

Extreme Automatic Plant Identification Under Constrained Resources

Jose Carranza-Rojas^{1,2}, Ruben Gonzalez-Villanueva¹, Kelvin Jimenez-Morales¹, Kevin Quesada-Montero¹, Esteban A. Esquivel-Barboza¹ and Nicole Carvajal-Barboza¹

¹Costa Rica Institute of Technology, Cartago, Costa Rica

²Microsoft, San Jose, Costa Rica

Abstract

The estimated amount of plant species in our planet Earth is calculated to be around 400,000. In order to create an automatic plant identification system that aims to identify *any plant species on Earth*, machine learning techniques must scale to a high volume of images and species. This leads to Extreme Classification, an area of machine learning that aims to develop models that can classify among hundreds of thousands, or even millions of classes. This work depicts BioMachina's team participation in the PlantCLEF 2022 challenge. Our approach was based on deep learning techniques for constrained environments, where resources are scarce for the creation of large models to deal with the considerable amount of species of the challenge. Additionally to using several training techniques to alleviate resource consumption, we developed a 2-level hierarchical softmax. By simulating a small and inferred plant taxonomy, we allowed the model to learn a 2 level hierarchy of classes on its own, reducing model sizes significantly. Our implementation of hierarchical softmax resulted in position 4 of the overall PlantCLEF 2022 ranking, while keeping model sizes reasonably small and computationally efficient, with a 5.67x reduction of parameters compared to vanilla softmax.

Keywords

Extreme Classification, Hierarchical Softmax, Automatic Plant Identification, Taxonomy

1. Introduction

Automatic plant identification based on images has evolved from a small amount of species [1], to gradually thousands of species [2]. This year's PlantCLEF competition has increased the number of species to 80,000 [3], which is, to our knowledge, the biggest automatic plant identification dataset known to date. Such dataset enters the realm of Extreme Multi-label Classification (XMLC), that aims to classify instances among hundreds of thousands, to millions of categories [4]. As we approach the total number of plant species on Earth, calculated to be around 400,000, the trend is to use bigger and more complex models to fit such large datasets [5]. Such progress is somehow similar to the Natural Language Processing (NLP) domain, where

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

✉ jcarranza@itcr.ac.cr (J. Carranza-Rojas); rgonzalezv@estudiantec.cr (R. Gonzalez-Villanueva);

kjjimenez@estudiantec.cr (K. Jimenez-Morales); kevin707qm@estudiantec.cr (K. Quesada-Montero);

eesquivel@estudiantec.cr (E. A. Esquivel-Barboza); nicole.carvajal.b@estudiantec.cr (N. Carvajal-Barboza)


🆔 0000-0002-9177-9173 (J. Carranza-Rojas); 0000-0001-8044-3474 (R. Gonzalez-Villanueva); 0000-0002-0263-0677

(K. Jimenez-Morales); 0000-0003-1110-2769 (K. Quesada-Montero); 0000-0002-2100-9712 (E. A. Esquivel-Barboza);

0000-0002-2957-5698 (N. Carvajal-Barboza)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

bigger and bigger transformer-based models have been the norm for a while now, such as BERT [6]. However, training such large models has environmental and financial costs associated with them [7]. In the plant identification domain, we believe there is a need to think beyond brute force and bigger models. There is also a need to decrease model sizes, complexity, and memory footprint, to save on computing and environmental costs. Additionally, making efficient models will have a positive impact on the deployment of automatic plant identification systems, such as Pl@ntNet [8]. Some techniques to reduce model sizes, such as knowledge distillation [9] and quantization [10], have been explored in related machine learning domains. However, very little work has focused on reducing plant identification model sizes and memory footprints.

The plant domain offers a unique setting where a class hierarchy exists, known as the plant taxonomy. It has not been thoroughly studied yet in machine learning, although some work has been done in such domain [11, 12]. The intuition is that, guided by the taxonomy, one may be able to improve model performance, and maybe decrease model sizes. As an example, in the NLP domain, Language Model (LM) training exhibits similar problems, as the amount of human language tokens is large. Some work has been done towards training with reduced logits layers, by implementing a hierarchical softmax, reducing time complexity from linear to $\mathcal{O}(\log(n))$ for token prediction.

The present work depicts the participation of the BioMachina team in the PlantCLEF 2022 challenge. Our focus was to try to get the best possible results with limited resources, as we did not count on a large GPU cluster to deal with this year’s amount of species. Therefore, our primary focus was on keeping models small to fit smaller computing resources. We applied a particular implementation of a 2-level hierarchical softmax, reducing by 5.67x the number of model parameters.

Our contributions are the following:

- The exploration of a different way to model the species probability distribution, such that fewer model parameters are needed, via a 2-level hierarchical softmax.
- The experimental comparison of using normal softmax versus using hierarchical softmax, in an extreme plant classification challenge, in terms of Moving Average Mean Reciprocal Rank (MA-MRR), parameter size, and memory footprint.
- Help democratize plant identification model development and raise awareness of the potential consequences of using brute force to deal with bigger plant datasets, and provide viable options to keep model sizes at bay under constrained computing environments.

The rest of the document is as follows: Section 2 contains the related work. Section 3 depicts the datasets, architectures used, and how we decreased models sizes. Section 4 contains a description of the experiments with their respective results, and discussion. Section 5 contains the conclusions we draw from our results, and lists potential future work in this line of research.

2. Related Work

PlantCLEF 2013 included for the first time other organs beyond leaves, such as fruits and flowers [1]. By 2017, the PlantCLEF challenge’s dataset had 10,000 species, leading to this year’s

challenge with 80,000 species [3]. Year by year, the challenge draws near to the total number of plants species on Earth.

Previous work on hierarchical loss functions has been studied in NLP, given the large number of tokens that particular human languages have [13]. In particular, hierarchical softmax has been widely used for training language models [14]. It is defined over a binary tree, where each node has only 2 children, and all language tokens are in the leaves of the tree. Clearly, this may not be well aligned with our taxonomy needs, as classes at a given level of the plant taxonomy, such as family or genera, may not have only 2 children. In order to approximate part of the plant taxonomy, a variation of the hierarchical softmax would be needed.

Several research has been done on how to use the plant taxonomy to inject additional knowledge into the learning process. In [11] the authors created 3 different architectures where a classifier is needed for family, genus and species. Each architecture connects the 3 classifiers in different fashions. Other work such as the Taxonomy Softmax [15] attempts to create a new loss function using a new type of softmax where probabilities are derived based on the taxonomy, however the results were not as good as expected. It also uses the taxonomy as ground truth, assuming the taxonomy is perfectly defined. In [12] the authors use 3 different models, each for family, genus and species, to improve the domain adaptation from herbarium to field images. They, however, did not aim to reduce model sizes, as they include 3 classifiers for each taxa, increasing the model size.

To our knowledge, no other work has been attempted to automatically learn a plant taxonomy to reduce model sizes. In this work, we borrow ideas from training big language models on NLP tasks, where hierarchical softmax is used to improve training times. We implement a 2-level hierarchical softmax, allowing the model to learn a 2-level taxonomy on its own, while keeping the number of parameters small compared to traditional softmax with cross entropy loss.

3. Methodology

Our methodology is based on the idea of keeping model sizes at bay, while trying to get the best results possible. The main technique used to reduce model size was a 2-level hierarchical softmax, but we also used Automatic Mixed Precision (AMP) to process bigger batches, as well as Gradient Accumulation. The following subsections depict the architectures, equipment, and software used, and this year's dataset, as well as the 2-level hierarchical softmax implementation.

3.1. Dataset

PlantCLEF 2022 challenge provides a large dataset based on 80,000 species. The trusted subset contains a total of 2.9 million images. Such trusted subset has been confirmed by human experts. A second subset is provided, called the web subset, which has 57,314 species and 1,065,129 images. Such web images have not been fully confirmed by human experts [3]. This dataset, to our knowledge, is the biggest dataset for plant identification to date. Consequently, it converts this year's task in an extreme classification task.

3.2. Architectures

Although Attention-based architectures, in particular, Visual Transformers [16] have been getting lots of attention, we used smaller known Convolutional Neural Network (CNN) architectures for our runs. This due to constrained resources. The models of choice were ResNet and EfficientNet. Both are relatively small compared to other models, and their performance is known to be good in generalistic challenges such as ImageNet.

ResNet Residual learning blocks were introduced by He et al. [17] in order to train deeper neural networks. The residual function tackles the degradation problems that arose with adding more layers. These type of networks learn residual functions that reference the layer’s inputs as shortcuts, as shown in Figure 1. Formally, the building block is defined as shown in the Equation 1, where x and y are the input and output vectors of the layers, respectively, and the function \mathcal{F} represents the residual mapping to be learned.

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{1}$$

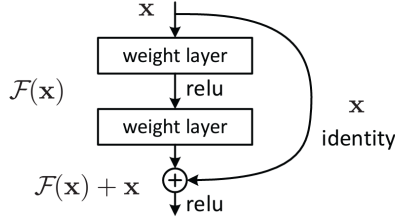


Figure 1: Residual learning building block [17].

We used two variants of the Residual Networks, namely ResNet50 and ResNet101, referring to residual nets with a depth of 50 and 101 layers, respectively.

EfficientNet Scaling processes of a CNN are done to produce better performance, such as scaling by their depth (i.e. ResNet [17]). Tan and Le [18] proposed the *compound scaling method*, which scales up a CNN by using a constant ratio to scale each dimension of network width/depth/resolution.

A CNN can be defined as shown in the Equation 2, where $\mathcal{F}_i^{L_i}$ denotes layer F_i is repeated L_i times in stage i , $\langle H_i, W_i, C_i \rangle$ denotes the shape of input tensor X of layer i [18].

$$\mathcal{N} = \bigodot_{i=1 \dots s} \mathcal{F}_i^{L_i}(X_{\langle H_i, W_i, C_i \rangle}) \tag{2}$$

Different from regular CNN designs, this scaling mechanism expands the network length (L_i), width (C_i), and resolution (H_i, W_i) without changing the \mathcal{F}_i in the baseline. The approach can be formulated as an optimization problem, where the model accuracy is being maximized for any given resource constraints, which can be seen in the Equation 3, where the w, d, r are coefficients for scaling the network width, depth and resolution and $\hat{\mathcal{F}}_i, \hat{L}_i, \hat{H}_i, \hat{W}_i, \hat{C}_i$ are

predefined parameters. The result of this meta-learning optimization is the EfficienNets, which outperform most of the CNN models using a considerably less number of parameters [18].

$$\begin{aligned} & \max_{d,w,r} \text{Accuracy}(\mathcal{N}(d, w, r)) \\ \text{s.t. } & \mathcal{N}(d, w, r) = \bigodot_{i=1 \dots s} \hat{\mathcal{F}}_i^{d \cdot \hat{L}_i} (X_{\langle r \cdot \hat{H}_i, r \cdot \hat{W}_i, w \cdot \hat{C}_i \rangle}) \end{aligned} \quad (3)$$

3.3. Hardware and Software

We worked in a fairly constrained environment proportionally to the size and complexity of the task. We used the cloud computing platform Vast.ai, with instances ranging from 1 to 2 NVIDIA RTX 3090 GPUs, with 16 to 32 GBs of RAM on average. This represents a good workstation, accessible to most people, but not a research cluster. Software-wise, we developed all models using Pytorch and Pytorch Lightning, and the code is publicly available [here](#).

3.4. Reducing output sizes

As the number of species for the task is 80,000, normal softmax would require a logits output of the same size, leading to a linear increase of model parameters in the logits layers, regardless of the model used. Traditionally, image classifiers rely on cross-entropy loss for training, shown in Equation 5, where y_c is the ground truth one-hot vector. Equation 4 depicts the softmax calculation, where M is the total of classes at hand. The input vector z represents the logits from a backbone model. Clearly, softmax has a linear time complexity $\mathcal{O}(M)$.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^M e^{z_j}} \quad \text{for } i = 1, 2, \dots, M \quad (4)$$

$$\mathcal{L} = -y_c \log(\sigma(z_c)) \quad (5)$$

The problem with extreme classification scenarios is that the value of M is large. In this case, $M \in \mathbb{R}^{80,000}$ species, which causes the model to grow importantly, as the last layer of the neural network will be of size $|L^{(n-1)}| \times 80,000$, where $|L^{(n-1)}|$ is the size of the layer before logits. For perspective, EfficientNet B4 increases its size with $2,048 \times 80,000$, summing up around 160 million parameters *only* for logits computation.

3.4.1. Hierarchical Softmax

Aiming to reduce the amount of parameters related to the output of the model, we defined a 2-level hierarchical softmax. The implementation is in Pytorch and based on the implementation found [here](#). Our 2-level hierarchical softmax differs to the original hierarchical softmax definition from [13] since it does not rely on a binary tree. The main reason is because the plant taxonomy cannot be implemented as a binary tree, as nodes may have more than 2 children. Instead, our implementation relies on 2 smaller layers, each one with its own softmax distribution, where the top layer leads to the bottom layer, and both probabilities are maximized. Each node of the

top layer has more than 2 children, allowing to place sibling species under the same learned top layer node, effectively simulating a potential 2 layer taxonomy. The comparison of an EfficientNet B4 with and without 2-level hierarchical softmax is shown in Figure 2. Notice how the hierarchical softmax can be easily plugged into other models by reducing the increasingly bigger logits layer of the backbone.

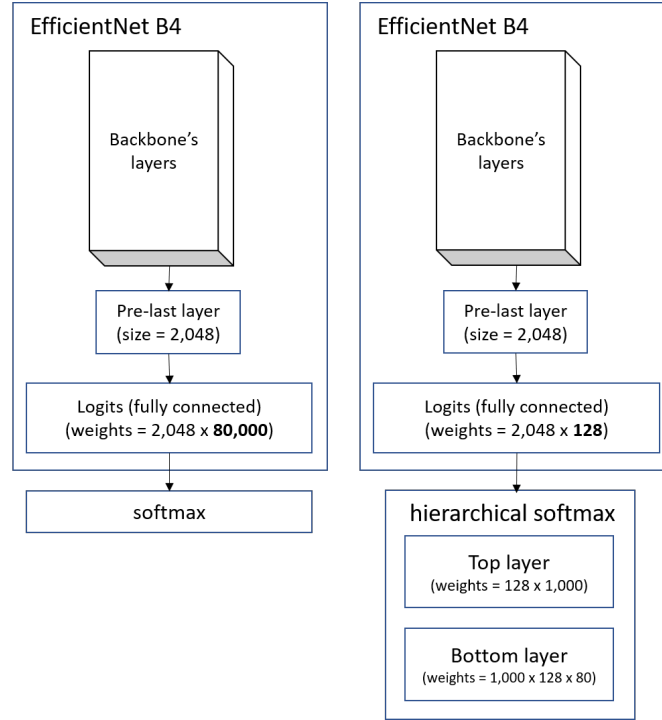


Figure 2: Architectural changes to add hierarchical softmax into EfficientNet B4. The left diagram shows a traditional fine-tuning scenario, with a pre-logits layer’s output of 2,048 and a vanilla softmax of size $M = 80,000$, forcing a same sized logits layer. To the right, a hierarchical softmax with logits $L^{(n)}$ layer is reduced to an arbitrary, smaller number than M . Then it connects to the 2 levels of the hierarchical softmax. The logits layer is reduced without a bottleneck effect.

The top layer $\sigma^{(t)}$, depicted by Equation 6, simulates a higher taxonomic level of the taxonomy, although it does not have exactly the real amount of families or genera in the dataset. The input z is the actual output of the logits $L^{(n)}$ of a backbone model, which would normally be injected in a plain softmax. We reduce such logits size to $z \in \mathbb{R}^{128}$, which is controlled as a hyperparameter. Such size does not have a bottleneck effect since we distribute the M species through 2 levels. The weights $W^{(t)} \in \mathbb{R}^{128 \times 1,000}$ learn how to route z to the right node of the top layer, with $b^{(t)}$ been its corresponding bias.

$$\sigma^{(t)} = \sigma(zW^{(t)} + b^{(t)}) \quad (6)$$

The bottom layer $\sigma^{(b)}$, shown in Equation 7, is another plain softmax of size 80. The input z is indexed by the biggest probability of $\sigma^{(t)}$. The weights $W^{(b)} \in \mathbb{R}^{1,000 \times 128 \times 80}$ learn the

mapping from one of the nodes in the top layer, to its children in the bottom layer. Again, the middle 128 size of the weights is another hyperparameter.

$$\sigma^{(b)} = \sigma(z_{\max(\sigma^{(t)})} W^{(b)} + b^{(b)}) \quad (7)$$

We chose an arbitrary number of nodes for the top layer of $\sigma^{(t)} \in \mathbb{R}^{1,000}$, where each top node is mapped to one of the $\sigma^{(b)} \in \mathbb{R}^{80}$ children in the bottom layer, accounting for all 80,000 species. Another potential distribution of nodes, for reference, would be 400 in the top layer, and 200 in the bottom layer.

The loss $\mathcal{L}^{(h)}$ for the 2-level hierarchical softmax is shown in Equation 8. Both top and bottom layers route the right species y_c by using as ground truth $y_t = y_c/80$ and $y_b = y_c \% 80$ respectively. This effectively forces the hierarchical softmax to learn mappings of the right class index on both top and bottom layers. Additionally, log is used for numerical stability. Our 2-level hierarchical softmax reduces dramatically the number of parameters needed to predict the correct class, as each species does not have dedicated parameters, but they are shared between species.

$$\mathcal{L}^{(h)} = -[y_t \log(\sigma^{(t)}) + y_b \log(\sigma^{(b)})] \quad (8)$$

3.5. Additional techniques for constrained resources

Additional to the hierarchical softmax explained in Section 3.4.1, we also used several techniques to limit the amount of memory, time, and GPU usage needed to train such complex models, while helping with generalization for unseen samples.

Automatic Mixed Precision (AMP) AMP uses both single- and half-precision representations, instead of only single-precision format [19]. This strategy nearly halves memory requirements by accessing half the bytes, and speeds up math-intensive operations like linear and convolution layers. It enables training larger mini-batches while keeping the accuracy achieved with single precision.

Batch Accumulation It is known that a small batch size usually results in performance degradation, especially in tasks where the number of classes is large [20]. An effective way to increase the batch size without compromising memory is the use of accumulated gradients. This method allows to run K batches of size N before doing the backward process, resulting in a large effective batch size of size $K \times N$. We used Pytorch Lightning [21] with a accumulate grad batches of 32.

Gradient Clipping Gradient clipping is used to solve the exploding gradients problem, by clipping the value of gradients to a certain threshold. We used Pytorch Lightning [21] with a gradient clip value of 9.

Table 1

Results of BioMachina’s runs on the PlantCLEF 2022 challenge with their respective model sizes. Best results were achieved by a small ResNet50 model with traditional softmax, pretrained with the web subset. Notice, however, how the HEfficientNetB4 got similar results with a small fraction of other model’s parameters.

Size (Millions)							
Model Name	Pretraining	Softmax Type	Backbone	Final Layer	Total	Memory	MA-MRR
EfficientNetB4	ImageNet	Normal	19	143.3	160	321.9MB	0.412
HEfficientNetB4	ImageNet	Hierarchical	17.8	10.4	28.2	56.4MB	0.419
ResNet50	ImageNet	Normal	25.6	163.84	180	749.7MB	0.438
ResNet50	Web	Normal	25.6	163.8	180	749.7MB	0.460
ResNet101	Web	Normal	98	163.8	206	825.6MB	0.450

4. Experiments and Results

Our experiments aimed to measure model performance under limited resources for a large number of species. We measured model and final layer parameter sizes, memory footprint, and MA-MRR using the official submission tool from the challenge.

4.1. Efficacy of adding more layers

We experimented with different depths of ResNet to see how the model would scale up given the large amount of classes. In general, by just adding more layers from ResNet50 to ResNet101, the task at hand did not improve dramatically, as shown in Table 1. This suggests that brute force may not be the best approach as the amount of species grows. We pre-trained both ResNet versions with the Web subset, improving the MA-MRR compared to runs without such pre-training, getting the best results we got for the challenge with 0.46 for ResNet50 and 0.45 for ResNet101. ResNet101, regardless of been a bigger model, did not provide better results. From this experiment, we also noticed the positive impact of using the Web subset for pre-training, with an increase in MA-MRR of 0.22 for ResNet50.

4.2. Softmax versus hierarchical softmax

We compared the performance of the EfficientNet with and without hierarchical softmax. We used EfficientNet as it is smaller than ResNet, but also because it shows better results in the ImageNet competition proportionally to its size. We used the B4 version of EfficientNet, as it was reasonable for our computing resources. Table 1 shows the comparison of EfficientNet B4 with and without hierarchical softmax with respect to number of parameters, memory footprint and MA-MRR. Notice how the use of hierarchical softmax dramatically reduces the number of model parameters up to 5.67x in the case of EfficientNet B4, from 160M to 28.2M parameters, while not sacrificing MA-MRR. This suggests that hierarchical softmax could be used with different sized models to reduce the complexity of the last layers while keeping similar performance.

5. Conclusions and Future Work

The usage of techniques such as hierarchical softmax reduces dramatically the size of the final model in terms of parameters and memory footprint, with a 5.67x reduction in the case of the PlantCLEF 2022 challenge, without affecting performance. As the number of species in the challenge grows each year, similar techniques may be worth exploring to keep model sizes at bay. In consequence, training such smaller models requires less time and computation power, making training more accessible to researchers and engineers. It may also positively impact the deployment of more extensive plant identification systems, even aiming to place models on mobile devices while still identifying thousands of species.

There are, however, additional lines of research worth exploring beyond this work. We developed a 2-level hierarchical softmax, but we would like to explore more depth in the hierarchy. We explored learning a small 2-level taxonomy, however it would be interesting to study how to leverage the existing plant taxonomy for additional supervision signal.

Additionally, applying other techniques such as knowledge distillation and quantization may further compress the memory footprint of such models. From the PlantCLEF challenge perspective, most likely transformers and other bigger backbone models may have provided better MA-MRR results. The techniques discussed here could also be applied to them, reducing their sizes while keeping high performance.

References

- [1] H. Goëau, A. Joly, P. Bonnet, V. Bakic, D. Barthélémy, N. Boujemaa, J.-F. Molino, The imageclef plant identification task 2013, in: Proceedings of the 2nd ACM International Workshop on Multimedia Analysis for Ecological Data, MAED '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 23–28. URL: <https://doi.org/10.1145/2509896.2509902>. doi:10.1145/2509896.2509902.
- [2] H. Goëau, P. Bonnet, A. Joly, Plant identification based on noisy web data: the amazing performance of deep learning (LifeCLEF 2017), in: CLEF: Conference and Labs of the Evaluation Forum, volume CEUR Workshop Proceedings, Dublin, Ireland, 2017. URL: <https://hal.archives-ouvertes.fr/hal-01629183>.
- [3] H. Goëau, P. Bonnet, A. Joly, Overview of PlantCLEF 2022: Image-based plant identification at global scale, in: Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, 2022.
- [4] R. Babbar, B. Schölkopf, Dismec: Distributed sparse machines for extreme multi-label classification, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 721–729. URL: <https://doi.org/10.1145/3018661.3018741>. doi:10.1145/3018661.3018741.
- [5] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Durso, H. Glotin, R. Planqué, W.-P. Vellinga, A. Navine, H. Klinck, T. Denton, I. Eggel, P. Bonnet, M. Šulc, M. Hruz, Overview of lifeclef 2022: an evaluation of machine-learning based

- species identification and species distribution prediction, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2022.
- [6] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *ArXiv abs/1810.04805* (2019).
- [7] E. M. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell, On the dangers of stochastic parrots: Can language models be too big? , in: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 610–623. URL: <https://doi.org/10.1145/3442188.3445922>. doi:10.1145/3442188.3445922.
- [8] A. Joly, A. Affouard, M. Chouet, B. Deneu, J. Estopinan, H. Goëau, H. Gresse, J.-C. Lombardo, T. Lorieul, F. Munoz, M. Servajean, P. Bonnet, Pl@ntNet, ten years of automatic plant identification and monitoring, in: IUCN - Congrès mondial de la nature, IUCN, Marseille, France, 2021. URL: <https://hal.inrae.fr/hal-03343235>.
- [9] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015. URL: <http://arxiv.org/abs/1503.02531>, cite arxiv:1503.02531Comment: NIPS 2014 Deep Learning Workshop.
- [10] A. Polino, R. Pascanu, D. Alistarh, Model compression via distillation and quantization, in: International Conference on Learning Representations, 2018. URL: <https://openreview.net/forum?id=S1XoIQbRW>.
- [11] J. Carranza-Rojas, A. Joly, H. Goëau, E. Mata-Montero, P. Bonnet, Automated Identification of Herbarium Specimens at Different Taxonomic Levels, in: A. Joly, S. Vrochidis, K. Karatzas, A. Karppinen, P. Bonnet (Eds.), *Multimedia Tools and Applications for Environmental & Biodiversity Informatics, Multimedia Systems and Applications*, Springer International Publishing, 2018, pp. 151–167. URL: https://doi.org/10.1007/978-3-319-76445-0_9. doi:10.1007/978-3-319-76445-0_9.
- [12] J. Villacis, H. Goëau, P. Bonnet, A. Joly, E. Mata-Montero, Domain adaptation in the context of herbarium collections a submission to plantclef 2020, 2020. URL: <https://www.imageclef.org/PlantCLEF2020>.
- [13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, p. 3111–3119.
- [14] A. Mnih, G. E. Hinton, A scalable hierarchical distributed language model, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), *Advances in Neural Information Processing Systems*, volume 21, Curran Associates, Inc., 2008. URL: <https://proceedings.neurips.cc/paper/2008/file/1e056d2b0ebd5c878c550da6ac5d3724-Paper.pdf>.
- [15] J. Carranza-Rojas, Towards Multi-Level Classification in Deep Plant Identification, Ph.D. thesis, Instituto Tecnológico de Costa Rica, 2018. URL: <https://hdl.handle.net/2238/10341>.
- [16] A. Kolesnikov, A. Dosovitskiy, D. Weissenborn, G. Heigold, J. Uszkoreit, L. Beyer, M. Minderer, M. Dehghani, N. Houlsby, S. Gelly, T. Unterthiner, X. Zhai, An image is worth 16x16 words: Transformers for image recognition at scale, 2021.
- [17] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015. URL: <https://arxiv.org/abs/1512.03385>. doi:10.48550/ARXIV.1512.03385.
- [18] M. Tan, Q. V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks

- (2019). URL: <https://arxiv.org/abs/1905.11946>. doi:10.48550/ARXIV.1905.11946.
- [19] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, H. Wu, Mixed precision training, 2017. URL: <https://arxiv.org/abs/1710.03740>. doi:10.48550/ARXIV.1710.03740.
- [20] D. Synn, X. Piao, J. Park, J.-K. Kim, Micro batch streaming: Allowing the training of dnn models using a large batch size on small memory systems, 2021. URL: <https://arxiv.org/abs/2110.12484>. doi:10.48550/ARXIV.2110.12484.
- [21] W. Falcon, The PyTorch Lightning team, PyTorch Lightning, 2019. URL: <https://github.com/PyTorchLightning/pytorch-lightning>. doi:10.5281/zenodo.3828935.

6. Online Resources

The code of this research is available via

- GitHub