On Expert-Defined Versus Learned Hierarchies for Image Classification

Ivan Panchenko(✉) and Adil Khan

Institute of Artificial Intelligence and Data Science, Innopolis University, 1, Universitetskaya str., Innopolis, Russia
{i.panchenko,a.khan}@innopolis.com

Abstract. For classification task involving a large number of classes, a decrease in recognition accuracy is observed for visually similar classes. We believe that forcing the model to learn appropriate features separately for each set of similar classes could improve classification performance. To justify our idea, we tried to improve classification performance by employing class hierarchy, which reflects visual similarities in data. More specifically, we used and compared two kinds of hierarchies to enhance classification performance of the model: (i) a hierarchy defined by experts (H-E), and (ii) a hierarchy created from performance results of a flat classifier and using DBScan clustering method (H-C). Moreover, we created a classification model that efficiently utilizes these hierarchies to learn appropriate features at different levels of the hierarchy. We evaluated the performance of the model on CIFAR-100 benchmark. Our results demonstrate that the hierarchical classification under H-C outperforms both H-E and the flat classifier.

Keywords: Hierarchical classification · Image classification · Confusion matrix · Class similarity

1 Introduction

In classification tasks, a trained model is used to predict the class of given data points, where classes are sometimes referred to as targets, labels, or categories. Machine learning approaches to solving a classification problem can be split into two categories. The first one is the classical machine learning approach, which uses algorithms like Decision Trees, Naive Bayes Classifiers and SVMs [9]. The second approach incorporates Deep Learning (DL) [13]. One of the main advantages of the DL models is that they require little or no manual feature engineering, meaning that the DL model is able to learn features itself. This property has given rise to machine learning models, able to solve tasks involving unstructured data, like videos, images and voice, which outperform approaches where features were created manually [12].

Functioning of a DL model can be split into three steps. The first step is features learning, which is performed by the first layers of the neural network.

These layers can be fully connected layers for simple networks [15] as well as more complex convolutional blocks [12] for image-recognition tasks and other options for different DL problems. The second step is responsible for features combination. This step is usually implemented as several fully connected layers which combine all available features and pass them into the third step. The third step involves prediction, consisting of a layer with activations having the same number of neurons as the number of classes. The output of the model is a probability distribution over all classes.

Despite their high-performance, even DL models tend to underperform in some cases. Misclassifications may occur when samples from different classes are very similar. For example, in Kinetics dataset [8] for action recognition, the top 3 misclassified class-pairs are: *riding a mule* and *riding or walking with horse*, *hockey stop* and *ice skating*, *swing dancing* and *salsa dancing*. Obviously, each of these class pairs shares many visual features.

These features allow a model to recognize mentioned classes among all other classes in a dataset. However, to distinguish among similar classes, specific features, which exists only in classes within a pair, are required. For example, features that can distinguish the kind of the animal within the first pair of actions, or the specific hand movements within the third one. Such examples lead to a statement that features that are useful for discrimination of similar samples among all categories are not necessarily discriminative for samples that fall in one compound category, and vice-versa.

To address this problem, we employ an artificial hierarchy in data in a way where classes that are hard to distinguish are united in one category. Besides, we propose a model that utilizes the learned hierarchy and performs classification in two steps. During the first step, a compound category is created. During the second step, the division of classes into categories is performed by an additional classifier. The proposed type of architecture, which exploits the similarity relationship between classes is called Hierarchical Classification (HC).

A number of works successfully utilizing HC were published in the last decade, e.g. [1,2,10,14]. This research effort contributes to the domain by addressing the question of creating an artificial hierarchy in data. More specifically, in this work we compare two hierarchies, obtained by different means on the same dataset. One hierarchy is a predefined one, or a human-labelled hierarchy. The second one is obtained by defining class separability metrics and utilizing them. We also introduce and compare a number of model architectures in order to identify the one that produces the highest accuracy on a selected dataset.

To conclude, this work aims to investigate three problems. The first problem is that of identification or creation a hierarchy in data. The second addresses building a hierarchical architecture that efficiently utilizes artificial hierarchy. The third problem involves the comparison of an expert defined hierarchy (H-E) with an artificially created hierarchy (H-C) in terms of model performance.

2 Related Works

The first question to attend to is how to create a hierarchy from the data. The simplest decision is to label classes manually or use predefined hierarchical structure [7,17,21], but we state that a hierarchy obtained in this way does not reflect separation between classes.

Another option to get a hierarchy is to create it. The process of creating an artificial hierarchy can be divided into three steps: defining a space for measurement of the distance between classes, building a hierarchical dependency tree of classes based on the defined distance and constructing a hierarchy from the dependency tree.

To define a space, where the distance between classes could be measured, two approaches are known in the literature. The first one is premature class separation, based on existing data. An example is clustering by labels' semantic [4]. The second approach is to create a hierarchy on the predictions of a trained model. This approach utilizes confusion matrix (CM) of a pretrained model as a distance matrix [6,22]. While the first approach requires additional information and has limited applicability, the second one is easy to implement and it reflects similarities of classes.

There are several widely used methods, as described in literature, to build a hierarchical dependency tree from CM. For example, Yan et al. [22] used spectral clustering method, while Silva-Palacios et al. [20] utilized an agglomerative clustering to create a dendrogram and a hierarchy from it. Both works treated CM as an inter-class distance measurement. Another option to perform clustering is DBScan algorithm [5]. An advantage of this method is its ability to detect outliers and perform grouping, based on class density.

CM values cannot be treated as distances between classes directly. There are transformations to CM that should be applied beforehand. An essential property that CM should hold is symmetry; otherwise, the distance from class A to class B is not guaranteed to be equal to the distance from class B to class A. An application of symmetric matrix is shown by Yan et al. [22]. The authors used a distance matrix, which is a transformed CM, with a symmetry property. Another valid transformation is called a similarity matrix, described in [20].

The next step after constructing a hierarchy is building a model that efficiently utilizes it. The task can be approached by: disregarding the hierarchy; connecting separated models, where each model corresponds to one node in a hierarchy; and building a single model, that has an internal hierarchical structure.

The first, and the simplest, approach ignores hierarchy and builds a flat classifier. This approach is shown to be less effective than a hierarchical classifier [22]. There are several possibilities to arrange models in the second approach. The literature distinguishes three of them: Local Classifier per Node (LCN), Local Classifier per Parent Node (LCPN) and Local Classifier per Level (LCL) [18]. A major drawback of this family of approaches is that they require separate models. To understand a problem with separate models, consider a Deep Neural Network (DNN). First layers of the DNN are responsible for learning small, low-level features like lines and edges (if we deal with pictures) [16]. These features are domain-specific, not class-specific. Therefore, learning low-level features for all classifiers is redundant. Moreover, local classifiers (every classifier that is not the root one) receives only part of training samples, which makes the model harder to train.

The third approach embeds a hierarchy directly inside the model. This idea aims to improve the shortcomings of other approaches. It was utilized by Yan et al. [22]. The authors build an architecture, called HD-CNN. HD-CNN has an intrinsic model that predicts coarse labels and a model for each set of fine labels. Coarse label prediction is responsible for choosing the most relevant subclass models and perform weighting on their output. Overall, the architecture is similar to an ensemble of models, but low-level features are shared.

Later work by Bilal et al. [3] utilizes hierarchy internally. The authors went further with the process of training a hierarchical model. They made a preliminary analysis of mistakes made by the model in hidden layers during training. Based on gained insights, the authors added intermediate outputs in the model architecture to enforce hierarchical structure. The disadvantage of this work is that building such hierarchy-aware model requires complicated preliminary analysis.

3 Methodology

3.1 Creation of a Hierarchy

We took a dataset called CIFAR-100 [11]. It consists of tiny 32×32 color images. The dataset has 100 classes with 500 train and 100 test images in each. Each class is also assigned to a superclass. A superclass is a disjoint group of classes. For CIFAR-100, superclasses were split by a human expert, relying on common knowledge. A label that identifies an image among 100 classes is called a fine label, while a label that maps an image to a superclass is called a coarse label. The list of classes and superclasses is presented in Table 1. Predefined mapping into fine and coarse categories implies a hierarchical structure of the data. This hierarchy is used as a baseline.

A better approach is to construct a hierarchy that reflects class similarity from the viewpoint of a classification model. The first step in the creation of such a hierarchy is getting a measurement of inter-class distance. The distance is obtained from CM, which in turn is obtained from the classification model. A well-known architecture for image classification is AlexNet [12]. Its architecture is shown in Fig. 1. A trained model with this architecture is used to create a CM.

Bare CM cannot be used to perform clustering on classes in order to build a hierarchy. A suitable matrix that can be obtained through transformation is symmetrical CM: F. To get an F, the CM (represented as C) is summed up with its transpose to make matrix symmetric and multiplied by a factor of 0.5 to normalize coefficients (Eq. 1a). Another option is distance matrix D which is

Table 1. Given hierarchy of classes

Coarse class	Fine classes
Aquatic mammals	Beaver, dolphin, otter, seal, whale
Fish	Aquarium fish, flatfish, ray, shark, trout
Flowers	Orchids, poppies, roses, sunflowers, tulips
Food containers	Bottles, bowls, cans, cups, plates
Fruit and vegetables	Apples, mushrooms, oranges, pears, sweet peppers
Household electrical devices	Clock, computer keyboard, lamp, telephone, television
Household furniture	Bed, chair, couch, table, wardrobe
Insects	Bee, beetle, butterfly, caterpillar, cockroach
Large carnivores	Bear, leopard, lion, tiger, wolf
Large man-made outdoor things	Bridge, castle, house, road, skyscraper
Large natural outdoor scenes	Cloud, forest, mountain, plain, sea
Large omnivores and herbivores	Camel, cattle, chimpanzee, elephant, kangaroo
Medium-sized mammals	Fox, porcupine, possum, raccoon, skunk
Non-insect invertebrates	Crab, lobster, snail, spider, worm
People	Baby, boy, girl, man, woman
Reptiles	Crocodile, dinosaur, lizard, snake, turtle
Small mammals	Hamster, mouse, rabbit, shrew, squirrel
Trees	Maple, oak, palm, pine, willow
Vehicles 1	Bicycle, bus, motorcycle, pickup truck, train
Vehicles 2	Lawn-mower, rocket, streetcar, tank, tractor

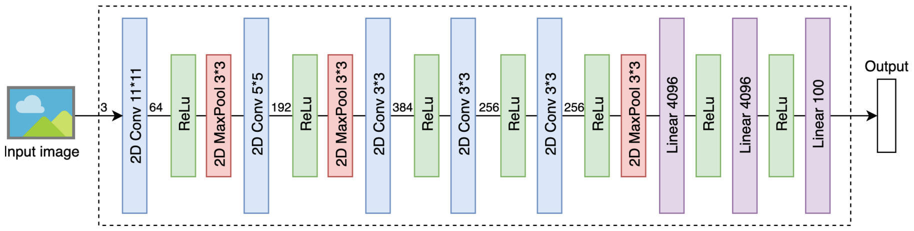


Fig. 1. Simplified AlexNet architecture

a modification of F (Eq. 1b).

$$F = 0.5 * (C + C^T) \tag{1a}$$

$$D = 0.5 * (D_{tmp} + D_{tmp}^T), \text{ where } D_{tmp} = 1 - C \tag{1b}$$

The last approach is called a similarity matrix, which is described by Silva et al. [20]. Similarity matrix S is calculated as in Eq. 2.

$$\begin{gathered}\text{Normalization} : C_{ij} = \frac{C_{ij}}{\sum_{j=1}^{n} C_{ij}} \\ \text{Overlap} : O_{ij} = \begin{cases} \frac{C_{ij}+C_{ij}}{2} & \text{if} i \neq j \\ 1 & \text{if} i = j \end{cases} \\ \text{Similaritymatrix} : S_{ij} = 1 - O_{ij}\end{gathered} \tag{2}$$

The next step is building a hierarchy, based on inter-class distances. To produce a hierarchy, we performed clustering on classes, using transformed CM as a measure. For the classes that occur in the same cluster, it is true that they are often confused with each other or they are visually similar. Classes in one cluster are said to belong to the same superclass. Moreover, for each outlier class, produced by the clustering method, its own superclass is created. Outliers are treated this way because not belonging to any cluster means the class is not often confused and is simple to predict.

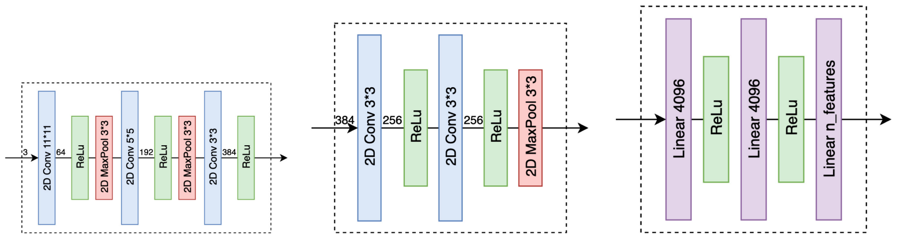
3.2 Building a Model

We built two models to investigate and compare performance on classification task involving class hierarchy. The first model is unaware of hierarchy and is referred to as the flat model. Architecture is the same as the one in Fig. 1.

The second model that we created is a modification of HD-CNN architecture [22]. The structure of its architecture is built upon basic blocks, depicted in a Fig. 2, whereas the complete model architecture is shown in Fig. 3.

There are three out of five possible types of blocks in the Fig. 2. Block Fig. 2a is responsible for learning low-level features. Block Fig. 2c is responsible for internal feature combination, which we assume is superclass specific. Finally, block Fig. 2c outputs probability vector for the sample over target distribution. Moreover, there are two kinds of blocks that are not in the picture. The first one is a fine category or coarse category prediction (shown in red in Fig. 3). It is just a convenient depiction of the output vector. The second non-drawn block also referred to as “Ones vector” block (shown in purple in Fig. 3) just returns a vector of ones, with the size equal to the input vector.

The architecture in Fig. 3, consists of three main parts: coarse category prediction, fine categories prediction, and weighting block. Convolutional layers are shared between the coarse category model and fine category models. Shared features block was used to address the problem of retraining the same low-level features. Coarse category classifier produces a probability distribution over coarse classes for an input sample. Fine category classifier outputs a distribution over fine classes in one superclass if a superclass consists of more than one class. If a superclass consists of one class, then fine classifier outputs vector with ones. Finally, inside a weighting block, each vector from fine category classifier is multiplied by the corresponding probability of a coarse category, obtained from coarse category classifier.



(a) Shared features block (b) Compound features block (c) Classification block

Fig. 2. Architecture building blocks: Numbers before and after 2D Convolutions are the number of input and output channels correspondingly. If no number specified, then the number of channels is propagated from previous output.

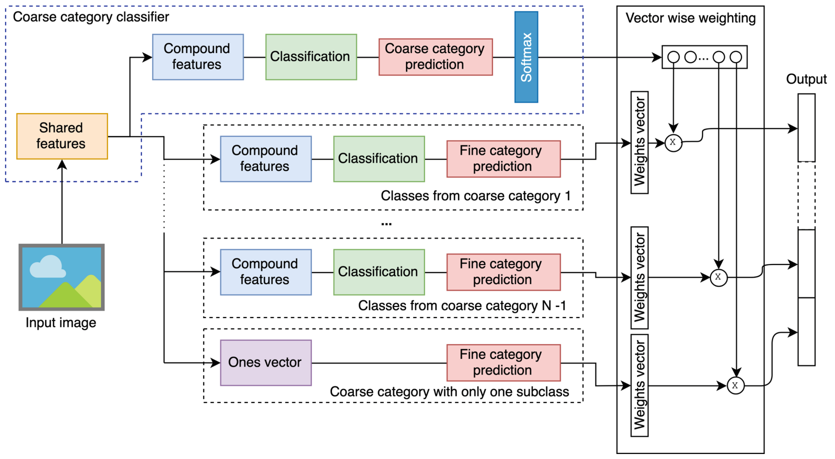


Fig. 3. Hierarchical model architecture (Color figure online)

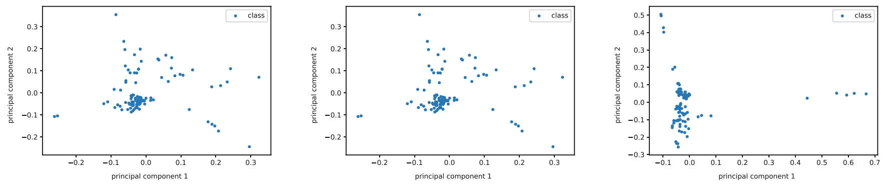
There are differences between the proposed model and the existing work. Firstly, we didn't select fine classifiers after a prediction of the coarse category; instead, we passed an input sample to all available fine category classifiers. Secondly, we allowed the hierarchy to contain categories with one class, which leads to the presence of fine category classifiers with only one element and constant output.

4 Implementation

4.1 Hierarchy Creation

To create a hierarchy from data, first, we trained AlexNet (Fig. 1) using implementation from PyTorch [19] framework. This model also acts as a flat classifier.

The model achieved 61.04% accuracy on the test set. For this and further models training results are shown in Table 3.



(a) Symmetric confusion matrix (b) Distances matrix (c) Similarity matrix

Fig. 4. 2D scatter plots after PCA decomposition of the three matrices.

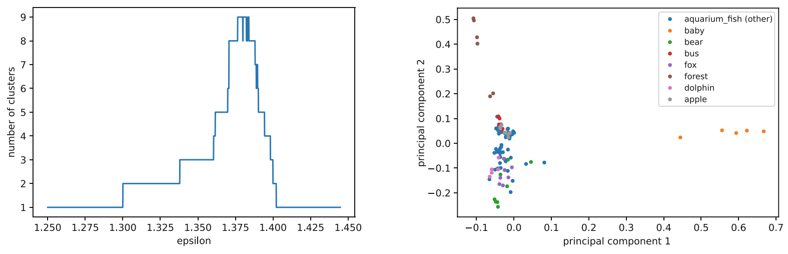
Next, the data from the training set were passed through the model again to obtain a CM of the classes. We applied three different methods to transform the matrix into distances spaces. Then PCA decomposition was performed on all transformed matrices: symmetric CM F (Fig. (4a), distances matrix D (Fig. 4b) and similarity matrix S (Fig. 4c). While decomposed F is similar to decomposed D and both of plots are diffused, similarity matrix has perceptible clusters. Therefore, similarity matrix is chosen for further processing.

The next step after choosing distances space is clustering. To perform clustering DBScan algorithm was utilized. We search for optimal epsilon value and a minimum number of neighbors for DBScan by finding the combination that returns a minimum number of clusters, while not allowing very small and very big ones. Epsilon value is a radius of a neighborhood around a point. A minimum number of neighbors is a restriction of a size for a valid cluster. So, the search space was bounded by minimum and maximum distances between points and a minimum number of neighbors. Distribution of the number of clusters over epsilon with a fixed number of neighbors is shown in Fig. 5a. The highest number of clusters is 9, according to the plot, but this marginal value is not stable, therefore, we decided to use 8 clusters. An epsilon, in this case, is equal to 1.374 and distance metric is Euclidean distance. A value for the minimum number of neighbors is 5.

The resulting clusters are plotted in Fig. 5b. The title of a cluster is the name of one of the classes, that belongs to it. Moreover, one cluster is marked as "other". These are outliers produced by DBScan. In our setup, it means that classes that occurred in this category are easily distinguishable from each other. A hierarchy that clustering produces is shown in Table 2.

4.2 Models Training and Testing

We trained two hierarchical models. The first model uses experts hierarchy (H-E), and the second one uses the created hierarchy (H-C). The training process



(a) Clusters over epsilon (b) DBScan results

Fig. 5. The distribution of number of clusters over epsilon obtained using DBScan, and the resulting clusters.

of each hierarchical model was split into three steps: training a coarse classifier, training full classifier, and fine-tuning fine-label models.

The first step is training a coarse classifier. Coarse category classifier part of the model is isolated to perform training. Labels for training are coarse labels from the hierarchy. Therefore, the number of classes for H-C is equal to the number of superclasses added to the number of outliers, or $56 + 7 = 63$ and 20 for H-E. H-E achieved 76.1%, while H-C achieved 75.79% accuracy. The second step is model training on fine classification. Shared features block is frozen to prevent low-level features from training.

Investigation of trained H-C model showed that it performs better in grouping classes into supergroups, compared to the flat model. Considering samples that belong to groups in an artificial hierarchy, 965 samples are confused with a sample from its group, when passed through the flat model. In the hierarchical model, there are 1380 confused samples. While a total number of samples, belonging to groups, is 4400.

Another insight is a prediction of non-grouped classes in the hierarchical model. Samples that don't belong to any group have the same labels for both coarse and fine classification. But the result of fine classification is weighted and affected by weighting layer. Still, the difference in the number of correctly classified samples equals 153, which is 0.015 from the original dataset size. It means that overall accuracy does not suffer from weighting layer.

Therefore, the next step in training models is improving performance of each group classifier. For this purpose, shared features are directly passed to a group classifier and backpropagation starts from the output of the group classifier. Fine tuning for the model improved the performance of both models. H-E scored 59.72% and H-C scored 63.67% accuracy.

We assume that the reason why H-C performed better than H-E is differences in the hierarchy. Expert defined hierarchy is built upon common knowledge of grouping. Created hierarchy is based on a model perception of the classes, which implies classes within a group share more class-specific features. Therefore, fine labels classifiers are able to concentrate on distinguishing features, thus improving overall accuracy.

Table 2. Created hierarchy of classes

Coarse class	Fine classes
Outliers	Aquarium_fish, bed, bee, beetle, bicycle, bottle, bowl, bridge, butterfly, camel, can, castle, caterpillar, cattle, chair, clock, cloud, cockroach, couch, crab, crocodile, cup, dinosaur, flatfish, hamster, house, keyboard, lamp, lawn_mower, leopard, lizard, lobster, motorcycle, mountain, mouse, mushroom, plain, plate, possum, raccoon, road, rocket, sea, skunk, skyscraper, snail, snake, spider, sunflower, table, tank, telephone, television, turtle, wardrobe, worm
0	Baby, boy, girl, man, woman
1	Bear, beaver, chimpanzee, elephant, otter, porcupine, seal, shrew
2	Bus, pickup_truck, streetcar, tractor, train
3	Fox, kangaroo, lion, rabbit, squirrel, tiger, wolf
4	Forest, maple_tree, oak_tree, palm_tree, pine_tree, willow_tree
5	Dolphin, ray, shark, trout, whale
6	Apple, orange, orchid, pear, poppy, rose, sweet_pepper, tulip

Table 3. Results

Model	Validation accuracy for coarse classes	Validation accuracy for fine classes
Flat model	–	0.6259
H-E	0.76101	0.5972
H-C	0.75792	0.6367

5 Conclusion

In this work we performed an experiment, proving that forcing a model to learn appropriate features separately for each set of similar classes could improve classification performance. There are several works in the field, that utilize the same idea [3,20,22]. In comparison with other works, we applied a novel approach to building a hierarchy using DBScan clustering and allowing the hierarchy to have single-element groups. Moreover, we changed an existing model architecture (HD-CNN) to simplify internal fine classifier selection and proposed an approach for fine-tuning hierarchical models.

The experiment results demonstrate that hierarchical classification with created hierarchy outperforms hierarchical classification with expert-created hierarchy, as well as flat classification. It means that introducing an artificial hierarchy into a classification task could improve the overall accuracy, regardless of the presence of a given hierarchy.

References

1. Babbar, R., Partalas, I., Gaussier, E., Amini, M.R.: On flat versus hierarchical classification in large-scale taxonomies. In: Advances in Neural Information Processing Systems, pp. 1824–1832 (2013)
2. Bennett, P.N., Nguyen, N.: Refined experts: improving classification in large taxonomies. In: Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 11–18. ACM (2009)
3. Bilal, A., Jourabloo, A., Ye, M., Liu, X., Ren, L.: Do convolutional neural networks learn class hierarchy? IEEE Trans. Vis. Comput. Graph. **24**(1), 152–162 (2018)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. KDD **96**, 226–231 (1996)
6. Griffin, G., Perona, P.: Learning and using taxonomies for fast visual categorization. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)
7. Jia, Y., Abbott, J.T., Austerweil, J.L., Griffiths, T., Darrell, T.: Visual concept learning: combining machine vision and Bayesian generalization on concept hierarchies. In: Advances in Neural Information Processing Systems, pp. 1842–1850 (2013)
8. Kay, W., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017)
9. Kotsiantis, S.B., Zaharakis, I., Pintelas, P.: Supervised machine learning: a review of classification techniques. Emerg. Artif. Intell. Appl. Comput. Eng. **160**, 3–24 (2007)
10. Kowsari, K., Brown, D.E., Heidarysafa, M., Meimandi, K.J., Gerber, M.S., Barnes, L.E.: HDLTex: hierarchical deep learning for text classification. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 364–371. IEEE (2017)
11. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, Citeseer (2009)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
13. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
14. Li, T., Zhu, S., Ogihara, M.: Hierarchical document classification using automatically generated hierarchy. J. Intell. Inf. Syst. **29**(2), 211–230 (2007)
15. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
16. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3074–3082 (2015)

17. Marszalek, M., Schmid, C.: Semantic hierarchies for visual object recognition. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–7. IEEE (2007)
18. Naik, A., Rangwala, H.: Large Scale Hierarchical Classification: State of the Art. SCS. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01620-3
19. Paszke, A., et al.: Automatic differentiation in PYTorch (2017)
20. Silva-Palacios, D., Ferri, C., Ramírez-Quintana, M.J.: Improving performance of multiclass classification by inducing class hierarchies. Procedia Comput. Sci. **108**, 1692–1701 (2017)
21. Verma, N., Mahajan, D., Sellamanickam, S., Nair, V.: Learning hierarchical similarity metrics. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2280–2287. IEEE (2012)
22. Yan, Z., et al.: HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2740–2748 (2015)