Integrating domain knowledge: using hierarchies to improve deep classifiers

Clemens-Alexander Brust[1] and Joachim Denzler[1,2]
[1]Computer Vision Group, Friedrich Schiller University Jena
[2]Michael Stifel Center Jena
Jena, Germany
{clemens-alexander.brust,joachim.denzler}@uni-jena.de

Abstract

One of the most prominent problems in machine learning in the age of deep learning is the availability of sufficiently large annotated datasets. While for standard problem domains (ImageNet classification), appropriate datasets exist, for specific domains, e.g. *classification of animal species, a long-tail distribution means that some classes are observed and annotated insufficiently. Challenges like iNaturalist show that there is a strong interest in species recognition. Acquiring additional labels can be prohibitively expensive. First, since domain experts need to be involved, and second, because acquisition of new data might be costly. Although there exist methods for data augmentation, which not always lead to better performance of the classifier, there is more additional information available that is to the best of our knowledge not exploited accordingly.*

In this paper, we propose to make use of existing class hierarchies like WordNet to integrate additional domain knowledge into classification. We encode the properties of such a class hierarchy into a probabilistic model. From there, we derive a special label encoding together with a corresponding loss function. Using a convolutional neural network, on the ImageNet and NABirds datasets our method offers a relative improvement of 10.4% *and* 9.6% *in accuracy over the baseline respectively. After less than a third of training time, it is already able to match the baseline's fine-grained recognition performance. Both results show that our suggested method is efficient and effective.*

1. Introduction

In recent years, convolutional neural networks (CNNs) have achieved outstanding performance in a variety of machine learning tasks, especially in computer vision, such as image classification [10, 4] and semantic segmentation [12]. Training a CNN from scratch in an end-to-end fashion not only requires considerable computational resources and experience, but also large amounts of labeled training data [20]. Using pre-trained CNN features [18], adapting existing CNNs to new tasks [6] or performing data augmentation can reduce the need for labeled training data, but may not always be applicable or effective.

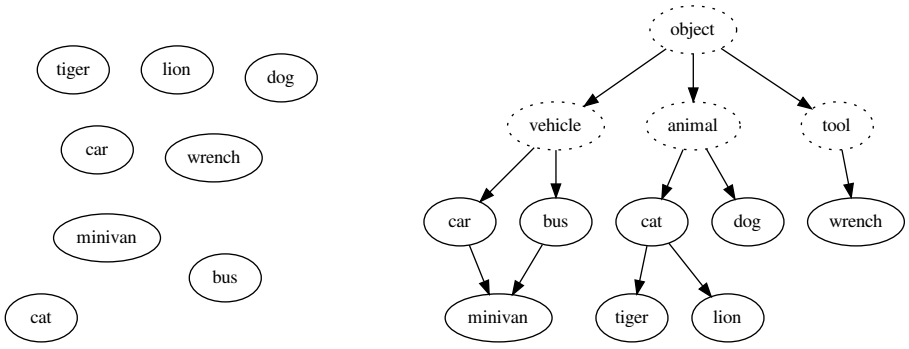
For specific problem domains, *e.g.* with a long-tailed distribution, the amount of labeled training data available is not always sufficient for training a CNN. When unlabeled data already exists, which is not always the case, active learning [17] to select valuable instances for labeling may be applied. However, labels still have to be procured which is not always feasible because of the cost of domain experts.

Besides information from more training data, domain knowledge in the form of high-level information about the structure of the problem can be considered. In contrast to annotations of training data, this kind of domain knowledge is already available in many cases from projects like iNaturalist [23], Visual Genome [8], Wikidata [25] and WikiSpecies[1].

In this paper, we use class hierarchies, *e.g.* WordNet [3], as an example of domain knowledge. In contrast to approaches based on attributes, where annotations are often expected to be per-image, class hierarchies offer the option of domain knowledge integration on the highest level with the least additional annotation effort. We encode the properties of such a class hierarchy into a probabilistic model that is based on common assumptions around hierarchies. From there, we derive a special label encoding together with a corresponding loss function. These components are applied to a CNN and evaluated systematically. The construction could also be interpreted as a special case of learning using privileged information (LUPI, [24]).

Our main **contributions** are: (i) a deep learning method based on a probabilistic model to improve existing classifiers by adding a class hierarchy which (ii) works with any form of hierarchy representable using a directed acyclic graph (DAG), *i.e.* does not require a tree hierarchy. We evaluate our method in experiments on the CIFAR-100 [9], ImageNet and NABirds [22] datasets to represent problem do-

[1] https://species.wikimedia.org/wiki/Main_Page



(a) Class Set (b) Class Hierarchy

Figure 1. Comparison between a loose set of independent classes and a class hierarchy detailing inter-class relations.

mains of various scales.

1.1. Related Work

In the following, we describe how our work relates to existing work in the field of hierarchical classification, where hierarchical data may be encountered and how it relates to knowledge transfer.

Hierarchical Classification The authors of [19] offer an extensive survey and a generalized view on hierarchical classification. Most hierarchical classifiers can be categorized under their proposed notation. Criteria include (i) how many classes may be predicted at once, (ii) whether non-leaf classes can be predicted, (iii) if the underlying hierarchy is a tree or a directed acyclic graph (DAG) and (iv) how the classifier's architecture maps to the hierarchy. A categorization scheme is also proposed for the hierarchical classification problems themselves, where they are grouped by (i) tree or DAG structure, (ii) how many classes are labeled per sample and (iii) whether the labels are always as specific as possible.

Considering this categorization, our method has the following properties: (i) it can predict many classes at once, but is limited to one for our experiments, (ii) it can predict non-leaf classes (see Section 3.6) if desired, (iii) it supports DAGs as underlying hierarchies, not just trees, and (iv) it does not fit any of the described policies (see Section 2.4). The problems tackled in our experiments are (i) defined by a DAG structure, (ii) have one class label per sample and (iii) the labels are always as specific as possible.

Hierarchical Data Typical image classification datasets rarely offer hierarchical information. There are exceptions such as the iNaturalist challenge dataset [23] where a class hierarchy is derived from biological taxonomy. Exceptions also include specific hierarchical classification benchmarks, *e.g.* [14, 21] as well as datasets where the labels originate from a hierarchy such as ImageNet [1]. The Visual Genome dataset [8] is another notable exception, with available metadata including attributes, relationships, visual question answers, bounding boxes and more, all mapped to elements from WordNet.

To augment existing non-hierarchical datasets, class hierarchies can be used. For a typical object classification scenario, concepts from the WordNet database [3] can be mapped to object classes. WordNet contains nouns, verbs and adjectives that are grouped into *synsets* of synonymous concepts. Relations such as hyponymy (**is-a**), antonymy (**is-not**), troponymy (**is-a-way-of**) and meronymy (**is-part-of**) are encoded in a graph structure where synsets are represented by nodes and relations by edges respectively. In this paper, we use the hyponymy relation to infer a class hierarchy. A hyponym is a specific instance of a more general concept, whereas a hyperonym is a more abstract notion.

Knowledge Transfer Adding domain knowledge in the form of a class hierarchy can be seen as a special case of knowledge transfer, where multiple tasks are learned simultaneously, sharing knowledge between tasks to ultimately improve results, possibly because there are too few training examples for individual tasks.

Several methods of knowledge transfer between object classes aimed at scalability towards large numbers of classes are presented in [16]. The authors note that while knowledge transfer does not generally improve classification in settings where training data is available for all classes, it is valuable in zero-shot learning scenarios [13], where some classes do not have any labeled training exam-

ples. One of their methods performs knowledge transfer based on the WordNet hierarchy underlying the ImageNet challenge dataset they use. In a zero-shot setting, it outperforms other methods based on part attributes and semantic similarity.

In [15], a method for learning from few examples using Gaussian processes is presented. Knowledge is transferred between related object categories to improve performance when only very few training examples are available. The relationships between categories are extracted from the WordNet hierarchy [3] to guide the knowledge transfer. The authors show improvements using their method compared to learning the categories individually.

2. Hierarchical Classification with Deep Classifiers

In this section, we propose a method to adapt existing classifiers to hierarchical classification. We start by acquiring a hierarchy and then define a probabilistic model based on it. From this probabilistic model, we derive an encoding and a loss function that can be used in a machine learning environment.

2.1. Class Hierarchy

For our model, we assume that a hierarchy of object categories is supplied, *e.g.* from a database such as WordNet[3] or WikiSpecies. It is modeled in the form of a graph $W = (S,h)$, where S denotes the set of all possible object categories, called *synsets* in the WordNet terminology. These are the nodes of the graph. Note that S is typically a superset of the dataset categories $C \subseteq S$, since parent categories are included to connect existing categories, *e.g.* `vehicle` is a parent of `car` and `bus`, but not originally part of the dataset.

A hyponymy relation $h \in S \times S$ over the classes, which can be interpreted as directed edges in the graph, is also given. For example, $(s,s') \in h$ means that s' is a hypernym of s, or s is a hyponym of s', meaning s **is-a** s'. In general, the **is-a** relation is transitive. However, WordNet only models direct relationships between classes to keep the graph manageable and to represent different levels of abstraction as graph distances. The relation is also irreflexive and asymmetric.

For the following section, we assume that W is a directed acyclic graph (DAG). However, the WordNet graph is commonly reduced to a tree, for example by using a voting algorithm [21] or selecting task-specific subsets that are trees [1]. The supplementary material outlines such a procedure. In this paper, we work on the DAG directly with no modifications needed.

2.2. Probabilistic Model

Elements of a class hierarchy are not always mutually exclusive, *e.g.* a `corgi` is also a `dog` and an `animal` at the same time. Hence, we do not model the class label as one categorical random variable, but assume multiple independent Bernoulli variables $Y_s, s \in S$ instead. Formally, we model the probability of any class s occurring independently (and thus allowing even multi-label scenarios), given an example x:

$$P(Y_s = 1|X = x), \tag{1}$$

or, more concisely,

$$P(Y_s^+|X). \tag{2}$$

The aforementioned model on its own is overly flexible considering the problem at hand, since it allows for any combination of categories co-occurring. At this point, assumptions are similar to those behind a one-hot encoding. However, from the common definition of a hierarchy, we can infer a few additional properties to restrict the model.

Hierarchical decomposition A class s can have many independent parents $S' = s'_1,\ldots,s'_n$. We choose $Y^+_{S'}$ to denote an observation of at least one parent and $Y^-_{S'}$ to indicate that no parent class has been observed:

$$\begin{aligned} Y^+_{S'} &\Leftrightarrow Y^+_{s'_1} \vee \ldots \vee Y^+_{s'_n} \Leftrightarrow Y_{s'_1} = 1 \vee \ldots \vee Y_{s'_n} = 1, \\ Y^-_{S'} &\Leftrightarrow Y^-_{s'_1} \wedge \ldots \wedge Y^-_{s'_n} \Leftrightarrow Y_{s'_1} = 0 \wedge \ldots \wedge Y_{s'_n} = 0. \end{aligned}$$

Based on observations $Y_{S'}$, we can decompose the model from Equation (2) in a way to capture the hierarchical nature. We start by assuming a marginalization of the conditional part of the model over the parents $Y_{s'}$:

$$\begin{aligned} P(Y_s^+|X) = P(Y_s^+|X,Y^+_{S'})P(Y^+_{S'}|X) \\ + P(Y_s^+|X,Y^-_{S'})P(Y^-_{S'}|X). \end{aligned} \tag{3}$$

The details of this decomposition are given in the supplementary material.

Simplification We now constrain the model and add assumptions to better reflect the hierarchical problem. If none of the parents $S' = s'_1,\ldots,s'_n$ of a class s occur, we assume the probability of s being observed for any given example to be zero:

$$P(Y_s^+|X,Y^-_{S'}) = P(Y_s^+|Y^-_{S'}) = 0. \tag{4}$$

This leads to a simpler hierarchical model, omitting the second half of Equation (3) by setting it to zero:

$$P(Y_s^+|X) = P(Y_s^+|X,Y^+_{S'})P(Y^+_{S'}|X). \tag{5}$$

Parental independence To make use of recursion in our model, we require the random variables $Y_{s'_1}, \ldots, Y_{s'_n}$ to be independent of each other in a naive fashion. Using the definition of $Y^+_{S'}$, we derive:

$$P(Y^+_{S'}|X) = 1 - \prod_{i=1}^{|S'|} 1 - P(Y^+_{s'_i}|X). \tag{6}$$

Parentlessness In a non-empty DAG, we can expect there to be at least one node with no incoming edges, *i.e.* a class with no parents. In the case of WordNet, there is exactly one node with no parents, the root synset `entity.n.01`. A marginalization over parent classes does not apply there. We assume that all observed classes are children of `entity` and thus set the probability to one for a class without parents:

$$P(Y^+_s|X,S' = \emptyset) = 1. \tag{7}$$

Note that this is not reasonable for all hierarchical classification problems. If the hierarchy is composed of many disjoint components, $P(Y^+_s|X,S' = \emptyset)$ should be modeled explicitly. Even if there is only a single root, explicit modeling could be used for tasks such as novelty detection.

2.3. Inference

The following section describes the details of the inference process in our model.

Restricted Model Outputs Depending on the setting, when the model is used for inference, the possible outputs can be restricted to the classes C that can actually occur in the dataset as opposed to all modeled classes S including parents that exist only in the hierarchy. This assumes a fixed class set at test time and does not apply to open-set problems. We denote this setting *mandatory labeled node prediction (MLNP)*. The unrestricted alternative is named *arbitrary node prediction (ANP)*.

Prediction To predict a *single* class s given a specific example x, we look for the class where the joint probability of the following observations is high: (i) the class s itself occurring (Y^+_s) and (ii) none of the children $S'' = s''_1, \ldots, s''_m$ occurring ($Y^-_{S''}$):

$$s(x) = \underset{s \in C \subseteq S}{\operatorname{argmax}} P(Y^+_s|X) P(Y^-_{S''}|X,Y^+_s). \tag{8}$$

Requiring the children to be pairwise independent similar to Equation (6), inference is performed in the following way:

$$s(x) = \underset{s \in C \subseteq S}{\operatorname{argmax}} P(Y^+_s|X) \prod_{i=1}^{|S''|} 1 - P(Y^+_{s''_i}|X,Y^+_s). \tag{9}$$

Because $P(Y^+_s|X)$ can be decomposed according to Equation (3) and expressed as a product (cf. Equation (6)), we infer using:

$$\begin{aligned} s(x) = \underset{s \in C \subseteq S}{\operatorname{argmax}} & P(Y^+_s|X, Y^+_{S'}) \\ & \cdot \underbrace{(1 - \prod_{i=1}^{|S'|} 1 - P(Y^+_{s'_i}|X))}_{\text{Parent nodes } S'} \\ & \cdot \underbrace{\prod_{i=1}^{|S''|} 1 - P(Y^+_{s''_i}|X,Y^+_s)}_{\text{Child nodes } 'S} . \end{aligned} \tag{10}$$

Again, $P(Y^+_{s'_i}|X)$ can be decomposed. This decomposition is performed recursively following the scheme laid out in Equation (3) until a parentless node is reached, where the probability is assumed to be one (cf. Equation (7)) or modeled explicitly.

2.4. Training

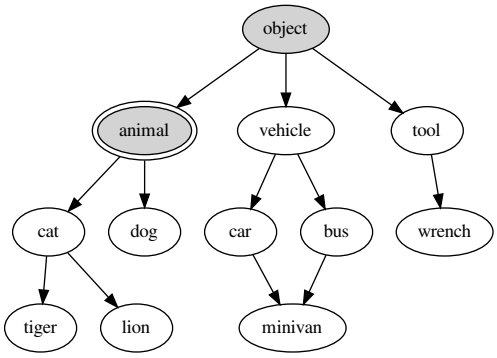
In this section, we describe how to implement our proposed model in a machine learning context. Instead of modeling the probabilities $P(Y^+_s|X)$ for each class s directly, we want to estimate the conditional probabilities $P(Y^+_s|X, Y^+_{S'})$. This changes each individual estimator's task slightly, because it only needs to discriminate among siblings and not all classes. It also enables the implementation of the hierarchical recursive inference used in Equation (10).

The main components comprise of a label encoding $e : S \to \{0,1\}^{|S|}$ as well as a special loss function. A label $y \in S$ is encoded using the hyponymy relation $h \in S \times S$, specifically its transitive closure $\mathcal{T}(h)$, and the following function:

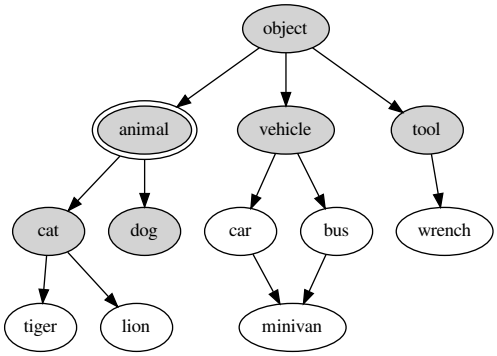
$$e(y)_s = \begin{cases} 1 & \text{if } y = s \text{ or } (y,s) \in \mathcal{T}(h), \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

A machine learning method can now be used to estimate encoded labels directly. However, a suitable loss function needs to be provided such that the conditional nature of each individual estimator is preserved. This means that, given a label y, a component s should be trained only if one of its parents s' is related to the label y by $\mathcal{T}(h)$, or if y is one of its parents. We encode this requirement using a *loss mask* $m : S \to \{0,1\}^{|S|}$, defined by the following equation:

$$m(y)_s = \begin{cases} 1 & y = s \text{ or} \\ & \exists (s,s') \in h : y = s' \text{ or } (y,s') \in \mathcal{T}(h), \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$



(a) Encoding $e(y)$



(b) Loss Mask $m(y)$

Figure 2. Hierarchical encoding and loss mask for $y = \texttt{animal}$. Shaded nodes represent 1 and light nodes 0 respectively.

Figure 2 visualizes the encoding $e(y)$ and the corresponding loss mask $m(y)$ for a small example hierarchy. Using the encoding and loss mask, the complete loss function $\mathcal{L}$ for a given data point (x,y) and estimator $f : X \to \{0,1\}^{|S|}$ is then defined by:

$$\mathcal{L}_f(x,y) = m(y)^T(e(y) - f(x))^2. \tag{13}$$

The function $f(x)_s$ is then used to estimate the conditional probabilities $P(Y_s^+|X, Y_{S'}^+)$. Applying the inference procedure in Section 2.3, a prediction is made using the formula in Equation (10) and substituting $f(x)_s$ for $P(Y_s^+|X, Y_{S'}^+)$.

3. Experiments and Evaluation

The following section describes the setup of our experiments as well as the evaluation protocol. We aim to assess the effects of applying our method on three different scales of problems using datasets described in Section 3.1. We also validate the requirement of restricting the predictions to a known set of possible classes (see Section 2.3) as opposed to allowing any element within the hierarchy.

3.1. Datasets

CIFAR-100 For our experiments, we want to work with a dataset that does not directly supply hierarchical labels, but where we can reasonably assume that an underlying hierarchy exists. The CIFAR-100 dataset [9] fulfills this requirement. It is an object classification dataset composed of natural images. Because there are only 100 classes, each can be mapped to a specific synset in the WordNet hierarchy without relying on potentially faulty automation. Our mapping between CIFAR-100 classes and WordNet synsets is detailed in the supplementary material.

The target hierarchy is composed in three steps. First, the synsets mapped from all CIFAR-100 classes make up the foundation. Then, parents of the synsets are added in a recursive fashion. With the nodes of the graph complete, directed edges are determined using the WordNet hyponymy relation.

Mappings are not always obvious or unique. For instance, the CIFAR dataset contains `aquarium_fish` – a concept not captured by any WordNet synset. Intuitively, one would expect all CIFAR classes to map to leaves in the resulting hierarchy. This is true, with one exception. `dolphin` and `whale` have a direct hyperonymy relationship in WordNet because technically, dolphins are whales. As a consequence, classifiers that mandate prediction of only leaf nodes cannot be used in this context. The labels need to be encoded in a manner that allows a sample to be a `whale`, but not a `dolphin`.

Mapping all classes to the WordNet synsets results in 99 classes being mapped to leaf nodes and one class to an inner node (`whale`). In total, there are 267 nodes as a result of the recursive adding of hyperonyms.

ImageNet The aforementioned ambiguity in the mapping from dataset labels to WordNet synsets can only be reduced to a certain degree. A complete solution would require a dataset using WordNet as its label space. Because of WordNet's popularity, such datasets exist, *e.g.* ImageNet [1] and 80 Million Tiny Images [21]. We use ImageNet, specifically the dataset of the 2012 ImageNet Large Scale Visual Recognition Challenge.

It contains around 1000 training images each for 1000 synsets. The categorization into synsets eliminates mapping from classes as a potential error source. All 1000 synsets are leaf nodes in the resulting hierarchy with a total of 1860 nodes.

NABirds Quantifying performance on object recognition datasets such as CIFAR and ImageNet is important to prove the general usefulness of a method. However, more niche applications such as fine-grained recognition stand to benefit more from improvements because the availability of labeled data is much more limited. We use the NABirds dataset [22] to verify our method in a fine-grained recognition setting. NABirds is a challenge where 555 categories of North American birds have to be differentiated. These categories are comprised of 404 species as well as several variants of sex, age and plumage. It contains 48,562 images split evenly into training and validation sets.

Offered annotations include not only image labels, but also bounding boxes and parts. Additionally, a class hierarchy based on taxonomy is supplied. It contains 1010 nodes, where all of the 555 visual categories are leaf nodes.

3.2. Experimental Setup

Convolutional Neural Networks For our experiments on the CIFAR-100 dataset, we use a ResNet-32 [4] in the configuration originally designed for CIFAR. The network is initialized randomly as specified in [4].

We use a minibatch size of 128 and the adaptive stochastic optimizer Adam [7] with a constant learning rate of 0.001 as recommended by the authors. Although SGD can lead to better performance of the final models, its learning rate is more dependent on the range of the loss function. We choose an adaptive optimizer to minimize the influence of different ranges of loss values.

In our NABirds and ImageNet experiments, we use a ResNet-50 [4, 5] originally designed for ImageNet classification because of the larger image size and overall scale of the dataset. The minibatch size is reduced to 64 and training is extended to $120{,}000$ steps for NABirds and $234{,}375$ steps for ImageNet. We crop all images using the given bounding box annotations and resize them to 224×224 pixels.

To improve generalization, all settings use random shifts of up to 4 pixels for CIFAR-100 and up to 32 pixels for NABirds and ImageNet as well as random horizontal flips during training. All images are normalized per channel to zero mean and standard deviation one, using parameters estimated over the training data. Code will be published along with the paper. We choose our ResNet-50 and ResNet-32 baselines to be able to judge effects across datasets, which would not be possible when selecting a state-of-the-art method for each dataset individually. Furthermore, the moderately sized architecture enables faster training and therefore more experimental runs compared to a high performing design such as PNASNet [11].

Evaluation We report the overall accuracy, not normalized w.r.t class instance counts, averaged over different random initializations of the classifier. Each experiment consists of six random initializations per method for the CIFAR-100 dataset and three for the larger-scale NABirds and ImageNet datasets. We choose to compare the methods using a measure that does not take hierarchy into account to gauge the effects of adding hierarchical data to a task that is not normally evaluated with a specific hierarchy in mind. Using a hierarchical measure would achieve the opposite: we would measure the loss sustained by omitting hierarchical data.

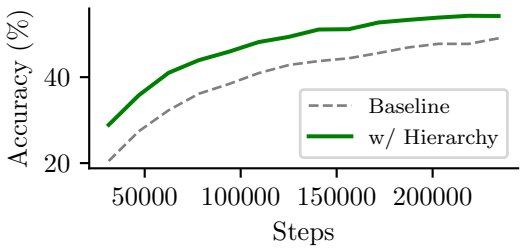


Figure 3. Accuracy on the ImageNet validation set over time. Our hierarchical training method gains accuracy faster than the flat classifier baseline. We report overall classification accuracy in percent.

Table 1. Results on the ImageNet validation set. We report classification accuracy in percent as well as the number of optimization steps needed by the baseline method to reach comparable accuracy.

	Accuracy (%)		vs. Baseline	
Steps	Baseline	w/Hierarchy	Steps	Speedup
31250	20.5 ± 0.32	28.9 ± 0.06	62500	2.00
46875	27.4 ± 0.20	35.8 ± 0.18	78125	1.67
62500	32.3 ± 0.09	41.0 ± 0.33	125000	2.00
93750	38.4 ± 0.19	45.9 ± 0.20	187500	2.00
187500	46.9 ± 0.21	53.3 ± 0.42	—	—
234375	49.0 ± 0.33	54.2 ± 0.04	—	—

3.3. Overall Improvement — ImageNet

In this experiment, we quantify the effects of using our hierarchical classification method to replace the common combination of one-hot encoding and mean squared error loss function. We use ImageNet, specifically the ILSVRC2012 dataset. This is a classification challenge with 1000 classes whose labels are taken directly from the WordNet hierarchy of nouns.

Figure 3 shows the evolution over time of accuracy on the validation set. After around 240,000 gradient steps, training converges. The one-hot baseline reaches a final accuracy of 49.1%, while our method achieves 54.2% with no changes to training except for our loss function and hierarchical encoding. This is a relative improvement of 10.4%. There is no discernible difference in computation times between both methods.

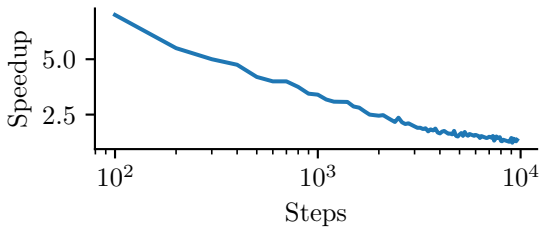


Figure 4. Initial Training speedup analysis comparing our hierarchical method to the one-hot baseline on CIFAR-100. For each training step of our method, the speedup indicates how much longer the baseline needs to train to match our performance.

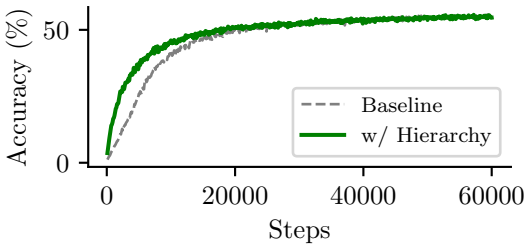


Figure 5. Results on the CIFAR-100 validation set. Our hierarchical training method gains accuracy faster than the flat classifier baseline. We report overall classification accuracy in percent.

While an improvement of accuracy at the end of training is always welcome, the effects of hierarchical classification more drastically show in the change in accuracy over time. The strongest improvement is observed during the first training steps. After training for 31250 steps using our method, the network already performs with 28.9% accuracy. The one-hot baseline matches this performance after 62500 gradient steps, taking twice as long. The baseline's final accuracy of 49.1% is matched by our method after only 125,000 training steps, resulting in an overall training speedup of 1.88x. We provide a more detailed picture of the initial training in Table 1. Speedup indicates how much longer the baseline needs to match our hierarchical method's performance.

Overall, our method performs better and learns more quickly at the same time with no significant increase in computational effort.

3.4. Speedup — CIFAR-100

We report the accuracies on the validation set as they develop during training in Figure 5. As training converges, we observe almost no difference between both methods, with our hierarchical method reaching 54.6% and the one-hot encoding baseline at 55.4%. However, the methods differ strongly in the way that accuracy is achieved. After the first

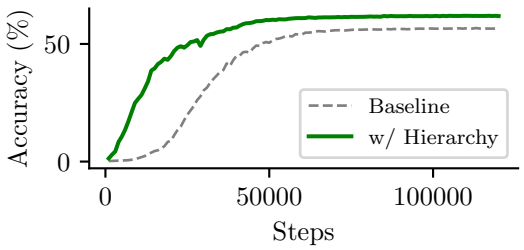


Figure 6. Accuracy on the NABirds validation set over time.

500 steps, our hierarchical classifier already predicts 10.7% of the validation set correctly, compared to the baseline's 2.8%. It takes the baseline another 1600 steps to match 10.7%, or 4.2 times as many steps.

This advantage in training speed is very strong during initial training, but becomes smaller over time. Figure 4 details the observed speedup during initial training. After the first half of training, the difference between both methods vanishes almost completely.

3.5. Fine-Grained Recognition — NABirds

To evaluate the performance of our hierarchical method in a more specific setting, we use the NABirds dataset [22], a fine-grained recognition challenge where the task is to classify 555 visual categories of birds. The dataset supplies a hierarchy which we use in this experiment.

We observe results similar to the ImageNet dataset (see Section 3.3), where our method leads to an improvement in both training speed and overall accuracy. Figure 6 shows the development of validation accuracy over time. The one-hot baseline converges to an accuracy of 56.5%. Our hierarchical classifier reaches 61.9% after the full 120,000 steps of training. It already matches the baseline's final accuracy at 39,000 iterations, reducing training time to less than a third. The relative improvement when applying the full training time is 9.6%.

Overall, our method leads to a faster and better solution of the fine-grained recognition task, while keeping the same classifier and requiring no discernible amount of addition computation time.

3.6. Mandatory Labeled Node Prediction

To assess the effects of restricting the class set as specified in Section 2.3, we compare two variants of our method by observing the validation accuracy on ImageNet over time. The first, mandatory labeled node prediction (MLNP), restricts the possible predictions of the classifier to the 1000 classes available in the original dataset. The test set does not contain samples of any other class. The second variant, arbitrary node prediction (ANP), removes this restriction and

Table 2. Results Overview.

Dataset	# of classes	Accuracy (%)		Speedup w/Hierarchy	
		Baseline	w/Hierarchy	Overall	Initial
CIFAR-100	100	**55.4** $\pm$ 0.84	54.6 $\pm$ 1.03	—	7.00
NABirds	555	56.5 $\pm$ 0.49	**61.9** $\pm$ 0.27	3.08	10.00
ImageNet (ILSVRC2012)	1000	49.1 $\pm$ 0.33	**54.2** $\pm$ 0.04	1.88	—

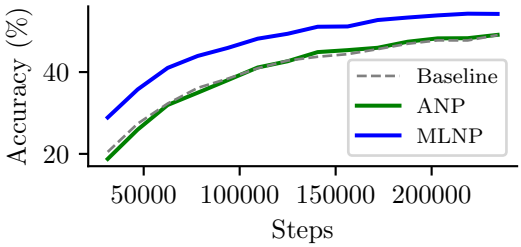


Figure 7. Effects of restricting the predicted classes. Graph shows validation performance on the ImageNet dataset. Mandatory labeled node prediction (MLNP) offers better performance than arbitrary node prediction (ANP) throughout training.

can predict any of the 1860 classes present in the complete hierarchy. This includes inner nodes as well as the root.

This distinction does not affect training, as both variants use the same loss function given in Section 2.4. The only difference is the set of classes over which the maximum is calculated (cf. Section 2.3). during inference. As a consequence, the accuracy of ANP cannot be better than MLNP, since any prediction of a non-dataset class is an error.

Figure 7 shows the results of this experiment, including the one-hot baseline for comparison. After training is complete, MLNP offers a validation accuracy of 54.2%. ANP finishes training with 49.1% validation accuracy, performing comparably to the one-hot baseline at 49.1%. The improvement in performance from adding the MLNP restriction over ANP is 5.1 percent points, leading to a relative increase in accuracy of around 10.4%.

This result illustrates the benefit of knowing that the test data is restricted to a specific set of classes, as is the case when our method is employed to augment a classifier for an existing problem with a predefined label space. It might still be interesting to observe the unrestricted output and interpret its hierarchical depth as an indicator of confidence.

3.7. Overview

This section provides an overview, combining the results over the different datasets. Table 2 provides the most important facts for each dataset. We report the accuracy at the end of training for the one-hot encoding baseline as well as our method. Speedup is quantified in two ways. Overall speedup indicates how much faster our hierarchical method achieves the end-of-training accuracy of the baseline, measured in number of training steps. Initial speedup looks at the accuracy delivered by our method after the first validation interval. We then measure how much longer the baseline needs to match that number.

On all 3 datasets, the initial training is faster using our method. However, we only observe an improvement in classification accuracy on ImageNet and NABirds. With CIFAR-100, the benefits of adding the hierarchical information to the training are limited to the speed of learning. There are a few possible explanations for this observation:

The CIFAR-100 dataset is the only dataset that requires a manual mapping to an external hierarchy, whereas the other datasets either supply one or have labels directly derived from one. The manual mapping is a possible error source and as such, could explain the observation.

The second possible reason lies in the difference between semantic similarity and visual similarity [2]. Semantic similarity relates two classes using their meaning. It can be extracted from hierarchies such as WordNet [3], for example by looking at distances in the graph. Other relationships can also be used to establish semantic distances, *e.g.* synonymy, troponymy or meronymy. Visual similarity on the other hand relates images that look alike, regardless of the meaning behind them. When classifying, we group images by semantic similarity because they contain objects of the same class, even if they share no visual characteristics, possibly making the classification problem more complex. This may be exacerbated when adding more information based on semantics and not properties of the images themselves.

4. Conclusion

We present a method to modify existing deep classifiers such that knowledge about relationships between classes can be integrated. The method is derived from a probabilistic model that is itself based on our understanding of the meaning of hierarchies. Overall, it is just one example of the integration of domain knowledge in an otherwise general method. One could also consider our method a special case of learning using privileged information [24].

Our method can improve classifiers by utilizing information that is freely available in many cases such as WordNet [3] or WikiSpecies. There are also datasets which include a hierarchy that is ready to use with very little effort [1, 22]. The additional computational expense of using our method in the context of deep learning is negligible.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 2, 3, 5, 8

[2] T. Deselaers and V. Ferrari. Visual and semantic similarity in imagenet. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1777–1784, 2011. 8

[3] C. Fellbaum. *WordNet*. Wiley Online Library, 1998. 1, 2, 3, 8

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 6

[5] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645, 2016. 6

[6] J. Hoffman, S. Guadarrama, E. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko. Lsda: Large scale detection through adaptation. In *arXiv preprint arXiv:1407.5035*, 2014. 1

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations (ICLR)*, 2014. 6

[8] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)*, 123(1):32–73, 2017. 1, 2

[9] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 1, 5

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. 2012. 1

[11] C. Liu, B. Zoph, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *arXiv preprint arXiv:1712.00559*, 2017. 6

[12] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[13] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1410–1418. 2009. 2

[14] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-R. Amini, and P. Galinari. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, 2015. 2

[15] E. Rodner and J. Denzler. One-shot learning of object categories using dependent gaussian processes. In *Joint Pattern Recognition Symposium*, pages 232–241, 2010. 3

[16] M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1641–1648, 2011. 2

[17] B. Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison, 2009. 1

[18] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPR-WS)*, 2014. 1

[19] C. N. Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1):31–72, 2011. 2

[20] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision (ICCV)*, pages 843–852, 2017. 1

[21] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(11):1958–1970, 2008. 2, 3, 5

[22] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 595–604, 2015. 1, 6, 7, 8

[23] G. Van Horn, O. Mac Aodha, Y. Song, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist challenge 2017 dataset. In *arXiv preprint arXiv:1707.06642*, 2017. 1, 2

[24] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009. 1, 8

[25] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. 1