MLT-Trans: Multi-level token Transformer for Hierarchical Image Classification

No Author Given

No Institute Given

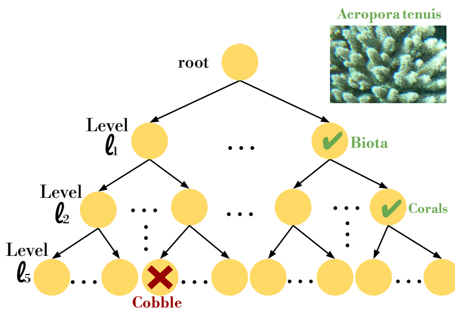
Abstract. This paper focuses on the topic of Multi-level Hierarchical Classification (MLHC) of images, presenting a novel architecture that exploits the "[CLS]" (classification) token within transformers – often disregarded in computer vision tasks. Given the nature of MLHC, where multiple output levels present a varying number of classes, our primary goal lies in utilizing the information of every [CLS] token in a hierarchical manner. Toward this aim, we introduce a Multi-level Token Transformer (MLT-Trans) approach. This model, trained with sharpness-aware minimization and a hierarchical loss function based on knowledge distillation, is capable of being adapted to various transformer-based networks, with our choice being the Swin Transformer as the backbone model in this work. Empirical results across diverse hierarchical datasets confirm the efficacy of our approach. The findings highlight the potential of combining transformers and [CLS] tokens, by demonstrating improvements in hierarchical evaluation metrics and accuracy up to 5.7% on the last level in comparison to the base network, thereby supporting the adoption of the MLT-Trans framework in MLHC.

Keywords: Hierarchical classification · Image processing · Transformer · Class tokens · Hierarchy taxonomy.

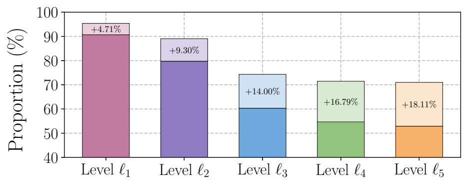
1 Introduction

Traditional image classification models often employ flat classification schemes, treating each category independently and disregarding potential hierarchical relationships that may exist between classes [1]. This lack of hierarchical consideration hinders the ability of algorithms to capture and leverage the underlying semantic structure of complex visual datasets [2]. This limitation is particularly influential in domains such as fine-grained categorization, where classes often exhibit nested relationships within broader taxonomies [3]. Recent works such as HERBS [4] and Metaformer [5] display the importance of incorporating supplementary information, such as background cues or object attributes, to improve overall classification accuracy. However, their models are mainly designed for flat classification, ignoring the potential benefits that could come from incorporating hierarchical relationships as supplementary information.

In contrast to flat classification, Multi-level hierarchical classification (MLHC) aims to correctly classify objects organized in a tree-based taxonomy [6]. MLHC



(a) Snapshot of a complex taxonomy.



(b) Accuracy for each level of Marine-tree.

Fig. 1: **(a)** An image of an "Acropora tenuis" classified by five independent classifiers as a "Biota", "Corals", and on the last level, incorrectly classified as a"Cobble" for the Marine-tree dataset. **(b)** Proportion of correctly classified images (depicted in a darker color) for each level of the taxonomy for Marine-tree dataset and the proportion of images incorrectly classified (depicted in a light color) but for which the other levels in the taxonomy were correctly identified.

has been attracting attention over the past few years mainly because it can provide order and structure to complex real-life datasets. Moreover, it significantly contributes to the development of recommendation systems, image captioning, annotation, scene graph generation, and visual question answering (VQA) leading to more accurate and interpretable data analysis. To the best of our knowledge, the majority of approaches for MLHC frequently neglect the incorporation of taxonomy structures and employ backbone networks whose outputs lack mutual constraint, thereby leading to potential inconsistency in predictions. Furthermore, these methodologies are typically tailored for a predetermined number of hierarchical levels.

While efforts have been made within the MLHC field, there is still considerable scope for future research in developing architectures that address these limitations. To illustrate the aforementioned advantages of MLHC, Figure 1 provides a concrete example featuring the classification of an image by five distinct independent classifiers. Figure 1b displays the proportions of accurate classifications at each hierarchical level, represented in a darker color. Furthermore, these figures include the proportion of images that were misclassified at that particular level but were correctly identified in previous levels represented in a lighter color. If we hypothesize that coarser levels of the hierarchy exhibit higher accuracy compared to finer-grained levels, it becomes evident that promoting communication between these coarser levels can enhance the classification accuracy of the fine-grained levels. This highlights our belief that MLHC warrants further exploration as an appealing solution for addressing such issues.

There have been several methods proposed for MLHC, and they can be categorized according to how the hierarchical structure is explored [7]. In particular, we distinguish: (i) *flat classification approaches*, typically involve a complete disregard for the class hierarchy, wherein predictions are made exclusively at the leaf nodes, with the implicit assumption that all ancestor classes are also attributed to the given instance. This provides an indirect solution to the hierarchical classification problem that could become suboptimal when dealing with a high number of classes, images with low intra-class, low inter-class separability or a combination [3, 8]; (ii) *local classification approaches*, a multi-class classifier is trained for every parent node within the class hierarchy. In the context of CNNs, we can distinguish HD-CNN [9] as one of the first attempts to embed CNNs using a taxonomy. As other predecessors, it lacks versatility and efficiency as it requires training several classifiers; and (iii) *global classification approaches* where in contrast, a single classifier is employed to manage the complete class hierarchy structure. Within this category, we can identify what is commonly referred to as "branch methods," exemplified by models such as Branch-CNN (B-CNN) [10] and Hierarchical-CNN (H-CNN) [11]. These approaches incorporate a backbone network that produces hierarchical-level outputs ("branches") at specific layers. While these methods improved performance compared to their flat classification counterparts, they are still susceptible to inconsistent predictions, primarily due to the absence of inter-level penalization.

As transformers continue to gain efficiency and importance in the field of Machine Learning, there have been some efforts to incorporate a taxonomy within these models. For instance, the Coarse-to-Fine Transformer, as proposed in [8], aims to enhance final classification accuracy. However, similar to other global approaches, it does not account for the hierarchical structure within the data. In contrast, the Nested Hierarchical Transformer (NesT) [12] and the ViT neural tree decoder (ViT-NeT) [13] introduce architectural designs that embrace the concept of taxonomy by generating decision trees. While these models demonstrate promising results, their effectiveness as MLHC methods remains challenging to evaluate because they do not present their results in hierarchical metrics.

The success of transformers can be attributed to several factors, one of them is their robust ability to model long-range dependencies. ViT introduced a special token, denominated [CLS] token or classification token, which was used to aggregate information from the entire sequence of the patch tokens from the image. This token was later replaced by average or global pooling as it would serve the same purpose. Within the domain of image segmentation, recent studies [14, 15] have emphasized the significance of incorporating [CLS] tokens to exploit class-specific information, ultimately yielding state-of-the-art results. This insight has served as motivation for harnessing class-specific information in the context of MLHC. However, our approach extends beyond the singular use of [CLS] tokens; we also seek to model the interrelationships between multiple levels of [CLS] tokens. In this work, we argue that the majority of existing MLHC transformer-based models do not capture attention between different hierarchical levels and therefore, do not learn interactions between them. To tackle this issue, we propose MLT-Trans, which incorporates [CLS] token interactions between several levels of hierarchy. In summary, our main contributions are: (i) We propose MLT-Trans to exploit class-specific transformer attention for MLHC, (ii) We suggest a simple but effective transformer framework denominated Multi-level

Token Transformer, MLT-Trans, which includes a multi-level token strategy to learn class-specific interactions between levels, (iii) as another contribution, we train our proposed framework with sharpness-aware minimization and a hierarchical loss function based on the knowledge distillation loss, and finally, (iv) we performed extensive experimental evaluation on multiple baselines across diverse hierarchical datasets reporting our results in terms of hierarchical metrics.

2 Mathematical notation and MLT-Trans

Multi-level hierarchical classification: The MLHC problem is defined by [16] as learning a mapping function $f : \mathbb{X} \rightarrow \mathcal{Y}$, which assigns to each feature vector $\mathbf{x}^{(i)}$ a prediction vector $\hat{\mathbf{y}}^{(i)} = (\hat{y}^{[\ell_1]}, \hat{y}^{[\ell_2]}, \cdots, \hat{y}^{[\ell_n]})$ s.t. $\hat{y}^{[\ell_i]}$ is the class label that f assigns for each level ℓ_i.
Taxonomy encoding: In contrast to flat classification, where classes are perceived as unrelated entities, hierarchical classification entails the organization of classes within a taxonomic structure. In this paper, our focus is only on *tree* taxonomies, which are organized with a hierarchy structure of n levels ℓ_i, such that $\ell_i \subset \mathcal{Y}$, $\ell_1 \cup \ell_2 \cdots \cup \ell_n = \mathcal{Y}$, and $\forall y_j \in \ell_{i+1}, \exists y_k \in \ell_i$ s.t. $y_k \prec y_j$, where $\prec$ is the "subclass-of" relationship [16]. Lastly, we encode the relationship between two successive levels ℓ_i and ℓ_{i+1} in a taxonomy using an $|\ell_i| \times |\ell_{i+1}|$ matrix $M^{[\ell_i, \ell_{i+1}]}$, where the binary value $M^{[\ell_i, \ell_{i+1}]}_{y_k, y_j} \in \{0(y_k \nprec y_j), 1(y_k \prec y_j)\}$, with $y_k \in \ell_i$ and $y_j \in \ell_{i+1}$.
MLT-Trans: Figure 2 shows an overview of our proposed MLT-Trans. Given an image $\mathbf{I}$, we first follow the conventional SwinTransformer-L [17] to obtain the patch embedding from the last transformer block $\mathbf{Z}$. According to [15], plugging cross-attention mechanisms after the last transformer block avoids large computational costs and additionally, there is no proof that performing these operations in the initial transformer blocks leads to a better performance. To calculate the attention between patch tokens $\mathbf{Z}$ and class tokens for every level ℓ_n, we first define class tokens as a matrix $\mathbf{T}_{\ell_i}$ for each level of the taxonomy. The size of this matrix is the number of classes of each ℓ_i and the embedding dimension of the transformer. To capture interactions between tokens $\mathbf{T}_{\ell_i}$, we calculate the multihead cross-attention [15] between each pair of tokens. Consider H as the number of heads where $h \in \{1, \ldots, H\}$. We have the attention between $\mathbf{T}_{\ell_i}$ and $\mathbf{Z}$ for the h^{th} head as:

$$AT_h = \delta\left(\frac{Q_h(\mathbf{T}_{\ell_{i+1}}) K_h(\mathbf{T}_{\ell_i})^T}{\sqrt{d_{K_h}}}\right) V_h(\mathbf{T}_{\ell_\mathrm{i}}) \tag{1}$$

where δ corresponds to the softmax function and $\sqrt{d_{K_h}}$ is the dimension of the key K_h. This attention mechanism occurs for each pair of subsequent levels. To calculate the attention between all levels, we decided to concatenate these attention outputs and use them to calculate the attention between them and the patch tokens $\mathbf{Z}$ similar to Equation (1). To update our multi-class tokens $\mathbf{T}_{\ell_i}$, we take the concatenated attention output AT and obtain the top-k tokens with the highest activation where $k = \ell_i$. Consequently, we average it on the batch dimension to update our trainable token $\mathbf{T}_{\ell_i}$. Finally, to update our patch tokens $\mathbf{Z}$, we concatenate the norm of the concatenated attention that we calculated before and we represent this as $\tilde{\mathbf{Z}}$.

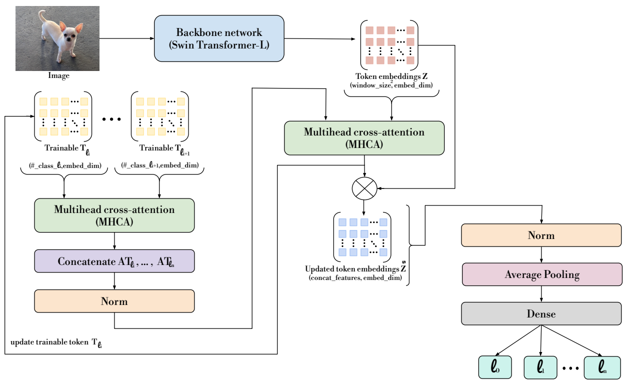


Image
Backbone network (Swin Transformer-L)
Token embeddings Z
(window_size², embed_dim)
Trainable T_{ℓ_i}
Trainable $T_{\ell_{i+1}}$
(#_class_ℓ_i,embed_dim)
(#_class_ℓ_{i+1},embed_dim)
Multihead cross-attention (MHCA)
Concatenate $AT_{\ell_i}, \ldots, AT_{\ell_n}$
Norm
update trainable token T_{ℓ_i}
Multihead cross-attention (MHCA)
Updated token embeddings $\tilde{Z}$
(concat_features, embed_dim)
Norm
Average Pooling
Dense
ℓ_0
ℓ_1
ℓ_n

Fig. 2: MLT-Trans architecture.

Hierarchical loss function: In the realm of hierarchical image classification, the hierarchical loss function emerges as a foundational element [1]. Consequently, in this work we propose the computation of a loss, which is the summation of classification losses across varying tiers of class abstraction. Our strategy takes inspiration from knowledge distillation loss proposed by Hinton et al. [18]. Essentially, this loss function is a pedagogical method in Machine Learning wherein a smaller model (denominated student model) learns from a larger, more complex model (teacher model) by capturing the teacher's learned knowledge, typically in the form of softened probability distributions over classes. This process allows for model compression, enabling more efficient and compact models to retain the insights and generalization abilities of their larger counterparts, which can be especially beneficial for tasks demanding resource-efficient solutions. The knowledge distillation loss (KD_{Loss}) [18] is defined as $KD_{Loss} = CE\big(y, \delta(z_S)\big) + \lambda KL(\delta(z_T/\tau)||\delta(z_S/\tau))$, where z_S are the logits coming from the student model and z_T are the logits from the teacher model. Also, CE stands for cross-entropy, and δ stands for the softmax function; the Kullback-Leibler (KL) divergence loss is used to minimize the discrepancy between the soft outputs probabilities of the student and teacher model. These soft probabilities are obtained by dividing these logits by a temperature factor τ. By transferring KD_{Loss} to the context of MLHC, we can consider the teacher model as the coarser level ℓ_i and the student as the model coming from ℓ_{i+1}. Specifically, we achieve this by replacing student logits by the prediction $\hat{y}^{(j)[\ell_i]}$ and its taxonomic parent $\breve{y}^{(j)[\ell_{i-1}]}$ as the teacher. This will encourage penalization when there is a mismatch between these predictions. Performing this loss for every pair of levels, we obtain HKD_{Loss} as the summation of $\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} CCE\big(y^{(j)[\ell_i]}, \delta(\hat{y}^{(j)[\ell_i]})\big)$ and $\sum_{i=2}^{n}\sum_{j=1}^{m} \lambda KL(\delta(\check{y}^{(j)[\ell_{i-1}]}/\tau)||\delta(\hat{y}^{(j)[\ell_i]}/\tau))$, where CCE denotes the categorical cross-entropy function. λ and τ are hyperparameters that need to be tuned to calibrate the importance of HKD_{Loss} and the effect of temperature on the probabilities. Note that to obtain the taxonomic parent of prediction $\hat{y}^{(j)[\ell_i]}$ we use the taxonomy $M^{[\ell_i,\ell_{i+1}]}$ explained in the last section.

3 Experimental evaluation

Our experiments draw upon a set of six hierarchical datasets and eight models, the specifics of which are detailed below. Note that our results are expressed in terms of hierarchical metrics [6] including Hierarchical Recall (HRecall), Hierarchical Precision (HPrecision), and Hierarchical F1 Score (HF1 Score). The other metrics as consistency and exact match (proportion of examples that are correct and consistent) were also reported by Boone et al. [16].

Datasets: We selected publicly accessible datasets, mainly employed for the purpose of coarse-to-fine classification, and subsequently transformed them into hierarchical datasets. This transformation involved utilizing existing semantic taxonomies wherever available, while also formulating our own taxonomic structures when none were provided. We've summarized this process in Table 1. Specifically, we performed experiments on FGVC-Aircraft [19] and hierarchical versions of CUB-200-2011, Stanford Cars, and Marine-tree where you can find their construction details used by Boone et al. [16].

Table 1: Summary of the hierarchical datasets employed in our experiments.

Dataset	**Stanford Cars**	**CUB 200-2011**	**Marine tree**	**Stanford Dogs**	**FGVC Aircraft**	**Food 101**
Training set	8,144	5,944	118,260	9,600	3,334	60,600
Validation set	4,020	3,000	16,126	2,400	3,333	15,150
Test set	4,021	2,071	26,798	8,580	3,333	25,250
#classes	196	200	60	120	100	101
Taxonomy						
#classes ℓ_1	13	39	2	8	30	3
#classes ℓ_2	113	123	10	120	70	15
#classes ℓ_3	196	200	38	-	100	101
#classes ℓ_4	-	-	46	-	-	-
#classes ℓ_3	-	-	60	-	-	-
#classes ℓ_3	-	-	60	-	-	-

– **Stanford Dogs [20]**: Following the guidelines established by the American Kennel Club (AKC), our classification scheme was aligned to include an additional hierarchical level that corresponds to the AKC's seven distinct dog groups: "hound group", "herding group", "working group", "sporting group", "non-sporting group", "toy group", and "terrier group". We added an additional super-class "wild dogs" for classes such as "dingo", "dhole", and "African hunting dog".

– **Food-101 [21]**: We transformed this dataset into a non-overlapping three-tier hierarchical structure. The initial level comprises three categories: "side dish," "main dish," and "dessert". Subsequently, the second level is comprised of 15 classes: "beef dish", "pork dish", "rice dish", "egg dish," "poultry dish," "pasta dish", "seafood dish", "sandwich", "spread", "appetizer", "salad", "soup", "cake", "pastry," and "cup confections".

For our augmentation and training strategy we followed [4]. The entire experimental setup was implemented using Tensorflow with Keras and was executed on an NVIDIA Tesla V100 GPU with a batch size of 8. Equal weights were assigned to all output components for the loss function. Additionally, we experiment with three different temperature and λ configuration similar to [22] and we selected the best of the three settings **Low(L)**($\tau = 2.5$,$\lambda = 500$), **Medium(M)**($\tau = 5$,$\lambda = 1000$), **High(H)**($\tau = 10$,$\lambda = 2000$). Additionally, our MLT-Trans is trained using sharpness aware minimization (SAM) [23] with default settings.

For our experiments, we utilize eight state-of-the-art networks that display the high accuracy in our datasets in their non-hierarchical representation. Therefore, one network can perform better in one dataset than the other. These baselines are: Swin Transformer, EfficientNetV2, InceptionNext, Uniformer, CoAtNet, CAFormer, TinyViT and MaxViT. You can find more details about the implementation of each baseline on the documentation from KerasCV [24].

Figure 5 shows the comparison between our proposed architecture against the baseline models. From the obtained results, we make the following key observations: (i) Our proposed MLT-Trans improved their backbone network Swin-L in all datasets for almost all metrics. Especially accuracy ℓ_i and Hierarchical F1-Score. (ii) The biggest increase in accuracy is observed at the last level while ℓ_0 is the level with the least increase in accuracy. This indicates that it is easier to improve the finer levels than the coarser levels even though we have a smaller number of classes. This problem could be related to over-fitting and top-down approaches because there is no feedback from the bottom levels to the top level. (iii) Stanford Dogs is the dataset that benefited the most from our proposed transformer, it improved 5.7% on its last level ℓ_2. (iv) Stanford Cars presents an improvement of 3%, 2.7% and 1.7% respectively compared to Swin-L. Also, there is a 2.5% in recall meaning that the model is able to provide examples more relevant to the class. Although there is a small increase in consistency, there is a 3% increase in exact match due to the increase in accuracy for every level. Of all the baseline models, MLT-Trans is the model with the best performance in all metrics. (v) Similarly, for CUB-210-2011 MLT-Trans is the model with the best performance in all metrics but with a smaller increase due to CUB-210-2011 being a dataset that is considered harder to train. The final accuracy is 92% which is only 1% less than the current state-of-the-art models HERBS [4] which also utilizes Swin-L as baseline model with the difference that we are not using extra training information such as background datasets. CAFormer presents slightly higher consistency than our method but our method outperforms their accuracy for all ℓ_i. (vi) MLT-Trans for Food-101 presents the highest accuracy against all models for the last two levels with the exception of ℓ_0 where there is a small decrease and lastly (vii) Marine-tree, the most complex dataset to train, presents a 74.58% on its last level (ℓ_5) establishing a state-of-the-art performance. The performance in all metrics is superior by a large margin with respect to the baseline models. The closest model in performance is its baseline model Swin-L. (vii) The number of parameters for Swin-L is $\approx$ 195M parameters, adding SAM minimization adds another 3M parameters and the complete MLT-trans architecture contains 270M parameters. Additionally, we want to point out that for datasets with a 3-level hierarchy, we only calculated the attention between the first two levels, and for Marine-tree we used 3 levels. Using all the levels would cause the attention calculation to slow the training process without a dramatic increase in performance. In conclusion, we observe that our proposed MLT-Trans manages to improve the results of their baseline model for most datasets but other datasets show that Swin-L might not be the best pick. For instance, for Stanford Dogs our InceptionNext implementation manages to beat the state-of-the-art performance by more than 1%, which opens the possibility of using InceptionNext as the backbone network for our proposed architecture. But for other datasets, InceptionNext might not be the best model. In terms of ablation study, Figure 3 shows the performance of the backbone network Swin-L(m1), the backbone network plus SAM(m2), and finally our proposed MLT-Trans(m3). The combination of Swin-L plus SAM increases both ℓ_1 and ℓ_2 and MLT-Trans increases them slightly more. We have observed that in some datasets, there is a small decrease in consistency compared to the backbone network. This is due to SAM causing a decrease in consistency. MLT-Trans is able to recover it and sometimes improve it. Nevertheless, the increase in accuracy that SAM provides is worth the small decrease in consistency.

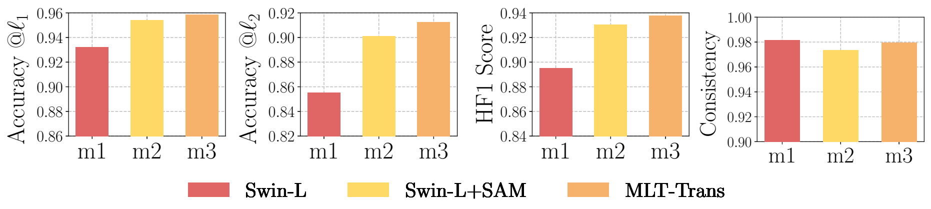
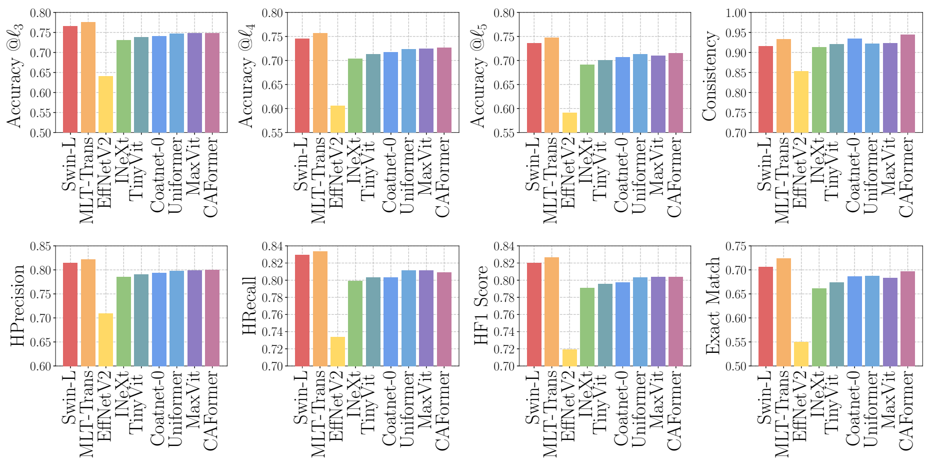
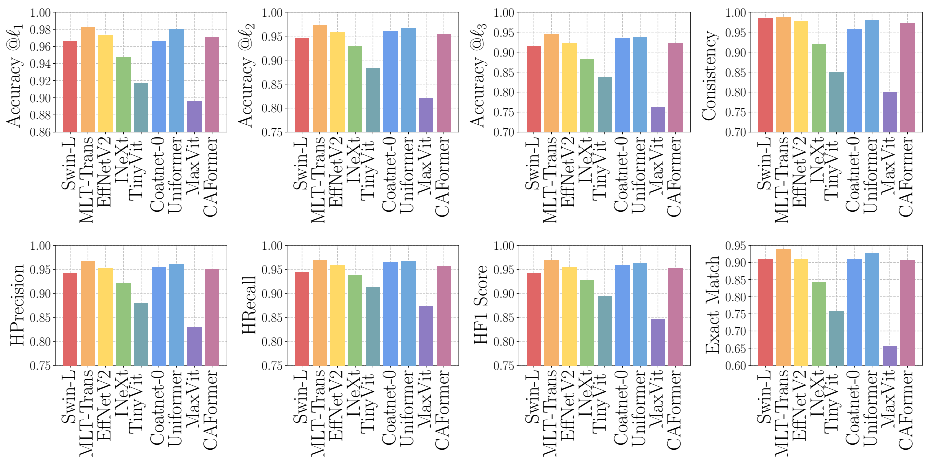


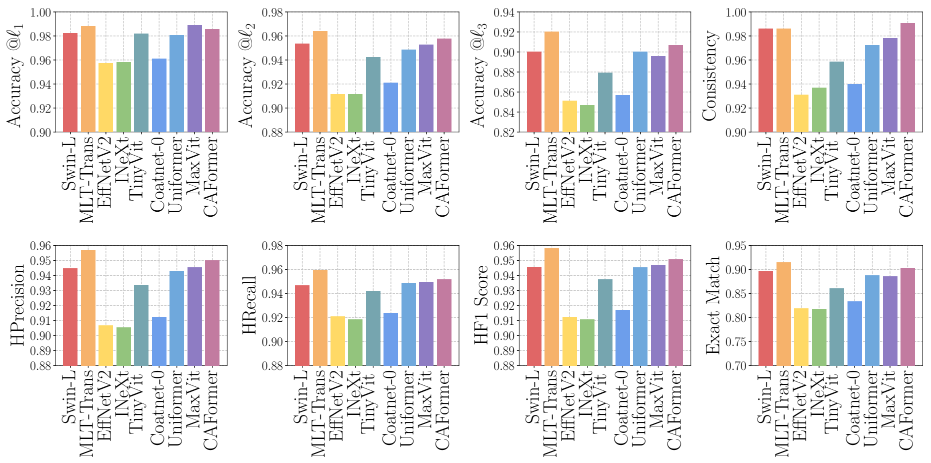
Fig. 3: Ablation study on Stanford Dogs.



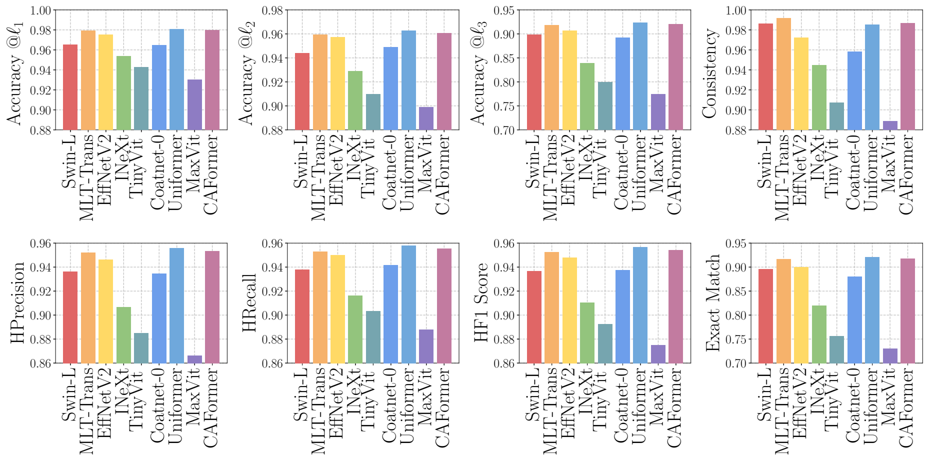
(a) Marine-tree



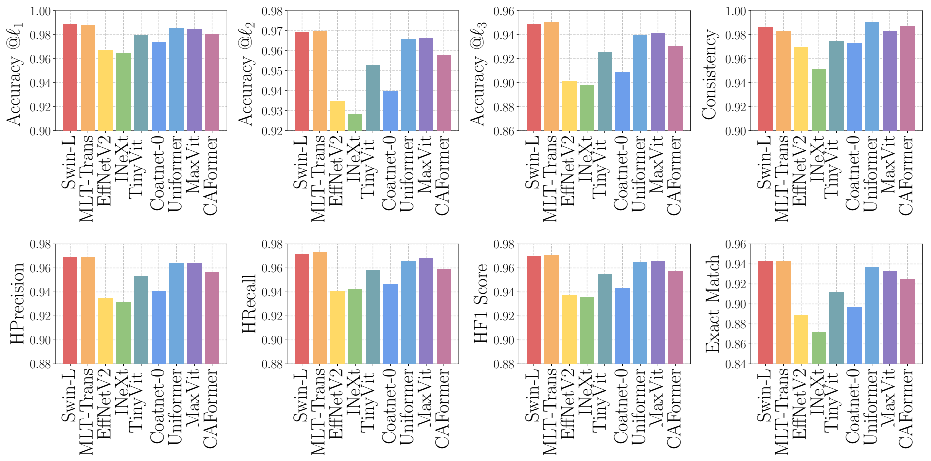
(b) Stanford Cars



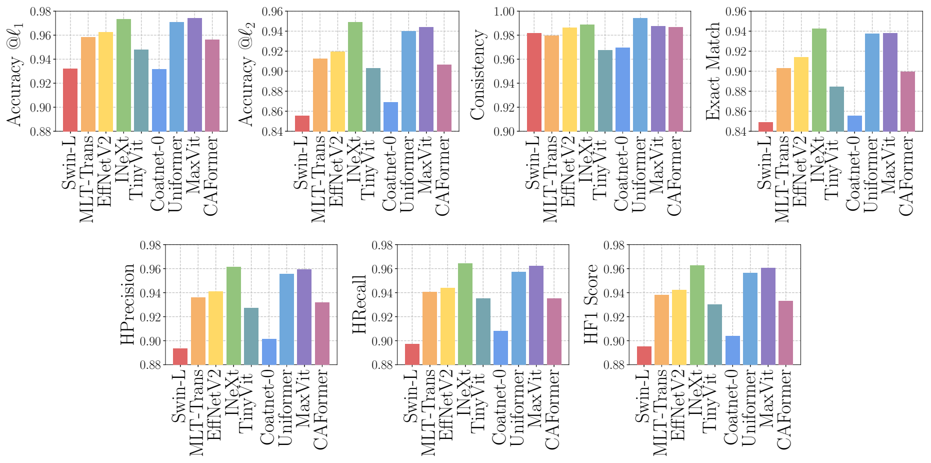
(c) CU-Birds-200-2011



(d) FGVC Aircraft



(e) Food-101



(f) Stanford Dogs

Fig. 5: Numerical performance of our proposed MLT-Trans against other models for all hierarchical datasets.

4 Conclusions and future work

In this work, we introduced a new transformer for Multi-level Hierarchical Classification we denoted Multi-level Token Transformer (MLT-Trans). Our proposed transformer implements an attention mechanism between trainable class tokens per level of a predefined taxonomy plus the addition of a hierarchical loss based on knowledge distillation loss. Focusing on image classification, we presented a thorough experimental evaluation of the performance of our method on six different datasets varying from two levels of hierarchy up to five. Experiments on these datasets show that our proposed architecture improves several hierarchical metrics, most importantly accuracy, showing that MLHC could be a strategy to beat state-of-the-art performances without doubling or triplicating the number of parameters. Future work includes testing our proposed methodology on different backbone networks as well as different hierarchical losses. Additionally, we could improve our attention mechanism to take into account all levels of hierarchical datasets with a large number of classes without being computationally expensive.

References

1. 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.
2. Yuntao Liu, Yong Dou, Ruochun Jin, and Peng Qiao. Visual tree convolutional neural network in image classification. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 758–763. IEEE, 2018.
3. Yuqi Huo, Yao Lu, Yulei Niu, Zhiwu Lu, and Ji-Rong Wen. Coarse-to-fine grained classification. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1033–1036, New York, NY, USA, 2019. Association for Computing Machinery.
4. Po-Yung Chou, Yu-Yung Kao, and Cheng-Hung Lin. Fine-grained visual classification with high-temperature refinement and background suppression. *arXiv preprint arXiv:2303.06442*, 2023.
5. Qishuai Diao, Yi Jiang, Bin Wen, Jia Sun, and Zehuan Yuan. Metaformer: A unified meta framework for fine-grained recognition. *arXiv preprint arXiv:2203.02751*, 2022.
6. Aris Kosmopoulos, Ioannis Partalas, Eric Gaussier, Georgios Paliouras, and Ion Androutsopoulos. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery*, 29(3):820–865, 2015.
7. Carlos N Silla and Alex A Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1):31–72, 2011.
8. Mengzhao Chen, Mingbao Lin, Ke Li, Yunhang Shen, Yongjian Wu, Fei Chao, and Rongrong Ji. Coarse-to-fine vision transformer. *arXiv preprint arXiv:2203.03821*, 2022.

9. Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hd-cnn: Hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
10. Xinqi Zhu and Michael Bain. B-cnn: branch convolutional neural network for hierarchical classification. *arXiv preprint arXiv:1709.09890*, 2017.
11. Yian Seo and Kyung-shik Shin. Hierarchical convolutional neural networks for fashion image classification. *Expert systems with applications*, 116:328–339, 2019.
12. Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, Sercan Ö Arik, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3417–3425, 2022.
13. Sangwon Kim, Jaeyeal Nam, and Byoung Chul Ko. Vit-net: Interpretable vision transformers with neural tree decoder. In *International Conference on Machine Learning*, pages 11162–11172. PMLR, 2022.
14. Lian Xu, Wanli Ouyang, Mohammed Bennamoun, Farid Boussaid, and Dan Xu. Multi-class token transformer for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4310–4319, 2022.
15. Bowen Dong, Pan Zhou, Shuicheng Yan, and Wangmeng Zuo. Towards class interpretable vision transformer with multi-class-tokens. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 609–622. Springer, 2022.
16. Tanya Boone-Sifuentes, Mohamed Reda Bouadjenek, Imran Razzak, Hakim Hacid, and Asef Nazari. A mask-based output layer for multi-level hierarchical classification. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3833–3837, 2022.
17. Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.
18. Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
19. S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.
20. Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, volume 2. Citeseer, 2011.
21. Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 446–461. Springer, 2014.
22. Florian Schmid, Shahed Masoudian, Khaled Koutini, and Gerhard Widmer. Knowledge distillation from transformers for low-complexity acoustic scene classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2022 Workshop*, 2022.
23. Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.
24. Luke Wood, Zhenyu Tan, Ian Stenbit, Jonathan Bischof, Scott Zhu, François Chollet, et al. Kerascv. https://github.com/keras-team/keras-cv, 2022.