CentraleSupélec

Report

Filière Métiers de la recherche

Research Project

Loss in Layers: Evaluating hierarchical loss functions for image classification (gr. 28)

Students:
Pierrick Bournez
Paul-Arno Lamarque
Kerrian Le Caillec
Samuel Sithakoul
Paul Tabbara

Supervisor :
Alix Chazottes

October 4, 2024

Abstract

Keywords— Neural Network - Hierarchical classification - Image recognition

1 Introduction

Supervised classification has become one of the major tasks of interest in machine learning for different applications or data. Classification can be applied to structured data and unstructured data like text [1] [2] or images [3]. In many real-life cases, target classes exhibit an explicit label structure where it is possible to group specific classes into semantic groups. However, the majority of models operate under the assumption that all target classes are mutually exclusive and can be predicted independently; this is the usual so-called "flat" classification. These methods fail to account for the hierarchical relationships between classes and assign equal weight to different predictions errors. In contrast, human reasoning often follows a dichotomous approach: some may first recognize a cat as an animal and then assimilate it as a cat while a truck is immediately separated in another group like automobiles.

Using such target structure to train a model is referred to as hierarchical classification, which is not to be confused with other research domains inducing hierarchy at other levels of the pipeline. The objective is to constraint the training procedure to respect the predefined class hierarchy. To explicit such class hierarchies, several authors have proposed tailored models for the task, introduced post-training regularization, or developed custom loss functions. In this study, we focus on loss function design. This design is often both model and task-agnostic, making it applicable to other tasks without incurring additional computational overhead.

We found during our study that there is a significant lack of research on producing benchmarks for such hierarchical losses. Study on the influence of the target structure chosen or loss designs would produce insightful comparison with existing approaches and define a clear scope for the usage of hierarchical classification.

To address the aforementioned challenges, we introduce a library to evaluate hierarchical losses and help research in this field.

Our main contributions are:

- An implementation of a tensor-optimized pipeline to facilitate experimentation with various loss parameters,
- A comparative analysis of our hierarchical loss against the conventional cross-entropy loss and a regularized version. We will compare it with different tree structures
- We provide a hyperparameter search for the hierarchical loss, enabling its fine-tuning for future experiments.

2 Related works

A number of methods to improve the performance of image classification algorithms have been explored in the research field, among them the use of tree [4] or at least graph-based structures [5] for the labels. Among the analyzed approaches, some papers design deep learning models for hierarchical classification, other papers design loss-specific functions to introduce hierarchical structure in the training. Adding regularization constraints to the model parameters has also been studied to improve accuracy [1].

A vast majority of the proposed methods, in both classification and segmentation tasks, revolves around architectures specifically conceived to answer hierarchical tasks. Those architectures are either a "network of experts" [6] or human parsing models [7]. Network of experts consists of an initial model that will predict the subset of classes and an expert model will then predict a class

under the predicted subset. Such models require more training steps as we increase the number of predicted labels and cannot recover an error from the initial model. Human parsing models aim to construct a sparse graph that explains observations: a football image is parsed into person which is further decomposed, e.g face or body patterns [8]. Such approach uses foundational models and tailors very-specific models for the task [9].

To further improve classification results, some have proposed regularizing prediction outputs. Such approach adds constraint during or after the training phase. In [5], Giunchiglia and Lukasiewicz add a constraint layer inside the model to ensure prediction coherence. After the training phase, [1] regularizes classes prediction based on ancestors predictions. It allows mitigating class imbalance or classes prediction among the same part of the tree.

Some authors design specific loss to ensure the hierarchical prediction of a model. In [10] [11], they construct a loss that leverage local class-relationship and penalizes over hierarchy violations. Such designed approach led to rethink about how to evaluate models [12]. Predictions that are far from another in the tree should be more penalized [13]. It improves in the same time performance and explainability [2]. Eventually, to relax the hierarchical classification, some authors create an implicit label hierarchy by embedding every class into a vector space. Similar semantic classes are being closer in the vector space, leading to an improvement in accuracy over novel classes [14] [15]. These embeddings facilitate effective training, and the model can be generalized when queries are in open vocabulary [16].

3 Mathematical theory

This section aims to define the hierarchical structure used for the data labelling, as well as the quantities used in the rest of the paper.

3.1 Definition of the loss functions

In this paper, our important loss is the hierarchical loss defined in Equation 2. We will compare it with the well established cross-entropy, defined in Equation 5. We will propose two minors modifications: an improved cross-entropy defined in Equation 6, and a hierarchical convex counterpart defined in Equation 4.

Let us consider a set of images $\mathcal{I}$. We denote the ground-truth label of an image $I \in \mathcal{I}$ as $\boldsymbol{y} = (y_i)_{i=1}^N \in \{0,1\}^N$. $\boldsymbol{y}$ is a column vector with y_i being equal to 1 if the image belongs to the i-th class of the model. The aim of the project is to reconstruct $\boldsymbol{y}$ using the inferred labels obtained by a deep model $\boldsymbol{z} = (z_i)_{i=1}^N \in [0,1]^N$.

We define the loss obtained by obtaining the Bregman divergence on $u \mapsto \sum_{i=1}^N u_i \log(u_i)$, we then obtain the modified Kullback-Leibler divergence between hierarchical ground-truth labels $\boldsymbol{y}$ and the inferred labels $\boldsymbol{z} = (z_i)_{i=1}^N \in [0,1]^N$, both of size N:

$$\ell_H(\boldsymbol{y}, \boldsymbol{z}) = -\sum_{i=1}^N y_i \log z_i + \sum_{i=1}^N z_i. \tag{1}$$

Note that the function ℓ_H is convex in z.

To truly define the hierarchical classification, we need to introduce the concept of Markov trees. Let us consider a spanning tree $\mathcal{T} = (V, E)$, where V is the set of nodes that represent our classes. For a node $j \in V$, we denote $a_j \in V$ its ancestor, the node j is called a subclass of a_j. The root of the tree is the only class which is not the subclass of another node. E is the set of edges in the tree,

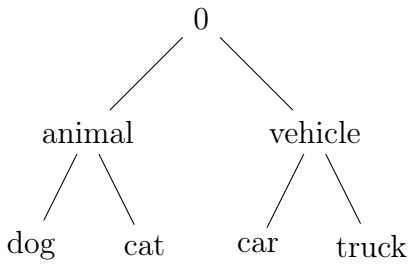


Figure 1: Markov tree example for hierarchical labelling problem, "car" is a subclass of "vehicle"

on which a probability operator $\mathbb{P}$ is defined. Let us define the inferred probability of $j \in V$ given its ancestor a_j: $p_{j,a_j} := \mathbb{P}(j|a_j)$. The probability of landing on a node j knowing the root is thus given by a Markov chain. An example of such tree is given Figure 1.

By denoting $\mathbb{J}_i \subset \{1, \dots, N\}$ the path from the root to a node i, we can write $z_i = \prod_{j \in \mathbb{J}_i} p_{j,a_j}$ because z_i is a Markov chain. We call the depth of a node its number of ancestors plus one. We also define the matrix $J = (\mathbb{1}_{j \in \mathbb{J}_i})_{1 \leq i,j \leq N}$ whose i-th row is the binary path to the node i.

Let us consider $s_j := \sum_{\substack{i=1 \\ j \in \mathbb{J}_i}}^{N} y_i$ the sum of all labels of j's descendants, we can rewrite ℓ_H as [17]:

$$\tilde{\ell}_H(\boldsymbol{y}, \boldsymbol{z}) = -\sum_{j=1}^{N} s_j \log p_{j,a_j} + \sum_{\substack{i=1 \\ i \text{ not a leaf}}}^{N} \prod_{j \in \mathbb{J}_i} p_{j,a_j}. \tag{2}$$

This latter formulation is what we used in the experimentation below to calculate the loss. We will refer to this function as the *hierarchical loss* in the following paragraphs. We can obtain the p_{j,a_j} by applying a softmax regularization to each set of nodes children:

$$p_{j,a_j} = \frac{e^{u_{j,i}}}{\sum_k e^{u_{k,i}} \mathbb{1}_{i=a_k}}.$$

We can express $\tilde{\ell}_H$ as a sum of matrix products, if we consider the column vectors $\boldsymbol{p} = (p_{j,a_j})_{j=1}^{N}$ and $\boldsymbol{m} = (\mathbb{1}_{i \text{ not a leaf}})_{i=1}^{N}$, and given that $\boldsymbol{s} := (s_j)_{j=1}^{N} = J^T \boldsymbol{y}$, we find that:

$$\tilde{\ell}_H(y, z) = -\boldsymbol{y}^T J \log(\boldsymbol{p}) + \boldsymbol{m}^T \exp(J \log(\boldsymbol{p})). \tag{3}$$

While the softmax function is surjective to the $(n-1)-$simplex on which the vector $\boldsymbol{p}$ is embedded, $\boldsymbol{p} \mapsto m^T \exp(\mathbb{J} \log(\boldsymbol{p}))$ is actually a non-convex function on the $(n-1)-$simplex. Indeed, the construction of z_i as a Markov chain has added a non-convex constraint to the problem, thus the optimization problem associated with $\tilde{\ell}_H$ is not convex.
To achieve convexity in the classification problem, another loss function is defined by Pesquet [17]. By noting that $y_i = \prod_{j \in \mathbb{J}_i} q_{j,a_j}$, where $q_{j,a_j} = \mathbb{1}_{j \in \mathbb{J}_i}$, for all i, we can then define the *convex hierarchical loss* using the Kullback-Leibler divergence $D_{\mathrm{KL}}(\cdot \| \cdot)$:

$$\ell_{Hc}(y, z) = -\sum_{\substack{i=1 \\ i \text{ not a leaf}}}^{N} \frac{D_{\mathrm{KL}}(\boldsymbol{q_i} \| \boldsymbol{p_i})}{\mathrm{depth}(i)}, \tag{4}$$

with $\boldsymbol{p_i} = (p_{j,i})_{j\in\mathbb{D}_i}$, $\boldsymbol{q_i} = (q_{j,i})_{j\in\mathbb{D}_i}$, and $\mathbb{D}_i$ the set of direct children of i. The loss is indeed convex in the probability output thanks to the convexity of the Kullback-Leibler divergence.

To assert the goodness of the results given by the hierarchical loss or the convex hierarchical loss, we use well-known models to compare the results. Thus, considering the cross-entropy of the neural model output softmax probabilities and the ground-truth labels:

$$\ell_{CE}(\boldsymbol{y}, \boldsymbol{u}) = -\sum_{i=1}^{N} y_i \log \circ \operatorname{softmax}(u_i). \tag{5}$$

Using this quantity is not enough to assert whether the hierarchical structure has an edge over traditional loss functions. One simple improvement over the cross-entropy is adding a regularization term to the loss. Pereyra et al. [18] proposes the following loss:

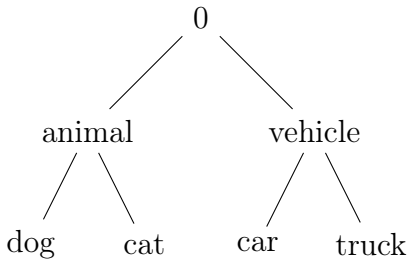
$$\ell_{rCE}(\boldsymbol{y}, \boldsymbol{u}) = \ell_{CE}(\boldsymbol{y}, \boldsymbol{u}) + \lambda \sum_{i=1}^{N} \operatorname{softmax}(u_i) \log(\operatorname{softmax}(u_i)), \tag{6}$$

where λ is a regularization constant. The right sum is the neg-entropy, it entails that the more confident the model is about a prediction, the more it is penalized. This is done to avoid overfitting over specific classes and re-balance the predictions.

3.2 Reformulation for parallel computing

For the algorithm implementation, it is essential to select an appropriate representation of the tree structure defined in the previous section, as this choice will significantly influence both the computational efficiency and the ease of interpretability.

As we will be using neural networks as primary models, we choose to represent the hierarchical trees with adjacency matrices. This structure is well-suited for parallel computing but it is necessary to make adjustments in the equations for an accurate implementation. Our choice is equivalent to the original formulation and allow us to compute all steps directly in batches for GPU computation. The implementation is open-source in a repository.



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 2: Adjacency matrix of a Hierarchical label tree. Each row corresponds to all children of a given parent, we leave the first column with only 0 for the root node. All computations of probabilities, paths and losses can be derived from this representation.

For instance, the tree represented in Figure 1 is implemented with the adjacency matrix in Figure 2.

4 Experimental setup

4.1 Dataset

We use the CIFAR-10 (Canadian Institute for Advanced Research, 10 classes) dataset which is a subset of the *Tiny Images* dataset with 60000 color images of size 32×32 in the experiments. The

images are divided into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 6000 images per class with 5000 for training and 1000 for testing per class so 50000 total images for training.

For our experiments, we use 80% of the training set provided by the CIFAR-10 API to train the CNN model and 20% for validation after each epoch. For final evaluation, we use the whole test set provided by CIFAR-10. We only modify hyperparameters between different runs after considering the performance on the validation set to avoid unnecessary biases in reported results.

4.2 Metrics

In order to evaluate the performances of our model, metrics have to be chosen. We will primarily focus on the accuracy as well as the F1 score and the recall metrics. Furthermore, the idea of incorporating the hierarchical structure in the metrics has been studied as well.

Semantic distance

Inspired by [19], we construct metrics based on the semantic distance. The semantic distance between two leaves $i, j \in leaves(\mathcal{T})$ is the difference between the deepest node layer and the layer of the first common parent, or mathematically $d(i,j) = \max(\text{depth}(i), \text{depth}(j)) - \max_{x \in \mathbb{J}_i \cap \mathbb{J}_j} (\text{depth}(x))$ [20]. The more distant a prediction is, the more penalized it should be. We denote $D = d(i,j)_{i,j \in leaves(\mathcal{T})}$, the distance matrix, and C the confusion matrix. We define wFP_i (resp. wFN_i), the weighted false positives (resp. weighted false negative), where each mislabel is multiplied by its distance with node i, as follows:

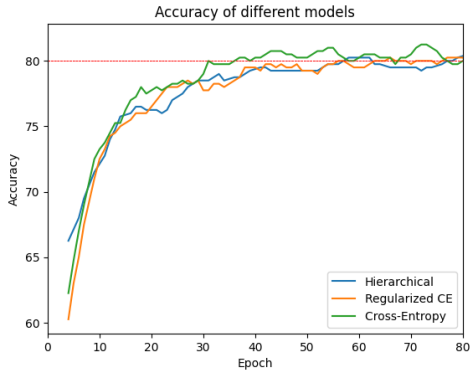
$$\boldsymbol{wFP} = diag(D(C - diag(C))^T),$$
$$\boldsymbol{wFN} = diag(D(C - diag(C))).$$

We rebuild the aforementioned metrics using the two previous quantities to obtain weighted metrics.

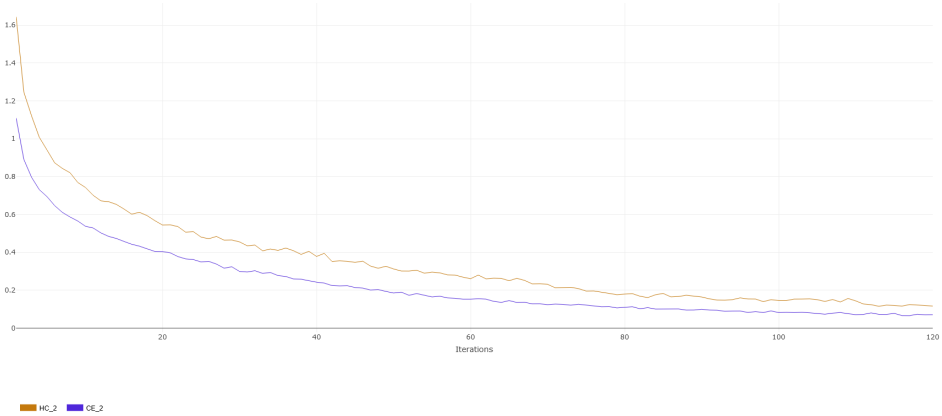
5 Experiments

5.1 Loss comparison

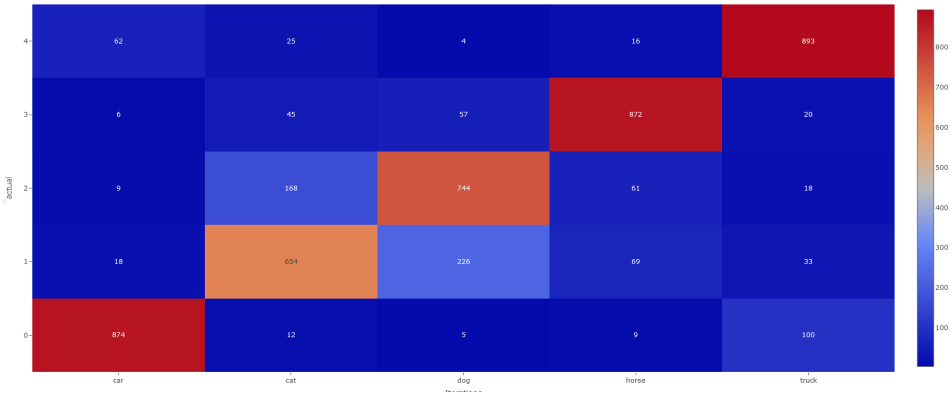
We compare the different losses and accuracies in Figure 3a. After convergence, both models all have the same accuracy and almost the same loss. In Figure 3c, most errors are between 100 cars mislabelled as truck and 168 cat mislabelled as dogs. As the original hierarchical classes is defined in Figure 2, cars and trucks, dogs and cats have the same parent, parent conditional probability cannot help to distinguish between those classes and only the last probability helps to distinguish the results.



(a) Total accuracy for the different loss functions. The different loss functions yield similar asymptotic results



(b) Profile of training loss for the hierarchical convex loss (brown) and ResNet cross-entropy loss (purple)



(c) Confusion Matrix for the hierarchical loss using LeNet. Counting the number of mislabelled data, one can see that the mislabelled clusters are based on the proximity of labels like for cats and dogs and less noticeably for truck and cars.

Figure 3: Results for the different loss functions. We compare the developed hierarchical loss with traditional cross-entropy and a regularized version with different representations.

5.2 Model comparisons

For the model specifications, several changes can be made to better tune our model. We want to compare the different modifications with a reference model. The aim is to check the validity and efficiency of the proposed model compared to other available options. We choose a neural network based on LeNet architecture as a reference [21].

To further investigate the effect of the use of a hierarchical loss, we also compared models with a higher number of parameters, the idea being that it could have been that a model too small may not be able to model the hierarchy in the data. Hence we implemented a CNN that uses 6 convolutional layers and we also implemented a small ResNet.
The base model as well as the other studied parameters are described in the Table 1.

Parameter name/model	LeNet	Resnet
Convolutional layers	3 (16,32,64)	6 (32,64,128,128,256,256)
max pooling	×	×
activation function	ReLU	ReLU
dense layers	2	3
scheduler	Constant Learning rate	Step Learning rate
Optimizer	Adam	SGD

Table 1: Parameter table. The reference column is the reference model while the other ones are the possible modifications.

We conducted experiments on both architectures with the classical Cross-Entropy and the Hierarchical Cross-Entropy to measure differences either between the same model (i.e. a loss effect) or between different models; results can be found in figure 4. Overall we find that after convergence, all models have similar performances. We can notice slight changes at the early stages of the training, but after 40 iterations all models performs equivalently.
As the number of parameter of our models increase, we would have expected that performance also increases. However, this was not what we observed in practice, suggesting that the CIFAR-10 dataset might be too simple for the objectives of our study.

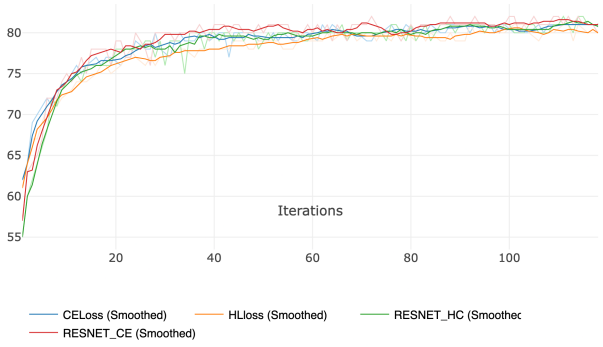


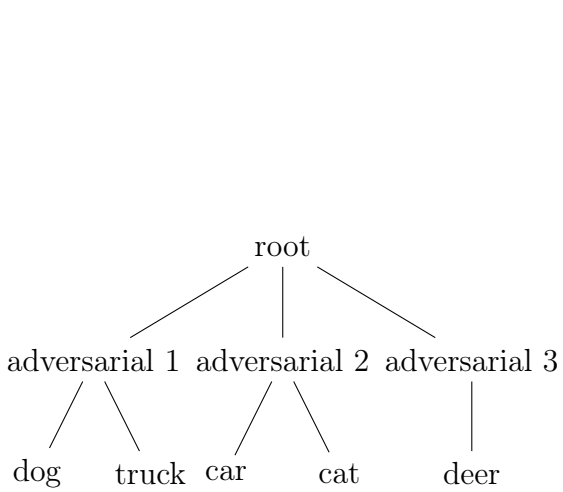
Figure 4: Total accuracy over test dataset for a training with a CrossEntropy loss for LeNet (blue) and ResNet (red), and with a Hierarchical CrossEntropy loss for LeNet (orange) and ResNet (green). – running average for 1 step. Though ResNet seems to perform a bit better, all curves are very similar.

5.3 Tree structure

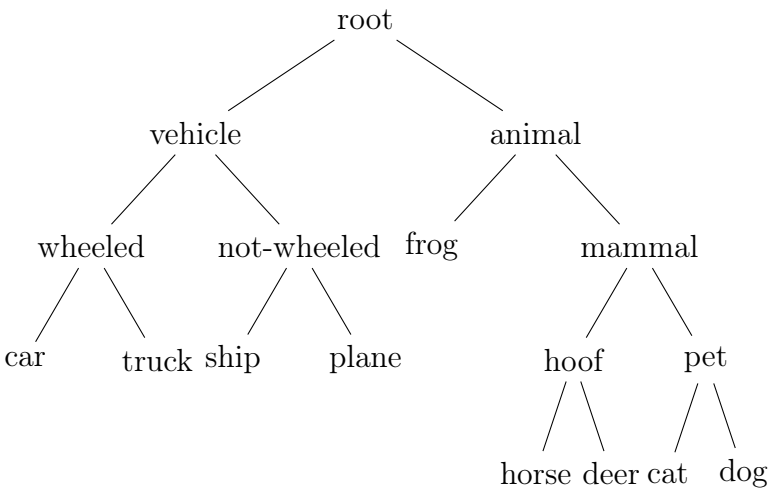
We also conducted experiments to measure the potential influence of the tree structure on models performances:

- We built an adversarial tree to measure how much the tree structure will influence the prediction output (Figure 5a),

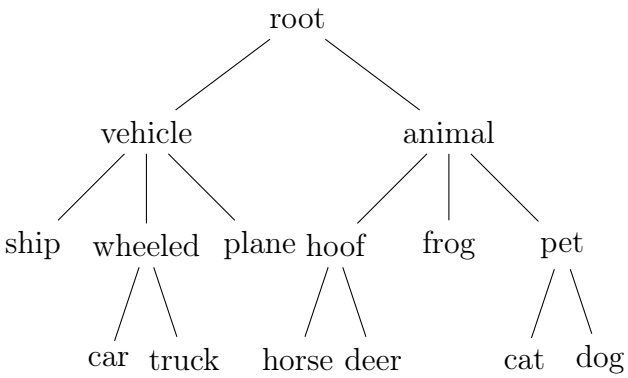
- We compared the use of a binary tree (Figure 5b) with a wider tree (Figure 5c) for modeling the data hierarchy.



(a) Adversarial tree



(b) Binary tree

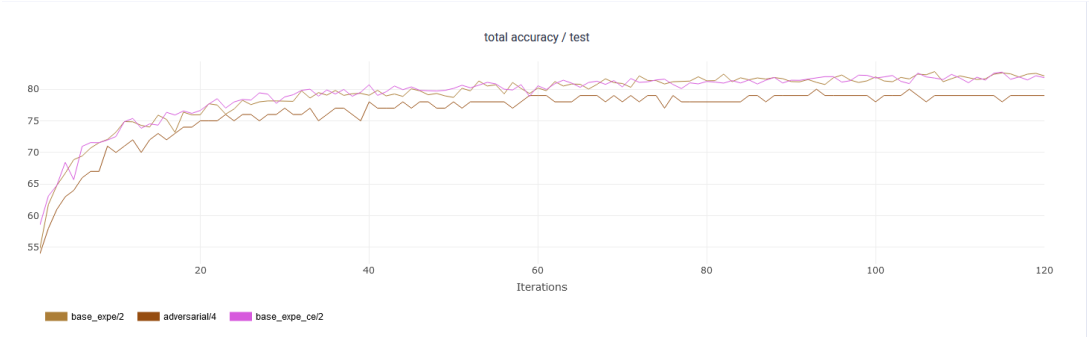


(c) Wider tree

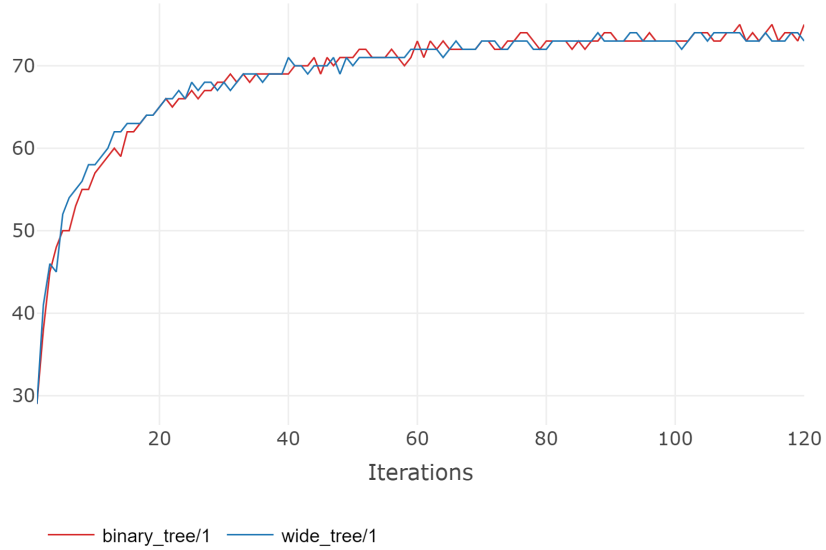
Figure 5: Trees used in the experiments. We compare structures with different depths or number of children to report the influence of tree design in the evaluation results.

The resulting experiments are shown in Figure 6. The confusion matrix comparing the adversarial tree and the conventional tree reveals minimal differences. In fact, according to internal tests, the probabilities at the final leaf of the adversarial trees are almost always either 0 or 1. The model primarily differentiates between classes at higher levels of the tree, where probabilities are more distributed.

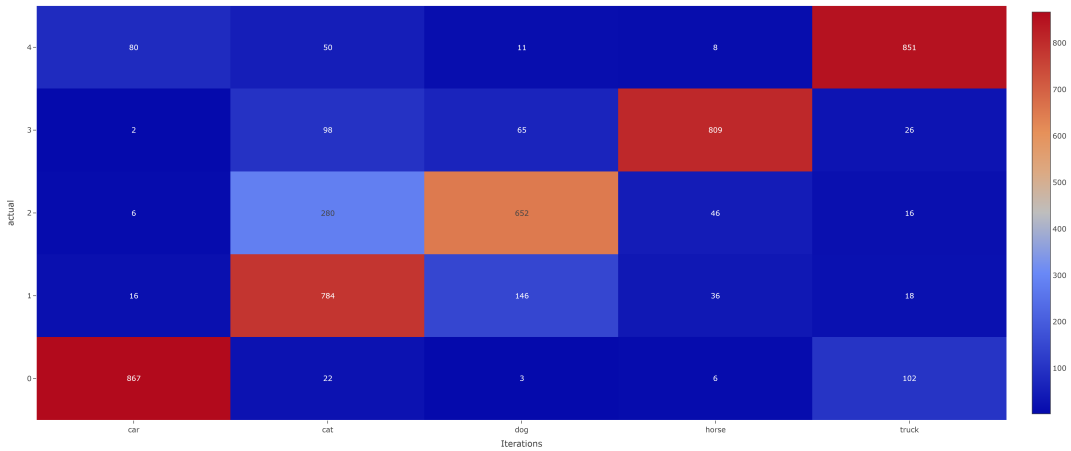
Furthermore, as shown in Figure 6b, we observed that, for our dataset, utilizing a wide tree instead of a binary tree does not result in a significant impact on accuracy. However, we hypothesize that our dataset may be too small to expose differences in performance arising from the tree structure.



(a) Accuracy on the test set for the adversarial tree. The results are slightly lower than those for the default tree with hierarchical or cross-entropy losses.



(b) Training test accuracy for the binary and wide trees. For our dataset, both cases follow the same trend.



(c) Confusion matrix for the adversarial tree. We observe that prediction results are close from the ground-truth and the slight accuracy difference is not significant.

Figure 6: Different results for the tree structures. We compare the adversarial, binary and wide tree with the based tree introduced for our experiments.

5.4 Soft Labels v.s. Binary Labels

After analyzing the results on different trees, we noticed some potential issues regarding the regularization of the network outputs. Let us suppose that we are trying to predict a cat label and the output of the networks is given by Figure 7. In this case, the probability of having a truck is $0.95 \times 0.4 = 0.38$ against $0.6 \times 0.4 = 0.24$. Because of very high probabilities on negative classes, it leads to changes in predictions.

To answer this issue, we decided to replace the binary encoded labels (either 0 or 1 for each class) by soft labels. For children with parents that don't have a label set to 1, we set their label to $1/\#\{\text{Parent's children}\}$, which could make the model learn to standardize outputs on negative labels.

The results shown in Figure 8 indicate that there is no significant difference between training with soft labels and binary labels. While the accuracy plot suggests a slightly faster initial learning with soft labels, this effect is not very clear, and both models eventually converge to the same accuracy.

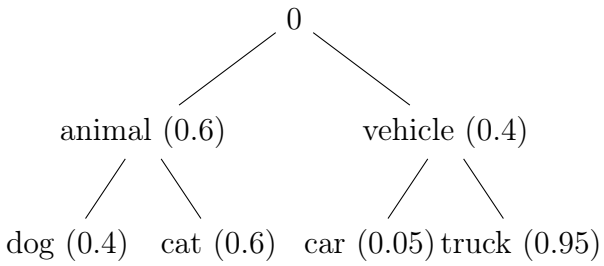


Figure 7: Example of a case where the label cat is replaced by truck due to softmax regularization on the negative side of the tree.

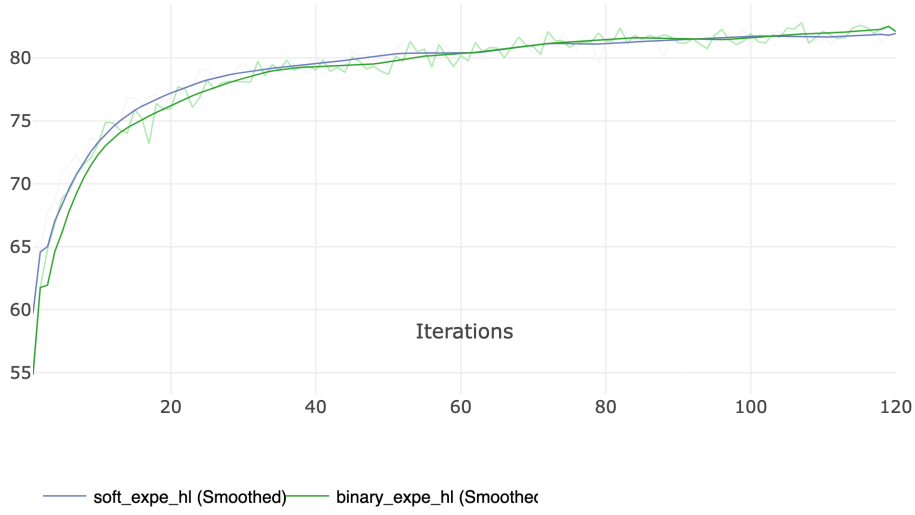


Figure 8: Total accuracy on test dataset for LeNet trained with soft labels (blue) and binary labels (green) – running average for 7 steps. We observe similar accuracy behaviours.

5.5 Hyperparameter search

5.5.1 Optimal learning rate

The accuracy for different learning rates is given in Figure 9. Globally, 10^{-3} seems to be the most optimal value for the learning process of the model.

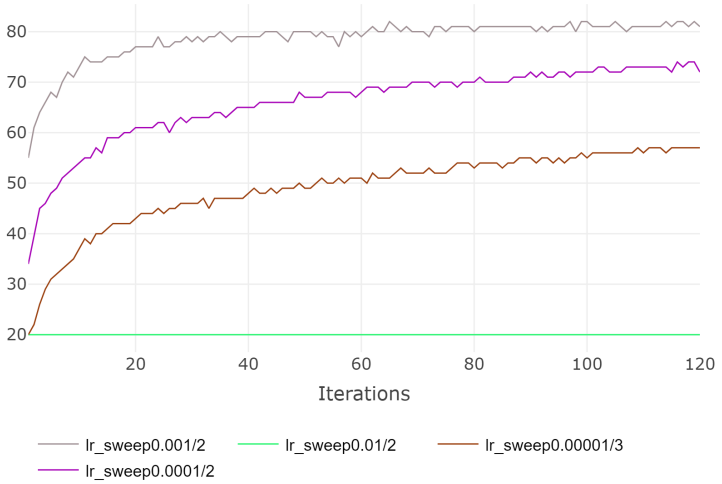


Figure 9: Total accuracy for different values of the learning rate. Lower learning rates lead to slower convergence and too high values lead to failed training, we observe that 10^{-3} is a good compromise between fast convergence and stability.

5.5.2 Learning rate optimizer and scheduler

The influence of the optimizer and learning rate scheduler have also been studied.

First, we chose to compare the Adam optimizer (Adaptative Moment Estimation) to the SGD optimizer (Stochastic Gradient Descent). Figure 10 presents the loss curves for both experiments, conducted using LeNet on the CIFAR-10 dataset with CrossEntropy loss. The results indicate a clear preference for the Adam optimizer in our experiments.

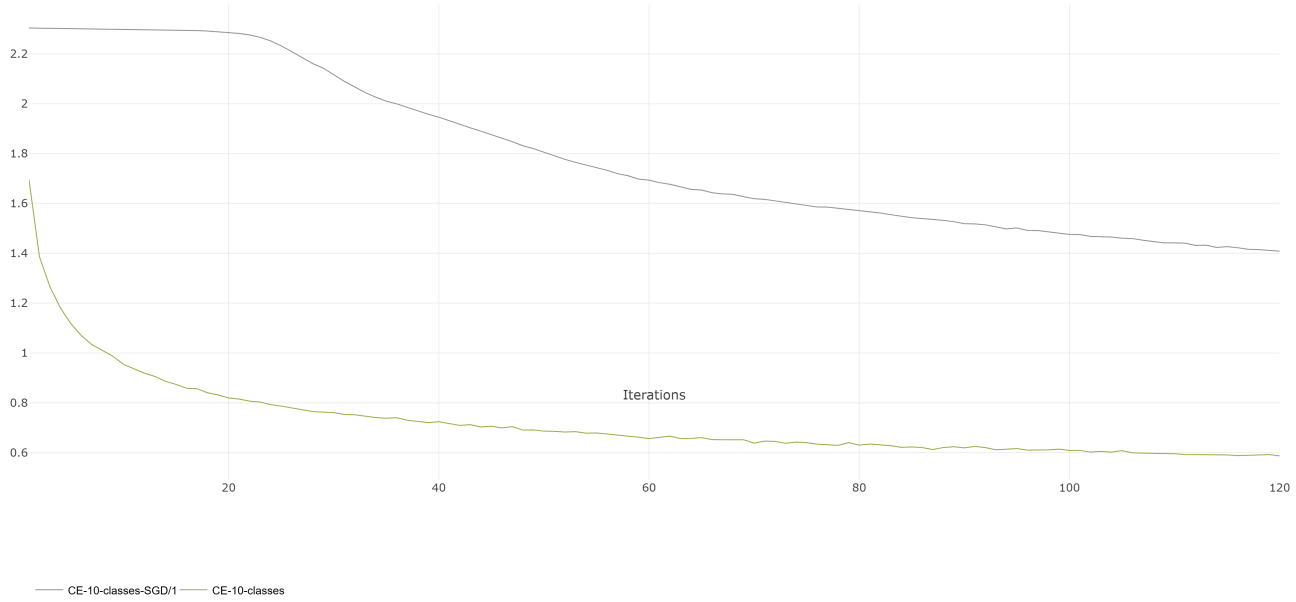


Figure 10: Loss during training, using different optimization algorithms. Using Adam optimizer (green) greatly accelerates loss minimization compared to SGD (gray).

Once the optimizer algorithm is chosen, we can study the impact of adding a scheduling algorithm to change the learning rate during training, according to a chosen pattern. We chose to study a step scheduler, that multiplies the current learning rate by a constant at every given number of steps. After experimenting, the best scheduler we found multiplied the learning rate by 0.8 every 5 epochs. The results shown in Figure 11 reveal that using a step scheduler improves loss minimization.

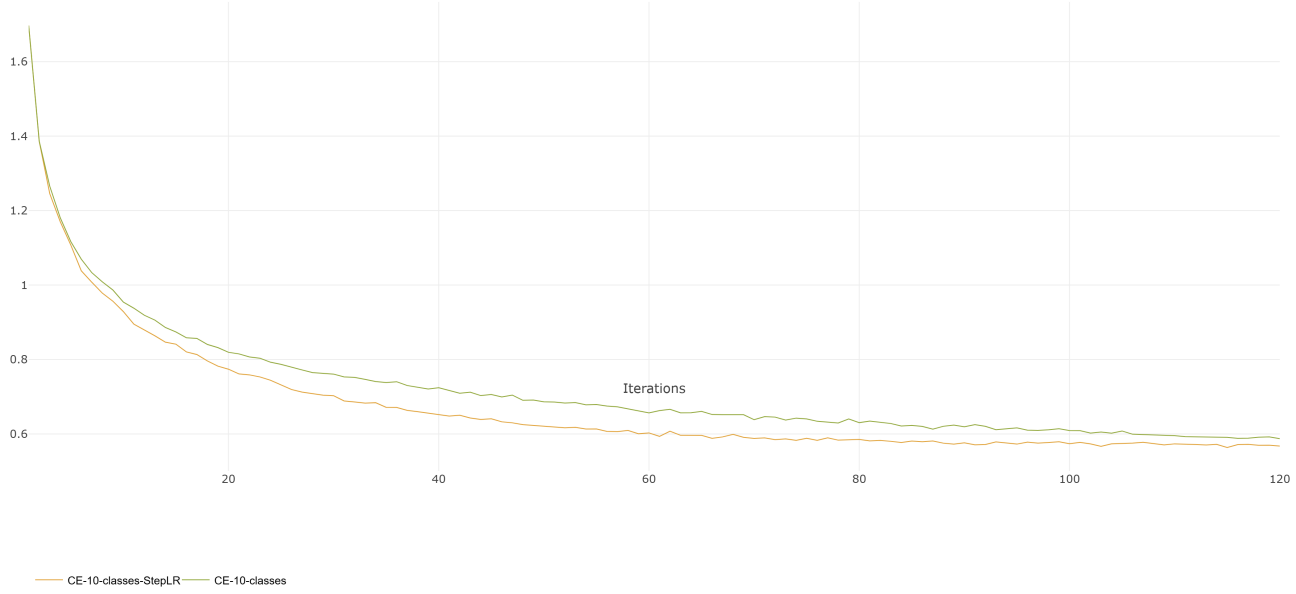
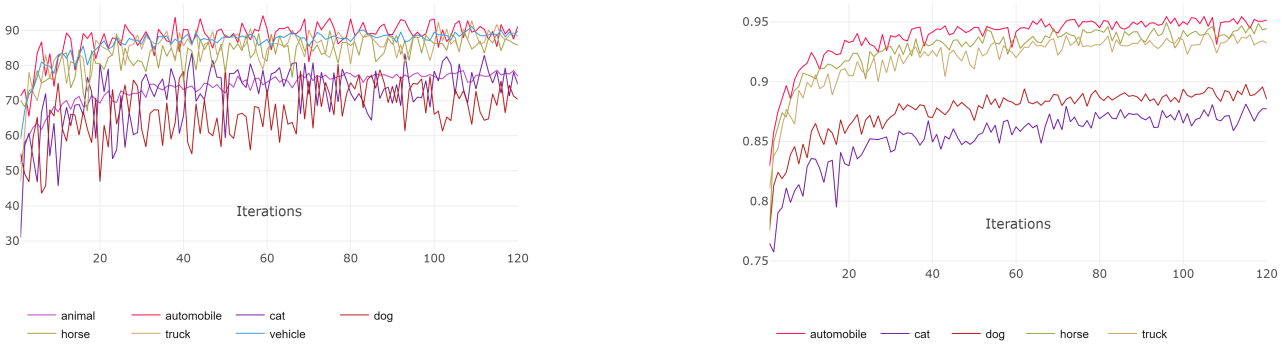


Figure 11: Loss during training, without learning rate scheduling (green) and with a step learning rate scheduler (orange). The implemented step learning rate performs better in our experiments.

5.5.3 Change of metrics

Rather than relying on base accuracy with the base model and hierarchical loss for training, we applied weighted accuracy, yielding the results shown in Figure 12b. On average, the results are comparable between the two metrics. However, the weighted class accuracy exhibits a smoother trend compared to the plots in Figure 12a. Notably, the most volatile classes in Figure 12a—dogs and cats— show less fluctuation in their weighted accuracy. This smoothing occurs because individual errors in these classes are, on average, offset by the weighting of errors between more distinct classes, dogs and cars for instance.



(a) Class accuracy

(b) Weighted class accuracy

Figure 12: Differences between the two types of class accuracy

6 Conclusion

This paper presents a thorough analysis on the application of hierarchical loss for classes with hierarchical semantic relationships. Our work provides an optimized implementation of such hierarchical loss which we hope will contribute to the development of new experiments surrounding this type of training strategy for neural networks. To compare our work with traditional loss functions, we made various experiments on research axes we found important to set a clear environment for using this new approach. Specifically, we explored different tree structures for labels and conducted a hyperparameter search to identify the optimal parameters for our new loss. Based on our results, we refined the original hierarchical structure by incorporating other strategies to improve neural networks training such as soft labelling.

Even though no significant improvements over the baseline were observed, we hypothesize that the CIFAR-10 dataset, due to its relatively simple nature, may not provide sufficient complexity to reveal clear trends resulting from our modifications. For future work, we plan to evaluate our approach on more complex datasets, such as CIFAR-100 or Cityscapes, to assess the effectiveness of these modifications under more challenging conditions. Such tasks would allow the development of larger tree structure for labels with further properties that could be challenged. Additionally, we aim to explore other forms of hierarchical embeddings, such as text embeddings, to further investigate whether loss-based approaches offer substantial benefits on other domains than computer vision.

Acknowledgements

This work is supported by CentraleSupélec which we thank for providing us the environment to develop our research. We thank Ms.Chazottes for supervising and guiding us during the whole

duration of this project; we are grateful for working on the subject proposed and we learnt a lot thanks to this experience. We also thank Ms.Bich-Lien Doan and all the professors supervising the MDR curriculum for giving us precious advice during these weeks.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Andrew McCallum, Ronald Rosenfeld, Thomas Mitchell, and Andrew Y Ng. Improving text classification by shrinkage in a hierarchy of classes. 1998.

[2] Samuel Sithakoul, Sara Meftah, and Clément Feutry. Beexai: Benchmark to evaluate explainable ai. 2024. URL `https://arxiv.org/abs/2407.19897`.

[3] Pierrick Bournez and Jules Salzinger. Pushing the limit to near real-time indoor lidar-based semantic segmentation. 2024.

[4] Liulei Li, Tianfei Zhou, Wenguan Wang, Jianwu Li, and Yi Yang. Deep Hierarchical Semantic Segmentation, March 2022. URL `http://arxiv.org/abs/2203.14335`. arXiv:2203.14335 [cs].

[5] Eleonora Giunchiglia and Thomas Lukasiewicz. Coherent Hierarchical Multi-Label Classification Networks, October 2020. URL `http://arxiv.org/abs/2010.10151`. arXiv:2010.10151 [cs, stat].

[6] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pages 516–532. Springer, 2016.

[7] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Learning hierarchical models of scenes, objects, and parts. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1331–1338. IEEE, 2005.

[8] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Describing visual scenes using transformed objects and parts. *International Journal of Computer Vision*, 77: 291–330, 2008.

[9] Tianfei Zhou, Wenguan Wang, Si Liu, Yi Yang, and Luc Van Gool. Differentiable multi-granularity human representation learning for instance-aware human semantic parsing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1622–1631, 2021.

[10] Jônatas Wehrmann, Ricardo Cerri, and Rodrigo C Barros. Hierarchical Multi-Label Classification Networks.

[11] Wenguan Wang, Tianfei Zhou, Fisher Yu, Jifeng Dai, Ender Konukoglu, and Luc Van Gool. Exploring Cross-Image Pixel Contrast for Semantic Segmentation, March 2021. URL `http://arxiv.org/abs/2101.11939`. arXiv:2101.11939 [cs].

[12] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 521–528. IEEE, 2001.

[13] Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A Lord. Making better mistakes: Leveraging class hierarchies with deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12506–12515, 2020.

[14] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38 (7):1425–1438, July 2016. ISSN 2160-9292. doi: 10.1109/tpami.2015.2487986. URL `http://dx.doi.org/10.1109/TPAMI.2015.2487986`.

[15] Rafael S. Pereira, Alexis Joly, Patrick Valduriez, and Fabio Porto. Hyperspherical embedding for novel class classification, 2022. URL `https://arxiv.org/abs/2102.03243`.

[16] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023.

[17] Jean-Christophe Pesquet. A probabilistic approach to hierarchical classification/segmentation. 2024.

[18] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions, 2017. URL `https://arxiv.org/abs/1701.06548`.

[19] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 521–528. IEEE, 2001.

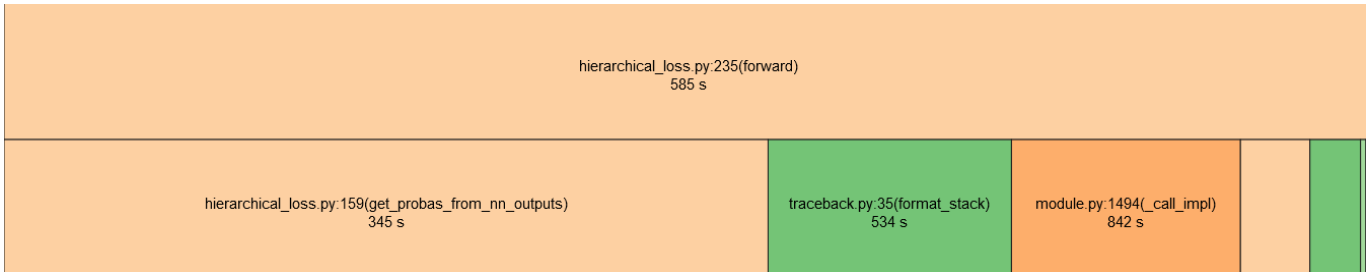
[20] Hristo N. Djidjev, Grammati E. Pantziou, and Christos D. Zaroliagis. Computing shortest paths and distances in planar graphs. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez Artalejo, editors, *Automata, Languages and Programming*, pages 327–338, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. ISBN 978-3-540-47516-3.

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

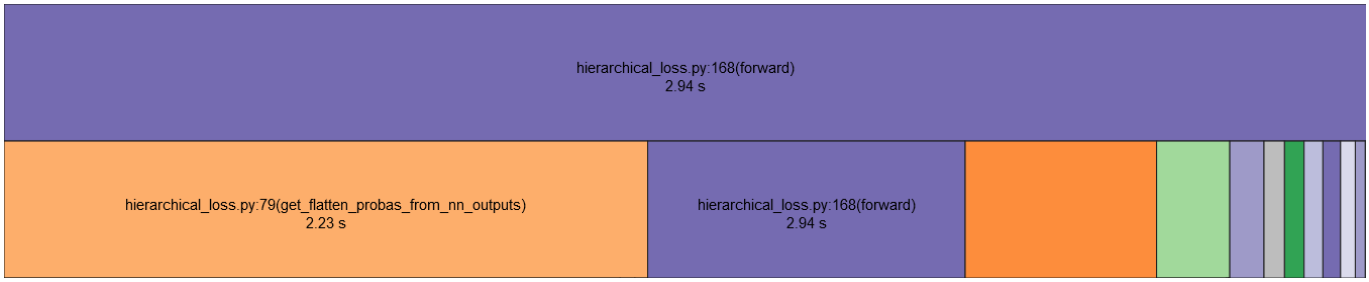
Appendix

A Code optimization

We report the improvement of code acceleration with the chosen tree implementation and tensorisation of the hierarchical loss computation. A comparison is done by profiling the code for previous implementation iterating on all samples and current optimized version. These experiments are performed on a device with an AMD EPYC 7742 64-core @ 3.4GHz CPU and a single 10GB partition of a NVIDIA A100 PCIe GPU.



(a) Total computation time for hierarchical loss with 4 epochs, previous implementation. Total time is 585 seconds.



(b) Total computation time for hierarchical loss with 4 epochs, new implementation. Total time is 2.94 seconds.

We observe an acceleration of $\times 198$ with the current improvements. This performance is justified by the conversion of most of the Python List objects to PyTorch tensors, utilization of matrix operations and batch computation. Previous implementation didn't benefit from the usage of GPU acceleration which accelerates the computation of the loss in parallel.

Such speed-up is necessary to cover a wider range of experiments and opens new possibilities previously impossible because of this computation bottleneck. For example, we explore larger Neural Network models in this work using the new implementation.