarchical image classification, we prepare three hierarchies for each dataset: one is defined by [16] (manual), and the others are generated by our method described in Sec. 3.1. We basically use WordNet [10] as the lexical database, and for unknown concepts, we refer to BabelNet [11], which is automatically constructed by integrating WordNet and Wikipedia's lexical and encyclopedic knowledge, as an auxiliary database. For example, in CIFAR-100, the class "aquarium fish" has no counterpart in WordNet. Thus, we referenced BabelNet to obtain a higher level concept for "aquarium fish", i.e., "fish"[1]. In the hierarchy generation process, we set the hyperparameter k for tree compression, which represents the number of children of a node. In our experiments, k was set to 1 and $\{2, 3\}$ for CIFAR-10 and CIFAR-100, respectively. The generated hierarchies are illustrated in Fig. 1 and Fig. 2.

Training In our experiments, we employ VGG-16 [13] with batch normalization (BN) as an image feature extractor and initialize with pretrained weights by ImageNet [1]. We follow the hyperparamter settings as in [2]. We train all models with 200 epochs with a mini-batch size of 128. We use SGD optimizer with momentum, and the initial learning rate, momentum, and weight decay are set

[1] In fact, the higher level word for "aquarium fish" on BabelNet is fish with the ID "bn:13520875n", but "bn:13520875n" did not have a link to WordNet. Therefore, we replaced "bn:13520875n" with "bn:00034816n" manually.



(a) CIFAR-10 Manual (b) CIFAR-10 by the proposed method ($k = 1$)

Fig. 1: Conceptual hierarchies on CIFAR-10. (a) Manually constructed by [16] on CIFAR-10. (b) Generated by our method with $k = 1$ on CIFAR-10.

to 0.1, $5e - 4$, and 0.9, respectively. The learning rate is decayed by 0.2 at 60, 120, and 160 epochs.

4.2 Evaluation Metrics

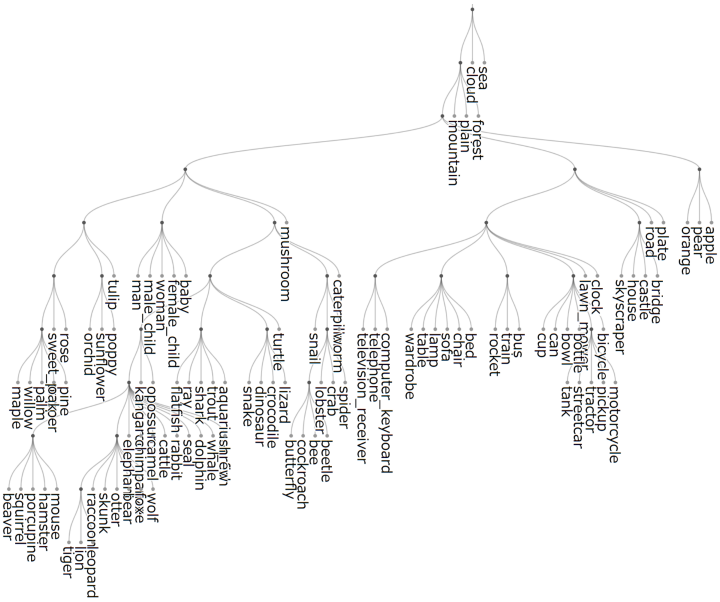
Fine Accuracy Since hierarchical image classification in this study does not necessarily classify into fine classes, we refer to the accuracy in general image classification as fine accuracy (FA). FA is the percentage of matches between the predicted class and the correct class when the predicted class is restricted to one of the fine classes.

Common Concept Ratio When restricting the prediction class to the fine classes, we evaluate whether the prediction class and the correct answer class contain common concepts. In a given conceptual hierarchy, we define the common concept ratio (CCR) as the percentage of cases where at least one node other than the root node is present between the path from the root to the predicted class and the path from the root to the correct class. This is an upper bound for prediction at the appropriate hierarchy and is never less than FA.

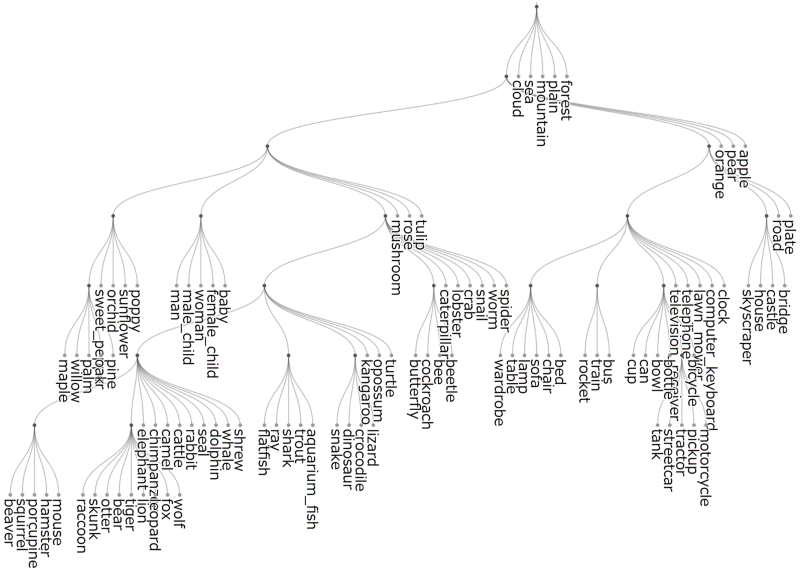
Depth-Weighted Hierarchical Classification Accuracy For practical use, it is necessary to determine at which hierarchy to predict hierarchical classes, but the way to do this is nontrivial. To measure how accurately the predictions follow the hierarchy, we define depth-weighted hierarchical classification accuracy (DWHCA):

$$DWHCA(\hat{\mathcal{Y}}, \mathcal{Y}) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{\hat{\mathcal{Y}}_i \cap \mathcal{Y}_i}{|\hat{\mathcal{Y}}_i|} \tag{7}$$

where $\mathcal{I}$ is an evaluation dataset consisting of images and labels, $\hat{\mathcal{Y}}$ is a list that has a set of predicted labels, $\mathcal{Y}$ is a list that has a set of the target fine class and whose hypernyms. This metric is inspired by multilabel classification and implies precision weighted by the depth of the hierarchy.



(a) CIFAR-100 by the proposed method ($k = 2$)



(b) CIFAR-100 by the proposed method ($k = 3$)

Fig. 2: Conceptual hierarchies on CIFAR-100. (a) and (b) are generated by our method with $k = 2, 3$ on CIFAR-100, respectively.

Table 1: Fine accuracy on CIFAR-10 and CIFAR-100.

Method	CIFAR-10	CIFAR-100
VGG-16 [13]	87.96	63.04
VGG-16 w/ BN* [13]	92.94	72.59
B-CNN [16]	88.22	64.42
CF-CNN [12]	-	65.11
Ours (manual)	93.04	**72.86**
Ours	**93.16**	72.81

*Our implementation

4.3 Comparison to Previous Methods

In Table 1, we compare the proposed method with our baseline models [13] and previous methods that use hierarchies [12,16] with FA. In the proposed method, we show the results of the same manual hierarchy as the B-CNN [16] and the best of the hierarchies generated by the proposed method. While our goal is to predict classes following a given hierarchy, our method achieved results comparable to or better than the baseline in both cases. This suggests that the hierarchy does not need to be constructed manually.

4.4 Ablation Study

We design three hierarchical classifiers and weightings for the hierarchical classes. We conduct ablation experiments on these and report the hierarchical classification performance as shown in Table 2. In Table 2, we denote all-at-once, top-down, and bottom-up classifiers as ALL, TD, and BU, respectively. In the experiment, for each node in the path from the fine class predicted by the hierarchical classifier to the root, the CCR and DWHCA are calculated by determining the most likely hierarchical class among it and its siblings.

The proposed method produces results comparable to or better than the baseline for all metrics. In particular, the method that applies hierarchical class weighting to the bottom-up hierarchical classifier is better overall. However, it is observed in all-at-once and top-down classifiers that the negative weights in the proposed hierarchical label weighting degrade several metrics, meaning that it follows the hierarchy less well than the positive weights alone. This is because negative weights are computed based on a bottom-up scheme, which is helpful for bottom-up classifiers but not compatible with the others.

5 Conclusion

We have explored network architectures for hierarchical image classification along conceptual hierarchies. Unlike previous work, we do not need to construct hierarchies manually, and our hierarchical classifier, which uses the hierarchies

Table 2: Ablation study on network architecture and hierarchical class weighting using CIFAR-100.

	manual			Ours ($k = 2$)			Ours ($k = 3$)		
Model	FA	CCR	DWHCA	FA	CCR	DWHCA	FA	CCR	DWHCA
VGG-16 w/ BN	72.59	88.60	81.11	72.59	98.83	86.79	72.59	97.02	84.87
Ours-ALL	72.45	89.79	81.61	**72.65**	98.67	87.20	**72.81**	96.74	85.26
Ours-ALL-w/neg	**72.86**	89.87	**81.80**	72.57	98.61	87.08	72.54	96.65	85.20
Ours-TD	72.62	89.89	81.76	72.58	98.64	87.14	72.58	96.64	85.21
Ours-TD-w/neg	72.52	89.52	81.49	72.32	98.69	86.98	72.52	96.72	85.15
Ours-BU	72.23	89.19	81.37	72.16	98.88	86.86	72.45	97.09	85.05
Ours-BU-w/neg	72.77	**89.97**	81.68	72.23	**98.90**	**87.77**	72.77	**97.17**	**85.62**

obtained by the proposed hierarchy generation method, achieved results comparable to or even superior to manually constructed hierarchies. Extensive experiments with various backbones and datasets are left for future work.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number JP22K17978, and a part of this work was conducted as a project of JST SPRING Grant Number JPMJSP2103.

References

1. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255 (2009)
2. Devries, T., Taylor, G.W.: Improved Regularization of Convolutional Neural Networks with Cutout. arXiv preprint arXiv:1708.04552 (2017)
3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: Proceedings of the 9th International Conference on Learning Representations (ICLR) (2021)
4. Guo, Y., Xu, M., Li, J., Ni, B., Zhu, X., Sun, Z., Xu, Y.: HCSC: Hierarchical Contrastive Selective Coding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9696–9705 (2022)
5. Guo, Y., Liu, Y., Bakker, E.M., Guo, Y., Lew, M.S.: CNN-RNN: a large-scale hierarchical image classification framework. Multimedia Tools and Applications **77**(8), 10251–10271 (2018)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)

7. Kim, J., Park, Y., Kim, G., Hwang, S.J.: SplitNet: Learning to Semantically Split Deep Networks for Parameter Reduction and Model Parallelization. In: Proceedings of the 34th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 70, pp. 1866–1874 (2017)
8. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. Tech. rep. (2009)
9. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9992–10002 (2021)
10. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM **38**(11), 39–41 (1995)
11. Navigli, R., Ponzetto, S.P.: BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence **193**, 217–250 (2012)
12. Park, J., Kim, H., Paik, J.: CF-CNN: Coarse-to-Fine Convolutional Neural Network. Applied Sciences **11**(8), 3722 (2021)
13. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Bengio, Y., LeCun, Y. (eds.) Proceedings of the 3rd International Conference on Learning Representations (ICLR) (2015)
14. Stoica, E., Hearst, M.A.: Nearly-Automated Metadata Hierarchy Creation. In: Proceedings of HLT-NAACL 2004: Short Papers. pp. 117–120. Boston, Massachusetts, USA (2004)
15. Stoica, E., Hearst, M.A., Richardson, M.: Automating Creation of Hierarchical Faceted Metadata Structures. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL). pp. 244–251 (2007)
16. Zhu, X., Bain, M.: B-CNN: Branch Convolutional Neural Network for Hierarchical Classification. arXiv preprint arXiv:1709.09890 (2017)