SUPPLEMENTARY MATERIAL
Making Better Mistakes: Leveraging Class Hierarchies with Deep Networks

Luca Bertinetto* Romain Mueller* Konstantinos Tertikas Sina Samangooei Nicholas A. Lord*
{luca.bertinetto, romain.mueller, konstantinos.tertikas, sina, nick.lord}@five.ai
www.five.ai

A. Outputting conditional probabilities with HXE

We investigated whether outputting conditional probabilities instead of class probabilities affects the performance of the classifier represented by our proposed HXE approach (Sec. 3.1). These two options correspond, respectively, to implementing hierarchical information as an architectural change or as modification of the loss only:

- When the model outputs the conditional probabilities $p(C^{(l)}|C^{(l+1)})$ directly, the output dimension is equal to the total number of nodes in the hierarchy and normalisation is ensured hierarchically with a different softmax at each independent level of the hierarchy. Eqn. 4 can be used directly on the output of the model.
- When the model outputs the class probabilities, the output dimension is equal to the number of leaf nodes in the hierarchy and normalisation can be performed with a single softmax. In this case, however, it is necessary to use Eqn. 3 in order to obtain the conditional probablilities in terms of the final probabilities in Eqn. 4.

The second method, which we advocate here, has the advantage that hierarchichal information is implemented in the loss only, meaning that it does not require direct knowledge of the hierarchy for inference.

Comparing different values of α for otherwise identical training parameters, we also observe that outputting the class probabilities consistently results in an improvement of performance across *all* the metrics, see Suppl. Fig. 2.

B. Note on methods based on hierarchical architectures

We opted not to evaluate against the "generalist/expert" hierarchical models surveyed in Sec. 2.3 for several reasons. For one, none of the listed methods perform experiments under hierarchical measures. As noted in the main text, they all rely on discovered hierarchies, ruling out direct comparison to methods accepting the hierarchies considered here. Crucially, these methods increase the capacity of their base models, which not only rules out controlled experimental comparison, but confounds the intuition behind why their designs demonstrate improved performance: the extent to which the hierarchical design per se is responsible for the gains in top-k accuracy observed is actually an open question. We also note that a recurring theme in these works is the observation that in practice, the use of generalists which make hard categorisations into disjoint coarse categories causes enough irrecoverable errors to motivate moving away from this design strategy. Mitigating approaches typically involve the use of non-disjoint coarse categories and probabilistic one-to-many coarse classification. Thus, while these methods are worthy of mention and further investigation, they do not yet represent attempts at the problem we examine in this paper.

C. More implementation details

In order to perform meaningful comparisons, we adopted a simple configuration (network architecture, optimiser, data augmentation, ...) and used it for all the methods presented in this paper.

We used a ResNet-18 architecture (with weights pretrained on ImageNet) trained with Adam [12] for 200,000 steps and mini-batch size of 256. We used a learning rate of $1e{-}5$ unless specified otherwise. To reduce overfitting, we adopted PyTorch's basic data augmentation routines with default hyperparameters: `RandomHorizontalFlip()` and `RandomResizedCrop()`. For both datasets, images have been resized to $224{\times}224$.

Below, we provide further information about the methods we compared against, together with the few minor implementation choices we had to make. As mentioned in Sec. 1, these methods represent, to the best of our knowledge, the only modern attempts to deliberately reduce the semantic severity of a classifier's mistakes that are gener-

*Equal contribution.

Listing 1: Network head used for DeViSE.

```
model.fc = torch.nn.Sequential(
  torch.nn.Linear(in_features=512,
                  out_features=512),
  torch.nn.ReLU(),
  torch.nn.BatchNorm1d(512),
  torch.nn.Linear(in_features=512,
                  out_features=300)
)
```

ally applicable to any modern architecture.

YOLO-v2. In motivating the hierarchical variant of the YOLO-v2 framework, Redmon & Farhadi [13, Sec. 4], mention the need of integrating the smaller COCO detection dataset [7] with the larger ImageNet classification dataset under a unified class hierarchy. Their approach too relies on a heuristic for converting the WordNet graph into a tree, and then effectively training a conditional classifier at every parent node in the tree by using one softmax layer per sibling group and training under the usual softmax loss over leaf posteriors. The authors report only a marginal drop in standard classification accuracy when enforcing this tree-structured prediction, including the additional internal-node concepts. They note that the approach brings benefits, including graceful degradation on new or unknown object categories, as the network is still capable of high confidence in a parent class when unsure as to which of its children is correct.

Since the model outputs conditional probabilities instead of class probabilities, we changed the output dimension of the terminal fully-connected layer, such that it outputs logits for every node in the hierarchy. Proper normalisation of the conditional probabilities is then enforced at every node of the hierarchy using the softmax function. Finally, the loss is computed by summing the individual cross-entropies of the conditional probabilities on the path connecting the ground-truth label to the root of the tree.

DeViSE. Frome *et al*. [4] proposed DeViSE with the aim of both making more semantically reasonable errors and enabling zero-shot prediction. The approach involves modifying a standard deep classification network to instead output vectors representing semantic embeddings of the class labels. The label embeddings are learned through analysis of unannotated text [9] in a separate step, with the classification network modified by replacing the softmax layer with a learned linear mapping to that embedding space. The loss function is a form of ranking loss which penalises the extent of greater cosine similarity to negative examples than positive ones. Inference comprises finding the nearest class embedding vectors to the output vector, again under cosine similarity.

Since an official implementation of DeViSE is not available to the public, we re-implemented it following the details discussed in the paper [4]. Below the list of changes we found appropriate to make.

- For the generation of the word embeddings, instead of the rather dated method of Mikolov *et al*. [9], we used the high-performing and publicly available[1] fastText library [3] to obtain word embeddings of length 300 (the maximum made available by the library).
- Instead of a single fully-connected layer mapping the network output to the word embeddings, we used the network "head" described in Listing 1. We empirically verified that this configuration with two fully-connected layers outperforms the one with a single fully-connected layer. Moreover, in this way the number of parameters of DeViSE roughly matches the one of the other experiments, which have architectures with a single fully-connected layer but a higher number of outputs (608, equivalent to the number of classes of *tieredImageNet-H*, as opposed to 300, the word-embedding size).
- Following what described in [4], we performed training in two steps. First, we trained only the fully-connected layers for the first $150{,}000$ steps with a learning rate of $1\mathrm{e}{-4}$. We then trained the entire network for $150{,}000$ exta epochs, using a learning rate of $1\mathrm{e}{-6}$ for the weights of the backbone. Note that [4] did not specify neither how long the two steps of training should last nor the values of the respective learning rates. To decide the above values, we performed a small hyperparameter search.
- [4] says that DeViSE is trained starting from an ImageNet-pretrained architecture. Since we evaluated all methods on *tieredImageNet-H*, we instead initialised DeViSE weights with the ones of an architecture fully trained with the cross-entropy loss on this dataset. We verified that this obtains better results than starting training from ImageNet weights.

Barz&Denzler [2]. This approach involves first mapping class labels into a space in which dot products represent semantic similarity (based on normalised LCA height), then training a deep network to learn matching feature vectors (*before* the fully connected layer) on its inputs. There is a very close relationship to DeViSE [4], with the main difference being that here, the label embedding is derived from a supplied class hierarchy in a straightfoward manner instead of via text analysis: iterative arrangement of embedding vectors such that all dot products equal respective semantic similarities. The authors experiment with two different loss functions: (1) a linear reward for the dot product between the output feature vector and ground-truth class embedding (*i.e*. a penalty on misalignment); and (2) the sum

[1] https://github.com/facebookresearch/fastText

of the preceding and a weighted term of the usual cross-entropy loss on the output of an additional fully connected layer with softmax. We only used (2), since in [2] it attains significantly better results than (1).

We used the code released by the authors [2] to produce the label embeddings. To ensure consistency with the other experiments, two differences in implementation with respect to the original paper were required.

- We simply used a ResNet-18 instead of the architectures Barz & Denzler experimented with in their paper [2] (*i.e.* ResNet-110w [6], PyramidNet-272-200 [5] and Plain-11 [1]).
- Instead of SGD with warm restarts [8], we used Adam [12] with a learning rate of $1e{-}4$ (the value performing best on the validation set).

D. Pruning the WordNet hierarchy

The ImageNet dataset [14] was generated by populating the WordNet [10] hierarchy of nouns with images. WordNet is structured as a graph composed of a set of IS-A parent-child relationships. Similarly to the work of Morin & Bengio [11] and Redmon & Farhadi [13], our proposed hierarchical cross entropy loss (HXE, Sec. 3.1) also relies on the assumption that the hierarchy underpinning the data takes the form of a tree. Therefore, we modified the hierarchy to obtain a tree from the WordNet graph.

First, for each class, we found the longest path from the corresponding node to the root. This amounts to selecting the paths with the highest discriminative power with respect to the image classes. When multiple such paths existed, we selected the one with the minimum number of new nodes and added it to the new hierarchy. Second, we removed the few non-leaf nodes with a single child, as they do not possess any discriminative power.

Finally, we observed that the pruned hierarchy's root is not PHYSICAL ENTITY, as one would expect, but rather the more general ENTITY. This is problematic, since ENTITY contains both physical objects *and* abstract concepts, while *tieredImageNet-H* classes only represent physical objects. Upon inspection, we found that this was caused by the classes BUBBLE, TRAFFIC SIGN, and TRAFFIC LIGHTS being connected to SPHERE and SIGN, which are considered abstract concepts in the WordNet hierarchy. Instead, we connected them to SPHERE, ARTIFACT and SIGNBOARD, respectively, thus connecting them to PHYSICAL ENTITY.

Even though our second proposed method (*soft labels*), as well as the cross-entropy baseline, DeViSE [4] and Barz & Denzler [2], do not make any assumption regarding the structure of the hierarchy, we ran them using this obtained pruned hierarchy for consistency of the experimental setup.

[2] https://github.com/cvjena/semantic-embeddings

References

[1] Björn Barz and Joachim Denzler. Deep learning is not a matter of depth but of good training. In *International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI)*, 2018.

[2] Björn Barz and Joachim Denzler. Hierarchy-based image embeddings for semantic image retrieval. In *IEEE Winter Conference on Applications of Computer Vision*, 2019.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 2017.

[4] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, 2013.

[5] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

[8] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2016.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[10] George A Miller. *WordNet: An electronic lexical database*. 1998.

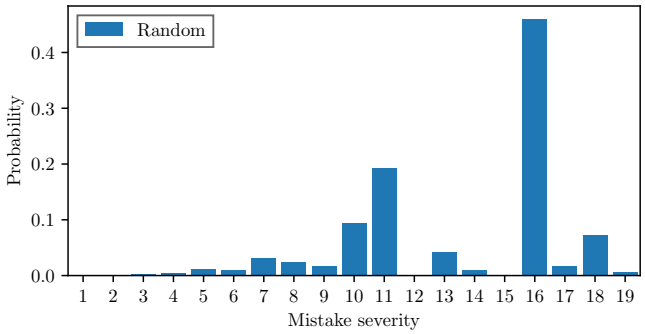
[11] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*. Citeseer, 2005.

[12] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2019.

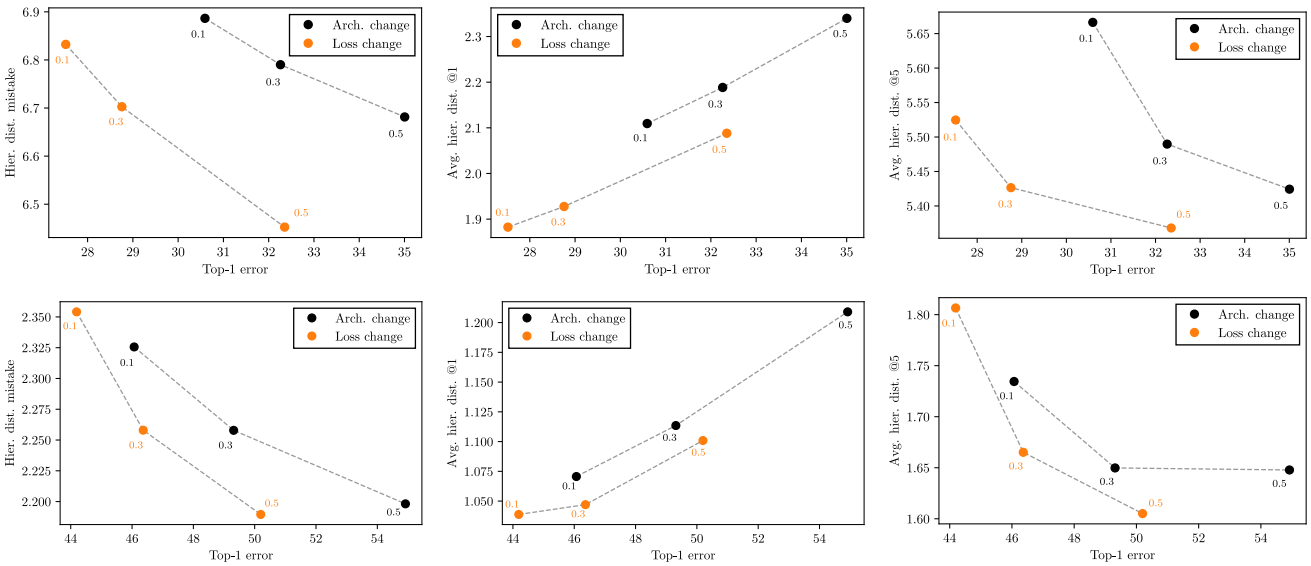
[13] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

E. Supplementary figures



Supplementary Figure 1: Distribution of mistake severity when picking random example pairs in ImageNet/ILSVRC-12. Note that though this distribution shares some qualitative similarities with the ones shown in Fig. 1, it is nonetheless substantially different. This demonstrates that the shapes of the mistake-severity distributions for the various DNN architectures studied cannot be explained by properties of the dataset alone.



Supplementary Figure 2: Outputting the conditional probabilities (architectural change) results in a degradation of performance compared to outputting the class probabilities directly (loss change) when using the hierarchical cross-entropy loss with exponential weights $\lambda(C) = \exp(-\alpha h(C))$. Results are shown both on *tieredImageNet-H* (top) and *iNaturalist19-H* (bottom). Points closer to the bottom-left corner of the plots are the ones achieving the best tradeoff.