Pattern Recognition 147 (2024) 110135

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

H-CapsNet: A capsule network for hierarchical image classification

Khondaker Tasrif Noor [a,*], Antonio Robles-Kelly [a,b]

[a] School of Information Technology, Deakin University, Waurn Ponds, VIC, 3216, Australia
[b] Defence Science and Technology Group, Edinburgh, SA, 5111, Australia

ARTICLE INFO

Keywords:
Capsule networks
Hierarchical image classification
Convolutional neural networks
Deep learning

ABSTRACT

In this paper, we present H-CapsNet, a capsule network for hierarchical image classification. Our network makes use of the natural capacity of CapsNets (capsule networks) to capture hierarchical relationships. Thus, our network is such that each multi-layer capsule network accounts for each of the class hierarchies using dedicated capsules. Further, we make use of a modified hinge loss that enforces consistency amongst the hierarchies involved. We also present a strategy to dynamically adjust the training parameters to achieve a better balance between the class hierarchies under consideration. We have performed experiments using several widely available datasets and compared them against several alternatives. In our experiments, H-CapsNet delivers a margin of improvement over competing hierarchical classification networks elsewhere in the literature.

1. Introduction

Classification is a fundamental problem in computer vision. It involves learning a function that predicts the membership of object instances to different classes. Image classification has found applications in medical imaging [1], satellite image processing [2], online marketing [3], image collection organisation and indexing [4] and tagging [5]. Note that these image classification tasks usually follow a supervised approach where all classes are treated equally, devoid of a taxonomical or hierarchical structure between them.

Viewed in this manner, traditional image classification tasks often comprise a single class prediction per image. These types of classification models treat all predicted classes equally, with no notion of hierarchy or precedence between them. Hierarchical classification, on the other hand, is a classical problem in which objects are organised into fine categories which are then grouped into coarse-level ones based upon semantic relations. This is because some classes are deemed to contain similar semantic features or traits and, therefore, can be grouped together to endow a hierarchical structure to the class set. Note that, in hierarchical image classification, the models generally employ the "coarse-to-fine" paradigm where coarse class predictions and features are used to guide those of fine classes. Further, nonetheless a dedicated classifier can be trained to predict fine classes [6], hierarchical classification is such that often the coarse level class takes precedence over the fine one. As a result, when performing hierarchical classification, an image classifier often first predicts coarse levels and then goes to predict finer ones. The advantage of this treatment is that, in hierarchical classification, the error can be confined to hierarchical levels whereby the model is expected to be well suited for the hierarchy under consideration [7].

This contrasts with CapsNets, which operate upon the principle of "routing by agreement", which allows them to model visual hierarchical relationships. Recall that capsule networks (CapsNets) [8] employ the likelihood of semantic features and the orientations of these as the instantiation parameter values. As a result, the CapsNet can learn both, the image features and their transformations allowing for a natural means of recognition by parts. Thus, here we profit from their capacity to learn relational information for hierarchical classification by using the capacity of CapsNets to model semantic relationships of image features. Furthermore, CapsNets often require less data to train since they exhibit an equivariant behaviour which helps them to learn rotational invariants, making the model robust to viewpoint changes.

In this paper, we present a CapsNet for hierarchical image classification (H-CapsNet) containing a dedicated capsule network for each hierarchical level. The dedicated capsule networks for the hierarchical levels in our H-CapsNet do share a similar capsule structure and routing mechanism as proposed in [8]. However, each dedicated capsule network has level-specific feature extractor blocks to allow level-specific capsules to learn hierarchical features. Further, in contrast with the flat classification approach in [8], the approach presented here combines the contribution of each level-specific capsule network following the "coarse-to-fine" paradigm. This also applies to the reconstruction error often employed in the training of capsule networks, such as that in [8],

* Corresponding author.
E-mail addresses: knoor@deakin.edu.au, k.noor@research.deakin.edu.au (K.T. Noor).

https://doi.org/10.1016/j.patcog.2023.110135
Received 25 September 2022; Received in revised form 30 August 2023; Accepted 17 November 2023
Available online 20 November 2023

whereby our proposed H-CapsNet employs the decoder outputs from all the hierarchical levels rather than just a single "flat" one.

Thus, the structure of the H-CapsNet presented here is motivated by the notion that hierarchical models should take advantage of the data-hierarchy, delivering predictions consistent with the applicable label tree [7]. In this manner, our network exploits the natural advantages of capsule networks while incorporating the hierarchical relations between image features. To this end, we introduce a special training strategy that encourages consistency with the class-hierarchy structure while adjusting the training parameters of the H-CapsNet model so as to better balance the contributions of each hierarchy level to the loss. As a result, our H-CapsNet model can effectively learn the class-hierarchy whereby coarse level parameters are prioritised earlier and, later on in the training process, more importance is given to finer categories based on the model performance. In our experiments, H-CapsNet yields a margin of improvements over alternatives elsewhere in the literature, converging at a faster rate while keeping consistency between class-hierarchies.

Thus, this paper is organised as follows. In the following section, we review prior work on both, hierarchical classification and capsule networks. In Section 3 we present our hierarchical capsule network, H-CapsNet. We also elaborate upon the loss function used here and the training strategy for adjusting automatically the contribution of each of the class hierarchies to the loss. In Section 4, we present results on widely available datasets and compare against alternatives. We also present an ablation study. Finally, in Section 5 we conclude upon the developments presented here.

2. Related work

As mentioned above, in this section, we review the literature on hierarchical classification and capsule networks.

2.1. Capsule networks

We commence by noting that capsule networks (CapsNet) can learn both, the image features and their transformations. This allows CapsNets to naturally identify image classes by executing recognition by parts. Recall that CapsNets use a set of neurons to obtain an "activity vector". The neurons in the CapsNet are thus grouped into capsules, whereby deeper layers can be viewed as the probability of the outputs from preceding capsules "agreeing" with one another.

Indeed, in the computer vision and machine learning communities CapsNets have attracted a lot of attention due to their viewpoint invariance capabilities. This is important since they can address the "Picasso effect" in classifiers. Furthermore, CapsNets are robust to input perturbations when compared to other CNNs of similar size [9]. As a result, several computer vision tasks have been tackled using CapsNets, including text classification [10], 3D data processing [11], target recognition [12] and image classification [13]. It is worth noting in passing that, despite the growing interest in CapsNets, to our knowledge, they have not been applied to hierarchical multi-label classification problems.

The theory of capsules was originally proposed in [14], where capsules were applied to maintain the relations of spatial features in the input data so as to learn instantiation parameters that are robust to variations in position, orientation, scale and lighting. In [8] the authors presented a dynamic routing technique for the routing algorithm. Their capsule implementation asserted that CapsNets can overcome viewpoint invariance problems and are effective for classifying highly overlapping images. Later on, in [15] Hinton et al. proposed a probabilistic routing technique based upon the EM-algorithm [16] to learn part-whole relationships. Based on the EM routing strategy in [15], Bahadori [17] proposes a spectral capsule network with improved convergence with respect to that employing the EM routing so as to compute the capsule activation and pose. Capsule networks have also been extended to architectures such as Siamese networks [18], generative adversarial networks (GANs) [19] and residual networks (ResNets) [20]. A Multi-Column Capsule Network is proposed in [21], which computes agreements between neurons in different layers using a two-phase dynamic routing protocol.

2.2. Hierarchical classification

Hierarchical image classification tasks intrinsically require multi-level predictions per instance which correspond to a hierarchical label tree. Note that hierarchical multi-label classification can be viewed as a generalisation of multi-class problems with a hierarchy rather than exclusive classes. As a result, it has attracted considerable research attention and remains a challenging problem in both machine learning and pattern recognition. Note that the hierarchical multi-label classification problem can be viewed as a tree or direct acyclic graph (DAG) problem [22] depending on the hierarchical structure of the target labels. Recall that, according to Silla and Freitas [22], hierarchical classifiers can be grouped into flat, local or global ones. It is worth noting that, viewed in this manner, the method presented here would be considered a global one where a single classifier model is learnt from the training set. Furthermore, as related to the classifier model itself, a number of methods have been proposed to address hierarchical classification tasks, ranging from kernel methods [23] to decision trees [24] and, more recently, artificial neural networks [25]. For a detailed survey of hierarchical classification, the interested reader can go to [22].

Note that, despite the fact that image hierarchical classification has been applied to the annotation of medical images [26], these methods are typically less concerned with images and instead focus on other data modalities corresponding to scopes of application such as protein structure prediction [24], data-dependent grouping [27] or text classification [28]. This is somewhat surprising considering the notion that incorporating hierarchies in the classification model is expected to allow it to generalise better, a trait that is particularly relevant to image classification tasks. This is since these semantic relationships of the target classes can be used as a guide for the classifier when utilising hierarchical methods for image classification. Along these lines, word hierarchies have been applied to provide consistency across multiple datasets [29], provide a post-conviction estimate [30] and optimise the tradeoff between accuracy and specificity in visual recognition [31].

As applied to image data, hierarchical classification models often employ convolutional neural networks (CNNs). This is since these can be viewed as naturally hierarchical [32], whereby the early layers of a CNN architecture can be related to the coarse levels in the hierarchy and deeper layers to the finer categories. Yan et al. [6] propose a hierarchical deep CNN (HD-CNN) by applying convolutional networks to a category hierarchy. Their method first separates the clearly separable classes into coarse categories and then more challenging classes are routed to the fine categories for prediction. Thus, in [6] the coarse-to-fine paradigm is used to improve the model performance making use of the hierarchical structure. Zhu and Bain [7] relate the network layers to the hierarchical levels on a label tree. To do this, they note that the top-level hierarchies are expected to account for features common to finer categories in the label tree, whereby finer classification often requires more class-specific features. In a similar fashion, [33] proposes a convolutional architecture aimed at classifying apparel categories reflecting a hierarchical structure. Further, in [34], the authors propose a hierarchical multi-label classification method for fashion images called Condition-CNN, which is based on the idea of conditioning the output of a CNN on the hierarchical structure of the labels. Recently, Dhall et al. [35] have studied order-preserving embeddings for modelling hierarchical semantic structures employing label-to-label and image-to-image hierarchical relations.

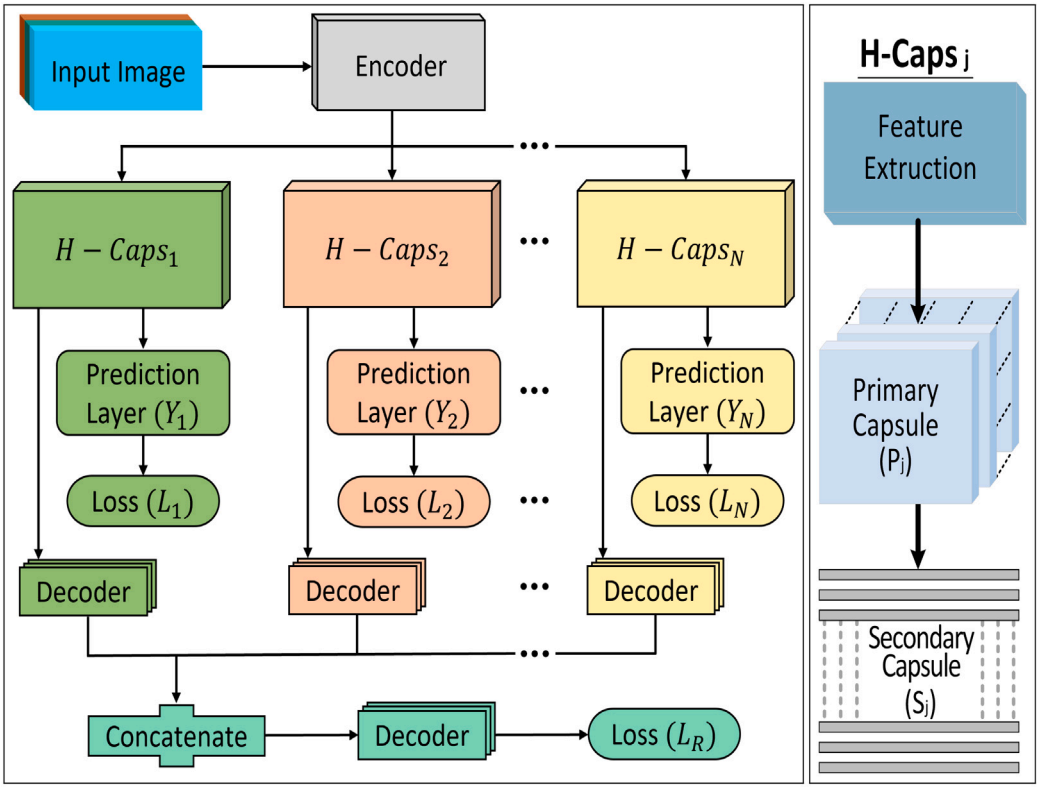

Fig. 1. (a) Architecture of our Hierarchical Capsule Network (H-CapsNet); (b) The architecture for each of the H-Caps block. Note the network is comprised of several branches, whereby each H-Caps block has dedicated capsules for each of the classes at each hierarchy level. The decoder uses the class predictions to reconstruct the encoded data so as to apply the reconstruction loss. Both, the decoder and encoder are comprised by convolutional nets as described in Section 4.2.

3. Hierarchical capsule networks

As mentioned earlier, our H-CapsNet uses a capsule architecture to obtain a multi-label prediction based on a hierarchical/taxonomical label tree. In Fig. 1(a), we show both, the architecture of our network and that of the H-Caps blocks that comprise it. Note that, our H-CapsNet has an encoder block designed to meet the needs of converting pixel intensities into features that can then be utilised as the inputs for the subsequent blocks in the network. This structure is quite general in nature whereby these encoders are comprised by a combination of convolutional, batch normalisation and pooling layers which can be adapted and modified depending on the complexity of the dataset. In Section 4.2 we detail the structure of these for the datasets under consideration in our experimental study.

In the figure, the $H-Caps_j$ blocks corresponding to the hierarchical levels are capsule networks themselves, whereby each of these account for the classes present at the jth hierarchical level in the label tree. In this manner, the $H-Caps_1$ refers to the coarsest class-level, whereas the $H-Caps_n$ corresponds to the finest out of n hierarchical levels. Thus, each of the hierarchical levels from coarse-to-fine, has its own capsule networks and, therefore there are as many of these as hierarchies. Our H-CapsNet employs a "coarse-to-fine" scheme based upon multiple capsule layers for each level on the hierarchical label tree. These branches follow the coarse-to-fine paradigm whereby all branches are used for the reconstruction term of the total loss. This allows the finer branches to leverage the features learned by the coarser ones, improving prediction accuracy. The loss used here is described in Section 3.2.

Note that, in a hierarchical-label tree, coarse level classes are a superclass of all the medium and fine classes. Conversely, medium classes are a superclass of all the fine classes and so on. This hints at the notion that each H-Caps block should share features which are common to several hierarchy levels. Thus, in our architecture, the encoder is common, as shown in the figure, where each H-Caps block in the network has its own prediction layer. In the figure, these are denoted as $\mathcal{Y}_j$ and estimate class probabilities at the corresponding jth class-hierarchy level making use of a logit function. Thus, when performing image classification, the prediction layers $\mathcal{Y}_j$ deliver the probabilities for the $\mathcal{K}_j$ classes at the jth hierarchical level in the label tree.

Recall that CapsNets [15] are trained using a hinge and a reconstruction loss. This reconstruction loss allows for the capsule network to encode the input's instantiation parameters. As a result, we make use of an auxiliary decoder network to reconstruct the input image while training the model. The decoder used here employs the predictions delivered by the layers $\mathcal{Y}_j$ and the activity vector outputs from the H-Caps blocks for each hierarchy. It is also worth noting that, as shown in Fig. 1(b), the decoder for the reconstruction loss takes at input the H-Caps block outputs via concatenation. This concatenation is a straightforward step for tree-structured label sets. For labels structured as direct acyclic graphs, the concatenation step is expected to take into account the variation in the number of outputs passed on by the decoders corresponding to each of the H-Caps blocks. This is achieved by introducing a padding operation. Regarding the decoders themselves, these are comprised by convolutional networks whose structure is detailed in Section 4.2.

3.1. Capsule network blocks

In our network, each of the capsule networks ($H-Caps$) blocks share the same structure. In this manner, each of the $H-Caps$ blocks has a feature extraction stage so as to extract features specific to the corresponding level under consideration in the hierarchy tree under

study. Each feature extraction block contains multiple convolutional layers, a batch normalisation and a max pooling layer with unique parameter settings according to the hierarchical levels of the dataset.

These features are then delivered to the primary capsules in the $H-Caps$ block. Thus, the feature map for the hierarchy level indexed j is reshaped so as to fit the input shape of the primary capsule network $\mathcal{P}_j$. Note the primary capsules used here are comprised by a convolutional capsule layer with r $n-dimensional$ vectors. The value of r and n is dependent upon the dataset and hierarchy under consideration. The output vector from primary capsules $\mathcal{P}_j$ is then fed into the secondary capsules $\mathcal{S}_j$ as shown in Fig. 1. Note that each $\mathcal{S}_j$ comprises $\mathcal{K}_j$ capsules, where each of these accounts for a class label in the same jth hierarchical level. Hence, there are as many of these as classes in the hierarchical level under consideration, whereby each capsule network $\mathcal{K}_j$ outputs a classification prediction vector for the jth level in the label-tree.

3.2. Loss function

The loss function for our H-CapsNet is a weighted summation of all the classification losses across the label-tree and the reconstruction loss. In this manner, the total loss for our network becomes

$$L_T = \lambda L_R + L_C \tag{1}$$

where L_R is the reconstruction loss, L_C is the classification loss given by

$$L_C = \sum_{j=1}^{N} \gamma_j L_{M_j} \tag{2}$$

and λ is a constant that controls the influence of the classification loss upon the total loss.

In Eq. (2), γ_j is a weight that adjusts the contribution of each class hierarchy to the overall loss and L_{M_j} denotes the hinge loss for the jth H-Caps block, which is expressed as

$$L_{M_j} = T_{k,j}\max(0, m^+ - \left\|v_{k,j}\right\|)^2 + \eta(1-T_{k,j})\max(0, \left\|v_{k,j}\right\| - m^-)^2 \tag{3}$$

where, $T_{k,j} = 1$ if class k belongs to the jth level in the hierarchy and $T_{k,j} = 0$ otherwise, m^+, m^- are hyper-parameters, η is the down-weighting of the loss and $v_{k,j}$ is the output vector of the secondary capsule corresponding to the kth class.

In this manner, the classification loss becomes a linear combination of the hinge losses for each of the N hierarchical levels in the label-tree. Moreover, the weights γ_j and λ regulate the balance between the hinge and reconstruction losses. Here we employ the $L-2$ norm between the input instance x and the reconstructed one $\hat{x}$ yielded by the final decoder in Fig. 1 for the reconstruction loss in Eq. (1). Therefore, the reconstruction loss becomes

$$L_R = \|x - \hat{x}\|_2^2 \tag{4}$$

3.3. Dynamic loss weights

Here, we also note that the weights γ_j used in the computation of the loss function L_C can be used to improve the model accuracy. This hinges in the notion that, since these control the contribution of the classification losses to the total loss, they can be used to balance the influence of each of these as related to the label-tree during training.

Thus, here we employ the training accuracy to compute the values of γ_j. We also note that the value of all the weights γ_j and λ should be equal to unity. Further, the values of γ_j should account for the number of classes in each hierarchy so as to balance these appropriately. As a result, we commence by computing the ratio of the number of classes K_j at the hierarchy level j to the total number of classes in the label tree. This is given by

$$\varrho_j = \frac{\left|K_j\right|}{\sum_{i=1}^{N}\left|K_i\right|} \tag{5}$$

With the ratio above in hand, we also note that, as the training progresses, some hierarchy levels will train at different rates, whereby we can use the training accuracy to govern the influence of the hinge loss for each level accordingly. Thus, we define the quantity

$$\tau_j = \left(1 - Acc_j\right)\varrho_j \tag{6}$$

where Acc_j is the value of the training accuracy for the jth hierarchy in the label-tree.

Note that τ_j is expected to be greater when the accuracy Acc_j is small. Thus, it can be used to increase the influence of the loss by setting the weight γ_j. As a result, we compute the classification loss weights as follows

$$\gamma_j = (1-\lambda)\frac{\tau_j}{\sum_{i=1}^{N}\tau_i} \tag{7}$$

where the term $(1-\lambda)$ accounts for the notion that the sum of all weights γ_j and λ should add up to unity and we have added the denominator for normalisation purposes.

4. Experiments

We have conducted numerous experiments to evaluate our H-CapsNet model. To this end, we have used four widely available datasets and compared our results to other methods elsewhere in the literature. We have also performed an ablation study.

4.1. Data-sets

As mentioned above, for evaluating our H-CapsNet model we have used several datasets. These are the Fashion-MNIST [36], Marine-tree [37], CIFAR-10 and CIFAR-100 [38] datasets. The Fashion-MNIST dataset contains 60,000 training and 10,000 testing grey-scale images depicting fashion items. Each example in the original dataset is a 28×28 greyscale image corresponding to one out of ten classes. In our experiments, we have used the same label-tree as that employed in [33], where the authors use three hierarchical levels. These are a coarse level comprised by two classes and a medium one composed of 6 classes. In Fig. 2, we show the hierarchical label-tree used here for the Fashion-MNIST dataset.

The Marine-tree dataset [37][1] contains 160,000 images of marine organisms divided into tropical and temperate subsets according to the climate. The dataset provides five hierarchical levels with a total of 161,185 images, where 118,260 were used for training, 16,127 for validation and 26,798 for testing. For the sake of consistency, in all our experiments, we employed the Marine-tree dataset with the first three hierarchical levels combining the tropical and temperate subsets into one. For these three hierarchical levels, the coarse one has two classes, the medium one has 10 and the fine one has 38 classes.

The CIFAR-10 and CIFAR-100 datasets consist of 60,000 32×32 colour images. There are 50,000 training images and 10,000 testing images in both datasets. For both datasets, we have used the hierarchies presented in [7]. As a result, both datasets are organised in a three-level label-tree, which, for the CIFAR-10 comprises a medium and a coarse level composed of 7 and 2 classes, respectively. In Fig. 3, we show the label-tree used here for the CIFAR-10 dataset. Since the CIFAR-100 has 100 fine classes grouped into 20 superclasses, Zhu and Bain [7] employ these as fine and medium levels in the label-tree and add a coarse one containing 8 classes in order to employ a three-level label-tree. Further, based on the CIFAR-10 dataset, we have also created a direct acyclic graph (DAG) dataset.[2] To do this, we have modified the CIFAR-10 label-tree by adding a multiple parent node for some fine level classes. In this manner, we have created a DAG dataset with 10 fine classes, 7 medium classes and 2 coarse classes.

[1] The Marine Tree dataset is widely available at https://github.com/tboone91/Marine-tree.

[2] The DAG version of the CIFAR-10 dataset is available at https://github.com/tasrif-khondaker/H-CapsNet.

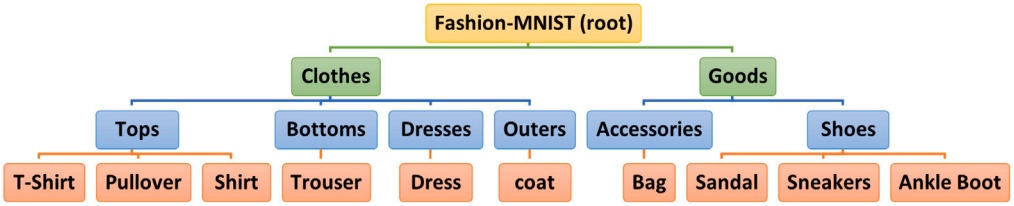

Fig. 2. Hierarchical label-tree for the Fashion-MNIST [36] dataset used in our experiments. The ten fine classes are grouped into 7 medium and two coarse ones as proposed in [33].

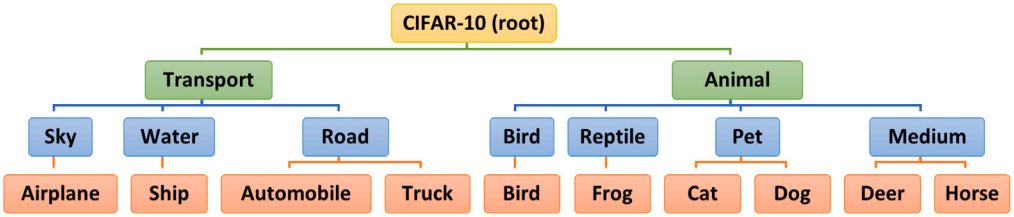

Fig. 3. Hierarchical label tree for the CIFAR-10 [38] dataset used in our experiments. In addition to the provided ten fine classes, the authors in [7] use 7 medium and two coarse ones.

4.2. Experimental setup

In all our experiments we normalise the training and testing data by subtracting the mean and dividing by the standard deviation. Furthermore, during training, in all our experiments we have applied the MixUp data augmentation technique [39] with an alpha value of 0.2 to all the models under consideration. Here we use the Adam optimiser with an exponentially decaying learning rate given by

$$\eta = \hat{\eta}\beta^{\max(0,E-\kappa)} \tag{8}$$

where β and κ are constants and E is the epoch number. In all our experiments we set the initial learning rate $\hat{\eta}$ to 0.001, β to 0.95 and κ to 10.

We have implemented our approach on TensorFlow/Keras.[3] The common encoder block in our network, which is shared amongst all the H-Caps blocks, consists of two convolutional layers followed by a batch normalisation layer. These convolutional layers are comprised by 32 and 64 filters for the fashion-MNIST dataset and 64 filters for the two layers for the other datasets. These make use of 3×3 filters with zero-padding with a stride value of 1 and $ReLu$ activation functions.

For all our results we have employed $8-Dimensional$ primary capsules $\mathcal{P}_j$ and $16-Dimensional$ secondary capsules $\mathcal{S}_j$ with dynamic routing [8] using 2 iterations between the capsule layers. The Decoder network for all hierarchical levels in the H-CapsNet model uses three sequential dense layers and a reshape layer. The first layer has 512 neurons, and the second layer has 1024 neurons with both layers using the $ReLu$ activation functions. Note that the number of neurons in the third layer of the decoder network corresponds to the input image dimensionality with a $sigmoid$ activation function.

As mentioned earlier, we configure the H-CapsNet model architecture based on both, the hierarchical levels and the dataset complexity. Like the encoder block, the feature extraction block in the H-Caps block is comprised by a set of convolutional layers followed by batch normalisation and max pooling layers. As said earlier, we have adjusted the number of layers in the feature extraction block by considering the complexity of the hierarchical levels in the datasets. Thus, for the Fashion-MNIST dataset, the feature extraction block at the coarse level in the hierarchy is comprised of a single convolutional layer with 512 filters and a kernel size of 7×7 followed by a batch normalisation layer. The medium and fine level H-Caps block feature extractors are comprised by three convolutional layers with 128, 256 and 512 filters, respectively, all of which have a kernel size of 3×3. For the CIFAR-10, CIFAR-100 and Marine-Tree datasets, the coarse level feature extractor has two convolutional layers with 128 filters. The medium level has 4 convolutional layers, and the fine class-level has 6 convolutional layers, all in pairs with a max-pooling layer with a stride set to 2×2 between each of these. These pairs have 128, 256 and 512 filters for the fine class-level and 128 and 256 for the medium one. All the kernel sizes have been set to 3×3 and $ReLu$ activation functions. Also, each convolutional layer is followed by a batch normalisation layer with TensorFlow's default settings.

In all our experiments we have set the hyper-parameters m^+, m^-, η value to 0.9, 0.1 and 0.5, respectively. We have set the loss weight λ in Eq. (1) to 0.0005 and have trained the H-CapsNet model for 100 epochs for Fashion-MNIST, Marine-tree, CIFAR-10 and CIFAR-100 datasets. In all our experiments these values were found by cross validation.

Table 1
Accuracy yielded by our H-CapsNet employing the dynamic loss weight distribution system and fixed loss weight values. The absolute best are in bold.

Dataset	Dynamic loss weight			Fixed loss weight		
	Coarse	Medium	Fine	Coarse	Medium	Fine
Fashion-MNIST	**99.73%**	**97.06%**	**93.95%**	99.69%	96.64%	93.09%
CIFAR-10	**97.67%**	**92.90%**	**91.41%**	97.26%	92.04%	90.65%
CIFAR-10[a]	**94.39%**	**88.06%**	**91.59%**	93.98%	87.20%	90.83%
CIFAR-100	**80.00%**	**77.02%**	**67.86%**	79.86%	75.48%	65.21%
Marine-tree	**88.38%**	**79.49%**	**62.44%**	89.64%	77.20%	56.28%

[a] Denotes the DAG version of the CIFAR-10 dataset.

4.3. Ablation study

As said previously, we have also conducted an ablation study using the four datasets under consideration. This consists in ablating the encoder and feature extraction blocks from our H-CapsNet architecture so as to better understand the contribution of these to the model performance. Additionally, we have conducted experiments where the dynamic loss weights have been replaced with fixed values. Finally, in order to analyse the contribution of the reconstruction loss to our training step, we have trained our H-CapsNet model with a loss devoid of the term L_R in Eq. (1). We do this by setting the reconstruction loss weight λ to 0. This implies that the overall loss function depends on

[3] The implementation of H-CapsNet is available on https://github.com/tasrif-khondaker/H-CapsNet .

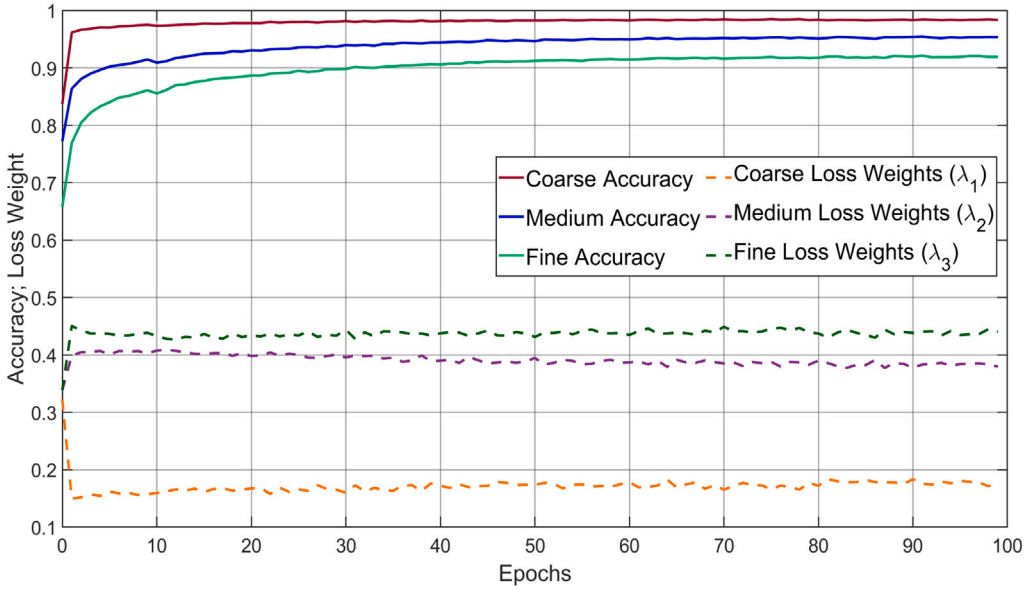

Fig. 4. Accuracy and loss weight values as a function of training epoch for the H-CapsNet model trained using the Fashion-MNIST dataset.

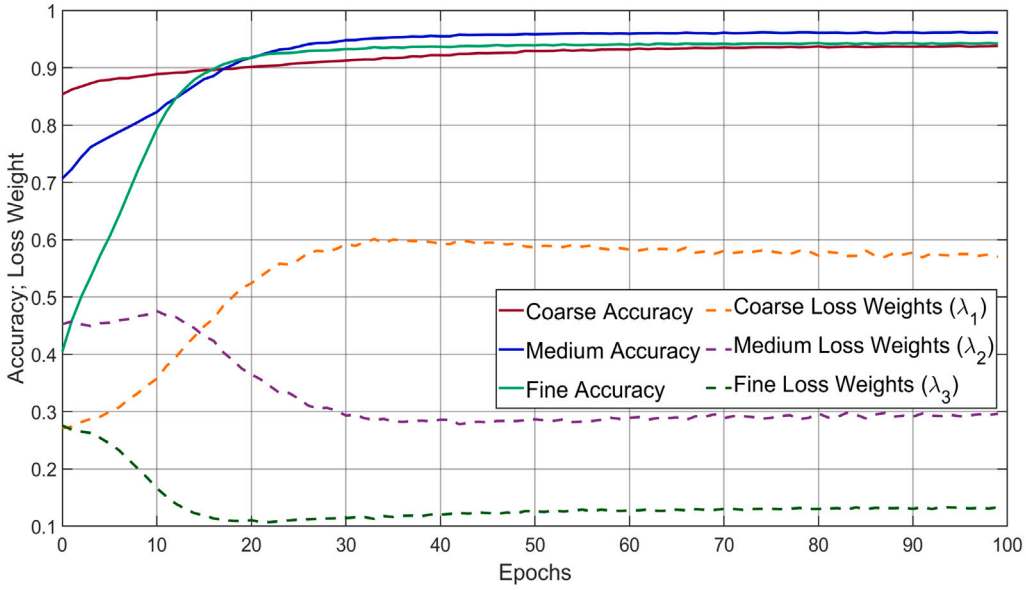

Fig. 5. Accuracy and loss weight values as a function of training epoch for the H-CapsNet model trained using the Marine-tree dataset.

the classification loss L_C whereby the loss weights γ_j are computed by setting $\lambda = 0$ in Eq. (7).

Thus, we have trained the model with all the combinations arising from the ablation of both, the encoder block and the feature extraction blocks. For the purpose of analysing the model by ablating the encoder block, we have directly fed the input image to the dedicated feature extraction blocks for each hierarchical level. Similarly, for ablating the feature extraction blocks from our model architecture, we have reshaped the output from the common encoder block to allow for it to be used as input by the primary capsules corresponding to each of the hierarchical levels in the label-tree. In the case where both the encoder and feature extraction blocks have been removed from the model, we directly feed the input instances to the primary capsules. For the sake of fairness, all the training hyperparameters are kept the same for the ablation study.

For the experiments involving the dynamic loss weights, we have trained our model by fixing these and noted that, as shown in Eq. (7), the dynamic loss weight distribution is based on both, the number of classes at each hierarchy level in the label-tree and the accuracy values, being the latter the ones that permit the weight to "adjust" to training performance. Therefore, in order to train the model with fixed loss weight values, we have only considered the classes in each level of the label-tree, computing the loss weight values by setting $\tau_j = \varrho_j$ in Eq. (7). This effectively bypasses the computation in Eq. (6). Thus, the loss weight values remain constant for each of the hierarchy levels while maintaining the balance with respect to the relative number of classes in the label-tree.

4.4. Results and discussion

We now focus on the results yielded by our network and a number of alternatives when applied to the four datasets under consideration. In our results, we have used as a baseline the CapsNet as originally proposed in [8] and, for purposes of comparison, we have also performed experiments using the HD-CNN in [6], the B-CNN [7] and the

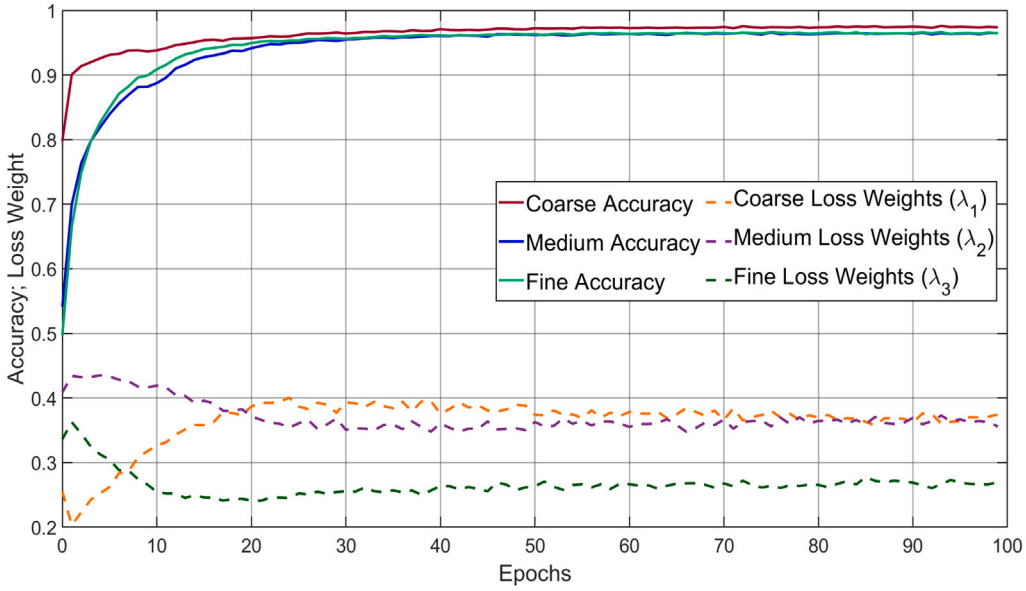

Fig. 6. Accuracy and loss weight values as a function of training epoch for the H-CapsNet model trained using the CIFAR-10 dataset.

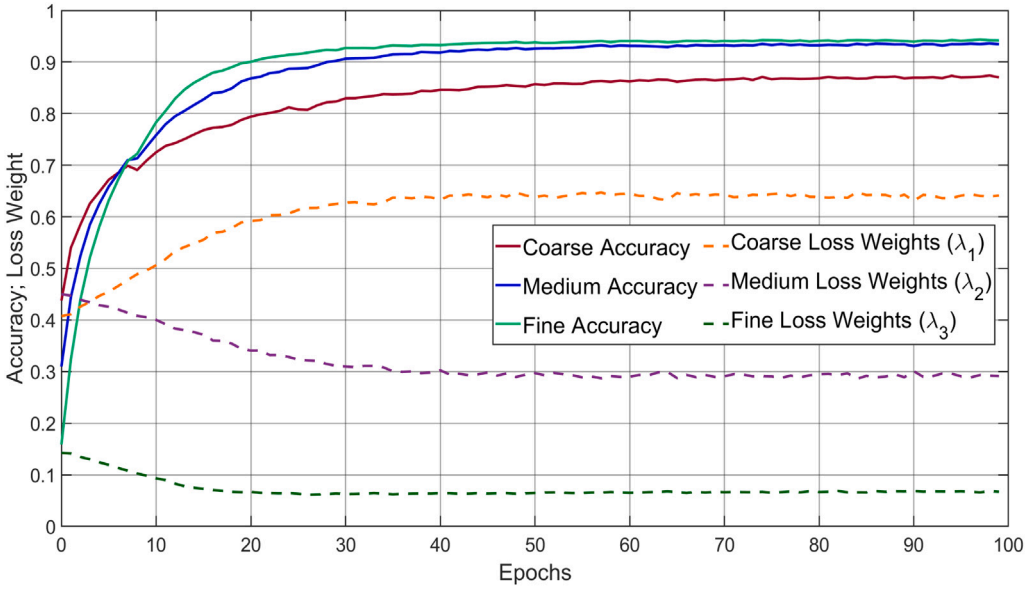

Fig. 7. Accuracy and loss weight values as a function of training epoch for the H-CapsNet model trained using the CIFAR-100 dataset.

Condition-CNN [34]. In all our experiments, the B-CNN model learning rates and loss weights have been set as stated in [7]. The B-CNN model used here is the Base B model [7]. The architecture of the Condition-CNN model used here is the same as the one described in [34], and we have trained the model from scratch without utilising any pre-trained weights.

We commence by comparing the effects of fixing the loss weights rather than employing the dynamic approach presented in Section 3.3. In Table 1, we present the model classification accuracy for both cases, i.e. fixed and dynamic loss weights. Note that, in the table, the dynamic loss weight approach achieves better classification accuracy for all the applicable levels across the four datasets. This is since the dynamic loss weight values deliver a better balance between the level-specific losses. In Figs. 4–7 we show the model accuracy and corresponding loss weight value for each of the class-hierarchies as a function of epoch number. It is worth observing that, in Fig. 4, the loss weight values for the Fashion-MNIST dataset are greater for the fine level, followed by those for the medium and coarse level hierarchies. This is due to the fact that, throughout the training, the classification accuracy dominates from coarse-to-fine. As a result, the model prioritises fine-levels over the coarse and medium ones. In contrast, in Fig. 5, the coarse-level and medium-level loss weight values for the marine-tree dataset switch places when the model achieves higher medium-level accuracy. The model then distributes the loss weights as performance improves. This trend is also present in the plots for the CIFAR-10 and CIFAR-100 datasets, where the coarse level weight increases only after the medium and fine level accuracy improve.

In Table 2, by showing the accuracy yielded by the models under consideration when applied to the Fashion-MNIST and Marine-Tree datasets. These include the ablated variants of our approach and the alternatives. Similarly, in Table 3, we show the accuracy of our network and the alternatives when applied to the CIFAR-10 and CIFAR-100 datasets. In the tables, we show the accuracy for all the applicable levels of the class-hierarchies for our networks, as well as for the B-CNN and Condition-CNN approaches. We also show the performance of the CapsNet in [8] on the fine class-hierarchy and HD-CNN in [6] on the

Table 2
Accuracy yielded by our H-CapsNet, the CapsNet in [8], the HD-CNN [6], the B-CNN [7] and the Condition-CNN [34] when applied to all the datasets. The absolute best are denoted in bold.

Dataset	Level	Accuracy (%)				
		CapsNet	HD-CNN	B-CNN	Condition-CNN	H-CapsNet
Fashion-MNIST	Coarse	–	–	**99.80**	99.78	99.73
	Medium	–	94.86	96.51	96.65	**97.06**
	Fine	91.20	90.46	93.52	93.42	**93.95**
CIFAR-10	Coarse	–	–	96.08	95.86	**97.67**
	Medium	–	85.08	87.13	83.78	**92.90**
	Fine	70.42	79.96	84.54	79.74	**91.41**
CIFAR-10[a]	Coarse	–	–	81.55	58.28	**94.39**
	Medium	–	87.60	14.98	32.43	**88.06**
	Fine	70.42	85.00	62.72	74.67	**91.59**
CIFAR-100	Coarse	–	–	71.08	73.38	**80.00**
	Medium	–	65.44	61.99	61.27	**77.02**
	Fine	34.93	48.39	56.38	47.91	**67.86**
Marine-tree	Coarse	–	–	88.28	**88.75**	88.38
	Medium	–	60.06	75.88	76.64	**79.49**
	Fine	46.73	45.62	54.48	53.99	**62.44**

[a] Denotes the DAG version of the CIFAR-10 dataset.

Table 3
Accuracy yielded by our H-CapsNet and its ablation versions when applied to all the datasets. The absolute best are denoted in bold. Here, the following shorthands apply: w/ = with, w/o = without, ENC = Encoder block, FE = Feature Extraction block, L_R = Reconstruction Loss in Eq. (1). Here, * denotes the DAG version of the CIFAR-10 dataset.

Dataset	Level	Accuracy (%)				
		H-CapsNet	H-CapsNet w/ ENC, w/o FE	H-CapsNet w/o ENC, w/ FE	H-CapsNet w/o ENC, w/o FE	H-CapsNet w/ Lr = 0
Fashion-MNIST	Coarse	99.73	99.72	99.41	98.77	**99.74**
	Medium	**97.06**	95.77	95.71	89.99	96.45
	Fine	**93.95**	91.90	93.07	82.19	93.54
CIFAR-10	Coarse	**97.67**	93.47	92.50	84.39	97.01
	Medium	**92.90**	75.12	89.29	57.14	92.42
	Fine	**91.41**	69.24	90.45	50.08	91.12
CIFAR-10[a]	Coarse	**94.39**	90.19	89.22	81.11	93.73
	Medium	**88.06**	70.28	84.45	52.30	87.58
	Fine	**91.59**	69.42	90.63	50.26	91.30
CIFAR-100	Coarse	**80.00**	65.35	66.06	44.31	79.35
	Medium	**77.02**	51.99	72.48	28.81	75.61
	Fine	**67.86**	38.20	65.86	19.41	65.71
Marine-tree	Coarse	**88.38**	87.11	85.88	86.13	88.00
	Medium	**79.49**	71.76	77.27	70.95	77.95
	Fine	**62.44**	43.06	54.77	39.43	56.75

[a] Denotes the DAG version of the CIFAR-10 dataset.

medium and fine class-hierarchies. We do this since the baseline is not a hierarchical classification one, rather a "flat" classifier as originally proposed in [8] and, therefore, the medium and coarse label hierarchies do not apply. In the other hand, the HD-CNN in [6] employs a two-level hierarchy with pre-training for each level before fine-tuning the complete network.

Note that, for all our experiments, our proposed H-CapsNet model achieved a margin of improvement over the alternatives, outperforming all the other methods under consideration. This trend is confirmed in Figs. 8–11, which show the fine-level classification accuracy for our H-CapsNet model, its ablated variants and the alternatives as a function of training epoch for the datasets under consideration. In the plots, we show the finer level of the class-label tree for our method, the B-CNN [7] and the Condition-CNN [34]. We have done this following the notion that fine level classification is the most challenging of all levels, consistently exhibiting the lowest performance for all classifiers and datasets. Following the authors, the HD-CNN model accuracy shown in the plots is calculated using the probabilistic averaging function in [6].

On the results, it is also worth noting that, as compared with the baseline, our approach improvement is more evident for the Marine-tree, CIFAR-10 and CIFAR-100 datasets, which are more challenging than the Fashion-MNIST dataset. Indeed, for the Marine-tree, CIFAR-10 and CIFAR-100 datasets, the improvement in convergence for our approach is more evident too. This trend is also consistent across the other datasets under consideration.

It is worth observing that, in Tables 2 and 3, the accuracy of our H-CapsNet model drops without the encoder and the dedicated feature extraction blocks. This is since the encoder block in our H-CapsNet architecture learns the features that are particularly useful for coarse classification. This can be observed in Tables 2 and 3, where the accuracy for the H-CapsNet without the encoder block drops for the coarse level accuracy as compared to our network without the dedicated feature extraction blocks. In the same way, the dedicated feature extraction blocks for different levels are crucial for extracting more level-specific features. Our H-CapsNet without these dedicated feature extraction blocks exhibits a drop in accuracy that is more pronounced for medium and fine level classification. This accuracy drop is again more apparent for the Marine-tree, CIFAR-10 and CIFAR-100 datasets. Further, from Figs. 9–11 we can observe there is a large difference between our H-CapsNet and that without the dedicated feature extraction and encoder blocks. Further, in Tables 2 and 3, the H-CapsNet model performance without the reconstruction loss states

Table 4
Precision, recall and F1-score yielded by our H-CapsNet, the CapsNet [8], the HD-CNN [6], the B-CNN [7] and the Condition-CNN [34] on the datasets under consideration.

Model	Level	Metrics	Fashion-MNIST	Marine-tree	CIFAR-10	CIFAR-10[a]	CIFAR-100
CapsNet	Fine	Precision	91.16%	38.40%	70.14%	70.14%	33.71%
		Recall	91.20%	46.73%	70.42%	70.42%	34.93%
		F1-Score	91.17%	41.24%	70.26%	70.26%	33.83%
HD-CNN	Medium	Precision	94.87%	75.00%	85.31%	88.10%	65.65%
		Recall	94.87%	60.06%	85.08%	87.64%	65.44%
		F1-Score	94.85%	65.51%	85.08%	87.66%	65.09%
	Fine	Precision	90.35%	52.84%	80.08%	85.23%	48.56%
		Recall	90.46%	45.62%	79.96%	84.99%	48.39%
		F1-Score	90.30%	47.44%	79.96%	84.89%	47.29%
B-CNN	Coarse	Precision	**99.80%**	86.81%	96.08%	82.10%	71.91%
		Recall	**99.80%**	88.28%	96.08%	81.55%	71.08%
		F1-Score	**99.80%**	85.69%	96.07%	81.47%	70.67%
	Medium	Precision	96.52%	73.28%	87.21%	28.39%	63.03%
		Recall	96.51%	75.88%	84.56%	14.98%	61.99%
		F1-Score	96.51%	71.43%	87.12%	10.39%	62.30%
	Fine	Precision	93.51%	53.38%	84.56%	63.14%	57.29%
		Recall	93.51%	54.48%	84.54%	62.72%	56.38%
		F1-Score	93.51%	52.88%	84.54%	62.89%	56.64%
Condition-CNN	Coarse	Precision	99.78%	87.56%	95.86%	64.67%	73.49%
		Recall	99.78%	**88.75%**	95.86%	58.28%	73.38%
		F1-Score	99.78%	**87.86%**	95.85%	53.18%	73.31%
	Medium	Precision	96.66%	74.46%	83.79%	19.61%	61.99%
		Recall	96.65%	**76.64%**	83.78%	32.43%	61.27%
		F1-Score	96.65%	**75.05%**	83.75%	21.69%	61.50%
	Fine	Precision	93.42%	50.80%	80.14%	74.36%	49.94%
		Recall	93.42%	53.99%	79.74%	74.67%	47.91%
		F1-Score	93.42%	51.78%	79.84%	68.64%	48.48%
H-CapsNet	Coarse	Precision	99.73%	**87.64%**	**97.67%**	**94.30%**	**80.01%**
		Recall	99.73%	88.38%	**97.67%**	**94.50%**	**80.00%**
		F1-Score	99.73%	85.28%	**97.67%**	**94.39%**	**79.91%**
	Medium	Precision	**97.06%**	**81.24%**	**92.92%**	**89.52%**	**76.95%**
		Recall	**97.06%**	35.40%	**92.90%**	**88.06%**	**77.02%**
		F1-Score	**97.06%**	49.31%	**92.88%**	**88.01%**	**76.92%**
	Fine	Precision	**93.96%**	**65.36%**	**91.39%**	**91.56%**	**67.62%**
		Recall	**93.95%**	**62.44%**	**91.41%**	**91.59%**	**67.86%**
		F1-Score	**93.95%**	**60.56%**	**91.38%**	**91.55%**	**67.50%**

[a] Denotes the DAG version of the CIFAR-10 dataset.

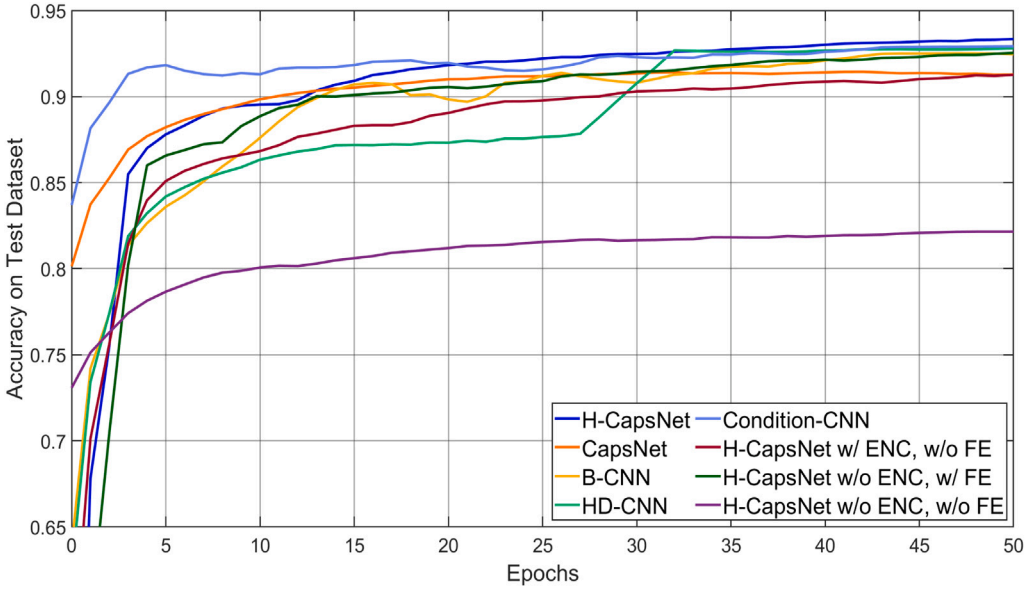

Fig. 8. Accuracy as a function of training epoch for all the models under consideration for the Fashion-MNIST dataset. The following shorthands apply: w/ = with, w/o = without, ENC = Encoder block, FE = Feature Extraction block.

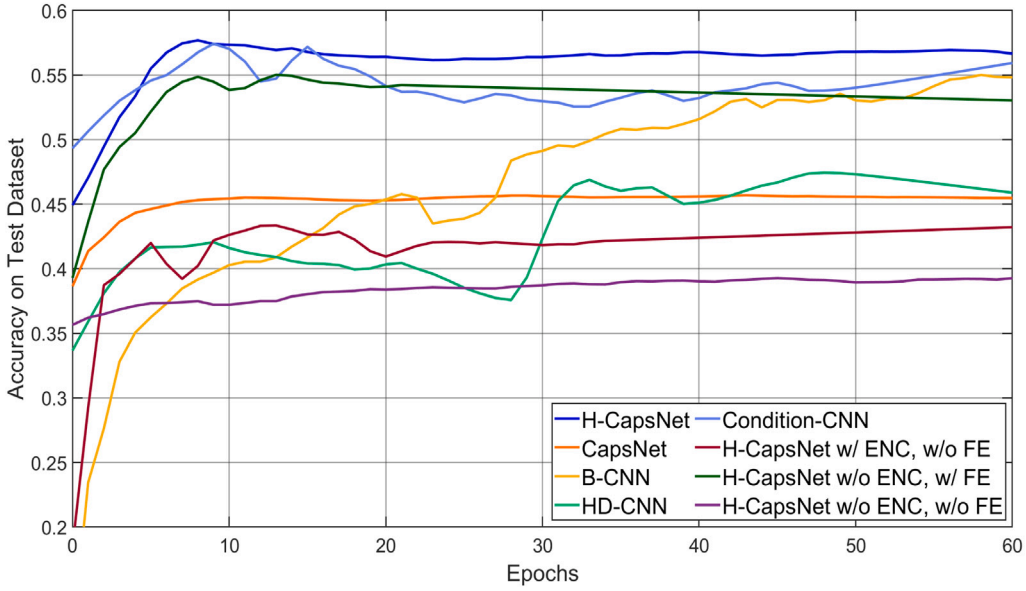

Fig. 9. Accuracy as a function of training epoch for all the models under consideration for the Marine-tree dataset. The following shorthands apply: w/ = with, w/o = without, ENC = Encoder block, FE = Feature Extraction block.

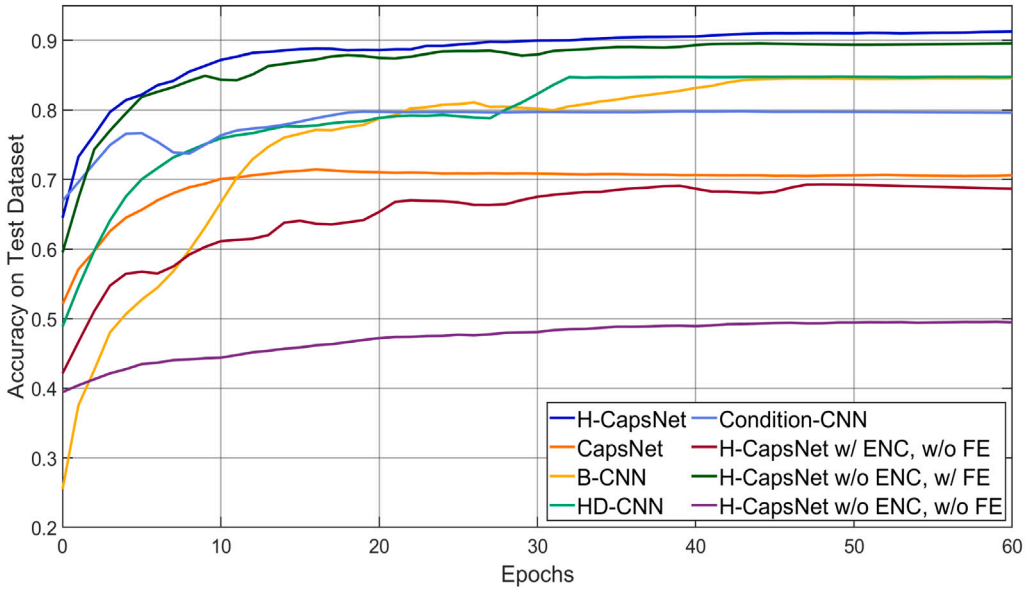

Fig. 10. Accuracy as a function of training epoch for all the models under consideration for the CIFAR-10 dataset. The following shorthands apply: w/ = with, w/o = without, ENC = Encoder block, FE = Feature Extraction block.

that the reconstruction part of the H-CapsNet model helps to improve the overall model performance. This difference in model performance is more apparent on the finer levels.

We now examine the precision, recall and F1-scores yielded by our method and the alternatives. In Table 4, we show the results of these performance metrics delivered by our H-CapsNet, the CapsNet in [8], the HD-CNN in [6], the B-CNN in [7] and Condition-CNN in [34] for each of the class-levels for the four datasets under consideration. In order to compute the performance metrics for each level, we have calculated these metrics for each class in the label-tree utilising the confusion matrix[4] corresponding to each level and then proceeded to compute their average. Note that our datasets exhibit hierarchical level-imbalances with different number of instances per class. Thus, here we compute the weighted average considering the number of instances per class per hierarchical level. Note that in Table 4, our model consistently outperforms the alternatives by achieving overall higher precision, recall and F1-scores. Again, our model margin of improvement is more evident on the CIFAR-10 and CIFAR-100 datasets. On the Fashion-MNIST and Marine-tree dataset, the B-CNN approach in [7] and Condition-CNN in [34] performed slightly better than our method for coarse and medium classification, but for the fine levels, our model still delivers the best performance.

Finally, we evaluate our approach and the alternatives making use of the hierarchical metrics in [40]. These metrics consider the entire set of predicted classes for each instance whereby these are expected to be consistent with the targets across the superclasses in the label-tree [41]. In contrast to the other evaluation metrics used here previously, hierarchical precision (HP) and hierarchical recall (HR) are

[4] The confusion matrices for our experiments are available at https://github.com/tasrif-khondaker/H-CapsNet.

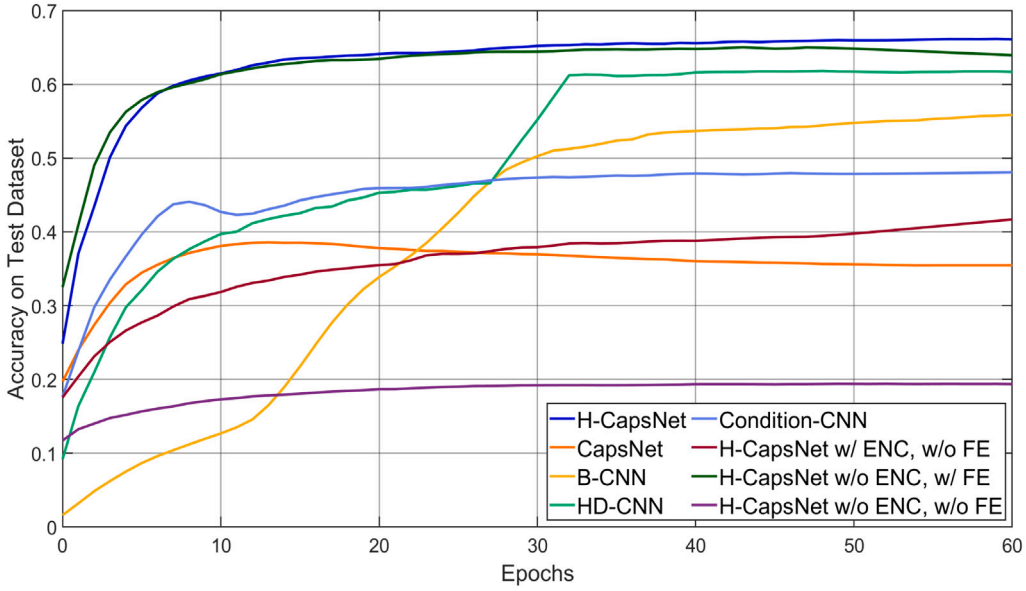

Fig. 11. Accuracy as a function of training epoch for all the models under consideration for the CIFAR-100 dataset. The following shorthands apply: w/ = with, w/o = without, ENC = Encoder block, FE = Feature Extraction block.

Table 5
Hierarchical performance metrics for our H-CapsNet, the CapsNet [8], the HD-CNN [6], the B-CNN [7] and the Condition-CNN [34] on the datasets used here. The absolute best are denoted in bold. The following shorthands apply: HP = Hierarchical Precision, HR = Hierarchical Recall, HF1 = Hierarchical F1-Score, Cons = Hierarchical Consistency, EM = Exact Match.

Model	Metrics	Fashion-MNIST	Marine-tree	CIFAR-10	CIFAR-10[a]	CIFAR-100
CapsNet	HP	91.20%	46.73%	70.42%	70.42%	34.93%
	HR	91.20%	46.73%	70.42%	70.42%	34.93%
	HF1	91.20%	46.73%	70.42%	70.42%	34.93%
	Cons	**100%**	**100%**	**100%**	**100%**	**100%**
	EM	91.20%	46.73%	70.42%	70.42%	34.93%
HD-CNN	HP	92.64%	55.67%	82.50%	87.99%	56.30%
	HR	92.77%	58.78%	82.61%	87.94%	57.38%
	HF1	92.69%	56.91%	82.54%	87.96%	56.73%
	Cons	99.47%	76.08%	99.40%	99.91%	86.09%
	EM	90.25%	42.61%	79.79%	79.10%	47.83%
B-CNN	HP	96.59%	72.69%	89.26%	58.83%	64.41%
	HR	96.97%	77.03%	91.48%	63.03%	73.42%
	HF1	96.75%	74.42%	90.18%	60.86%	67.93%
	Cons	98.26%	80.63%	89.72%	99.58%	56.87%
	EM	92.49%	47.29%	78.99%	9.59%	38.90%
Condition-CNN	HP	96.65%	72.91%	86.56%	55.27%	61.07%
	HR	96.84%	76.46%	88.36%	60.97%	67.18%
	HF1	96.73%	74.34%	87.30%	57.98%	63.45%
	Cons	99.16%	82.66%	91.30%	43.90%	65.01%
	EM	92.50%	49.10%	75.30%	15.09%	39.50%
H-CapsNet	HP	**96.86%**	**76.93%**	**93.90%**	**92.22%**	**75.11%**
	HR	**97.36%**	**80.97%**	**95.79%**	**92.75%**	**82.60%**
	HF1	**97.07%**	**78.54%**	**94.69%**	**92.48%**	**78.02%**
	Cons	97.60%	83.07%	91.49%	90.34%	66.04%
	EM	**92.69%**	**54.85%**	**86.77%**	**80.78%**	**54.79%**

[a] Denotes the DAG version of the CIFAR-10 dataset.

calculated considering the hierarchical relations in the label tree, where the hierarchical F1-score (HF1) is the harmonic mean of the HP and HR. Further, we reported hierarchical consistency and exact match for all the models. Note that, for each instance, hierarchical consistency implies that the prediction for all levels is consistent with those in the ground-truth label tree, whereas the exact match score applies to the predictions per-level without taking into account whether these are all correct across the coarse, medium and fine levels. In Table 5, we present the HP, HR, HF1, consistency and exact match for our H-CapsNet, the CapsNet [8], the HD-CNN [6], the B-CNN [7] and the Condition-CNN [34]. Note that the baseline CapsNet model in [8] is a non-hierarchical classification approach and, hence, these performance metrics become equivalent to their "flat" classification counterparts, and consistency is measured using only fine-level predictions. Also, recall that our model employs the hierarchical structure of the class-set to impose consistency through the loss, which, in turn introduces a structural constraint on the prediction. This results in an improved performance which can be noted in Table 5, where our proposed H-CapsNet model outperforms the alternatives.

5. Conclusions

In this paper, we have presented H-CapsNet, a Capsule Network for hierarchical classification which uses dedicated capsules to predict

specific classes. Compared with traditional capsule network models, our H-CapsNet delivers multiple predictions per instance which account for all those applicable to the hierarchical label-tree under consideration. Making use of a modified hinge-loss that takes into account the number of classes within each hierarchy and the relationship between each other in the label tree, we moderate the contribution of each hierarchical level to the loss. We do this by dynamically adjusting the loss-weights. This improves performance while balancing the contribution of each of the hierarchical levels making use of the classification error. It is also worth noting that our H-CapsNet employs a dynamic routing algorithm [8] between capsules for each hierarchical branch. Nonetheless, our architecture is quite general with respect to the routing scheme used and, therefore, other alternatives elsewhere in the literature may be used instead. We have shown results on widely available datasets and compared these against alternatives elsewhere in the literature. We have also performed an ablation study. Further, throughout our experiments we have presented results using both tree-based and DAG-based hierarchical label structures. Our experiments show that our H-CapsNet delivers a margin of advantage over the alternatives.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Z. Zhang, Y. Xie, F. Xing, M. McGough, L. Yang, Mdnet: A semantically and visually interpretable medical image diagnosis network, in: Computer Vision and Pattern Recognition, 2017, pp. 6428–6436.

[2] K.E. Joyce, S.E. Belliss, S.V. Samsonov, S.J. McNeill, P.J. Glassey, A review of the status of satellite remote sensing and image processing techniques for mapping natural hazards and disasters, Progress Phys. Geogr. 33 (2) (2009) 183–207.

[3] Z. Hussain, M. Zhang, X. Zhang, K. Ye, C. Thomas, Z. Agha, N. Ong, A. Kovashka, Automatic understanding of image and video advertisements, in: Computer Vision and Pattern Recognition, 2017, pp. 1705–1715.

[4] M. Markkula, E. Sormunen, End-user searching challenges indexing practices in the digital newspaper photo archive, Inf. Retr. 1 (4) (2000) 259–285.

[5] M. Chen, A. Zheng, K. Weinberger, Fast image tagging, in: International Conference on Machine Learning, PMLR, 2013, pp. 1274–1282.

[6] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, Y. Yu, HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition, in: International Conference on Computer Vision, 2015, pp. 2740–2748.

[7] X. Zhu, M. Bain, B-CNN: branch convolutional neural network for hierarchical classification, 2017, arXiv preprint arXiv:1709.09890.

[8] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, Adv. Neural Inf. Process. Syst. 30 (2017).

[9] T. Hahn, M. Pyeon, G. Kim, Self-routing capsule networks, in: Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 7658–7667.

[10] B. Chen, X. Huang, L. Xiao, L. Jing, Hyperbolic capsule networks for multi-label classification, in: ACL, 2020, pp. 3115–3124.

[11] Y. Zhao, T. Birdal, H. Deng, F. Tombari, 3D point capsule networks, in: Computer Vision and Pattern Recognition, 2019, pp. 1009–1018.

[12] H. Ren, X. Yu, L. Zou, Y. Zhou, X. Wang, L. Bruzzone, Extended convolutional capsule network with application on sar automatic target recognition, Signal Process. 183 (2021) 108021.

[13] C. Xiang, L. Zhang, Y. Tang, W. Zou, C. Xu, MS-CapsNet: A novel multi-scale capsule network, IEEE Signal Process. Lett. 25 (12) (2018) 1850–1854.

[14] G.E. Hinton, A. Krizhevsky, S.D. Wang, Transforming auto-encoders, in: International Conference on Artificial Neural Networks, 2011, pp. 44–51.

[15] G.E. Hinton, S. Sabour, N. Frosst, Matrix capsules with EM routing, in: International Conference on Learning Representations, 2018.

[16] A. Dempster, N. Laird, D. Rubin, Maximum-likehood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B Methodol. 39 (1977) 1–38.

[17] M.T. Bahadori, Spectral capsule networks, in: International Conference on Learning Representations Workshops, 2018.

[18] J.O. Neill, Siamese capsule networks, 2018, arXiv E-preprints.

[19] Y. Upadhyay, P. Schrater, Generative adversarial network architectures for image synthesis using capsule networks, 2018, arXiv E-preprint.

[20] M. Jampour, S. Abbaasi, M. Javidi, CapsNet regularization and its conjugation with ResNet for signature identification, Pattern Recognit. 120 (2021) 107851.

[21] B. Mandal, R. Sarkhel, S. Ghosh, N. Das, M. Nasipuri, Two-phase dynamic routing for micro and macro-level equivariance in multi-column capsule networks, Pattern Recognit. 109 (2021) 107595.

[22] C.N. Silla, A.A. Freitas, A survey of hierarchical classification across different application domains, Data Min. Knowl. Discov. 22 (2011) 31–72.

[23] J. Rousu, C. Saunders, S. Szedmak, J. Shawe-Taylor, Kernel-based learning of hierarchical multilabel classification models, J. Mach. Learn. Res. 7 (2006) 1601–1626.

[24] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Mach. Learn. 73 (2) (2008) 185–214.

[25] J. Wehrmann, R. Cerri, R. Barros, Hierarchical multi-label classification networks, in: International Conference on Machine Learning, 2018, pp. 5075–5084.

[26] I. Dimitrovski, D. Kocev, S. Loskovska, S. Džeroski, Hierarchical annotation of medical images, Pattern Recognit. 44 (10) (2011) 2436–2449.

[27] S. Ubaru, S. Dash, A. Mazumdar, O. Günlük, Multilabel classification by hierarchical partitioning and data-dependent grouping, in: Advances in Neural Information Processing Systems, 2020.

[28] Y. Meng, J. Shen, C. Zhang, J. Han, Weakly-supervised hierarchical text classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 6826–6833.

[29] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in: Computer Vision and Pattern Recognition, 2017, pp. 7263–7271.

[30] J. Davis, T. Liang, J. Enouen, R. Ilin, Hierarchical classification with confidence using generalized logits, in: 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 1874–1881.

[31] J. Deng, J. Krause, A.C. Berg, L. Fei-Fei, Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3450–3457.

[32] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, Springer, 2014, pp. 818–833.

[33] Y. Seo, K.-s. Shin, Hierarchical convolutional neural networks for fashion image classification, Expert Syst. Appl. 116 (2019) 328–339.

[34] B. Kolisnik, I. Hogan, F. Zulkernine, Condition-CNN: A hierarchical multi-label fashion image classification model, Expert Syst. Appl. 182 (2021) 115195, http://dx.doi.org/10.1016/j.eswa.2021.115195.

[35] A. Dhall, A. Makarova, O. Ganea, D. Pavllo, M. Greeff, A. Krause, Hierarchical image classification using entailment cone embeddings, in: Computer Vision and Pattern Recognition Workshops, 2020, pp. 836–837.

[36] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.

[37] T. Boone-Sifuentes, A. Nazari, I. Razzak, M.R. Bouadjenek, A. Robles-Kelly, D. Ierodiaconou, E.S. Oh, Marine-tree: A large-scale hierarchically annotated dataset for marine organism classification, in: CIKM, vol. 22, Association for Computing Machinery, 2022, pp. 3838–3842.

[38] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.

[39] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, Mixup: Beyond empirical risk minimization, in: International Conference on Learning Representations, 2018, pp. arXiv–1710.

[40] A. Kosmopoulos, I. Partalas, E. Gaussier, G. Paliouras, I. Androutsopoulos, Evaluation measures for hierarchical classification: a unified view and novel approaches, Data Min. Knowl. Discov. 29 (3) (2015) 820–865.

[41] S. Kiritchenko, S. Matwin, R. Nock, A.F. Famili, Learning and evaluation in the presence of class hierarchies: Application to text categorization, in: Conference of the Canadian Society for Computational Studies of Intelligence, Springer, 2006, pp. 395–406.