ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240730

Advancing ConvNet Architectures: A Novel XGB-Based Pruning Algorithm for Transfer Learning Efficiency

Igor Ratajczyk[a,*,1] **and Adrian Horzyk**[a,1]

[a]AGH University of Krakow
ORCID (Igor Ratajczyk): https://orcid.org/0009-0004-9179-3110, ORCID (Adrian Horzyk): https://orcid.org/0000-0001-9001-4198

Abstract. This paper introduces a novel pruning method designed for transfer-learning models in computer vision, leveraging XGBoost to enhance model efficiency through simultaneous depth and width pruning. Contemporary computer vision tasks often rely on transfer learning due to its efficiency after short training periods. However, these pre-trained architectures, while robust, are typically overparameterized, leading to inefficiencies when fine-tuning for simpler problems. Our approach utilizes Parallel Tree Boosting as a surrogate neural classifier to evaluate and prune unnecessary convolutional filters in these architectures effectively. This method addresses the challenge of interference that is often present in transfer-learning models and enables a significant reduction in the size of the model. We demonstrate that it is feasible to achieve both depth and width pruning concurrently, allowing for substantial reductions in model complexity without compromising performance. Experimental results show that our proposed XGB-based Pruner not only reduces the model size by nearly 50% but also improves the accuracy by up to 9% in test data compared to conventional models. Moreover, it can achieve a dramatic reduction in model size by over a hundred times with minimal accuracy loss, proving its effectiveness across various architectures and datasets. This approach marks a significant advancement in optimizing transfer-learning models, promising enhanced performance in real-world applications.

1 Introduction

In recent years, transfer learning has dominated deep learning tasks in natural language processing and computer vision. The transfer learning approach in computer vision assumes that various tasks require similar low-level features, such as edges, along with higher-level features, such as corners along triangles or circles and their compositions. This assumption has been proven correct in multiple problems, as transfer learning is widely applicable to various tasks. A plethora of pre-trained models are available via TensorFlow [1], HuggingFace, or Ultralytics [15], [16], and they are go-to solutions for deep learning tasks, especially in the case when available problem-specific training datasets are small.

An essential problem associated with deep learning and transfer learning is overparametrization [7]. Analysis of this problem yields Lottery Ticket Hypothesis [10], [11], which states that there exists a subnetwork capable of achieving similar or sometimes better results than the original network. This hypothesis's theoretical and experimental analyses result in Neural Architecture Search [2], [27], and approximate pruning algorithms.

Pruning algorithms may not change the overall model size or can reduce its width. Recent pruning methods show that reducing the depth of a model is also possible. However, pruning algorithms bother to prune transfer-learning models efficiently due to their already well-trained architectures.

In this paper, we address the following issues:

- Transfer learning uses architectures trained to solve typically more complex problems than the solved one, resulting in the possibility of reducing some convolutional filters that are unnecessary - we propose a heuristic for pruning transfer-learning backbones.
- Depth pruning was not used in parallel with width pruning yet - we propose a method of doing it.

The second section provides an overview of related state-of-the-art (SOTA) methods of pruning and eXtreme Gradient Boosting for trees. The third section introduces our proposed XGB-based Pruner algorithm. The fourth section focuses on the idea of depth pruning using the XGB-based Pruner. The fifth section presents the results of numerical experiments, while the sixth section compares the results of the proposed algorithm with those of other SOTA approaches. The last section provides general conclusions about the possibilities of the proposed algorithm.

2 Related Works

Pruning algorithms aim to reduce the size of the model, the used memory, and computational complexity by removing unnecessary parameters. Pruning is especially applicable for ConvNets as most model parameters are located in the convolutional backbones. Most pruning methods require multiple fine-tuning stages of the model after pruning to achieve satisfactory results [8]. Considering the well-chosen pruning rate, the entire process can result in a more effective and smaller network than the baseline model before pruning [3].

2.1 Structured, unstructured and depth pruning

Usually, pruning methods are divided into structured and unstructured pruning. Unstructured pruning refers to methods that aim to prune particular weights and thus reduce interference, i.e. redundant signals that introduce noise in an inference process in a model by

* Corresponding Author. Email: igor@ratajczyk.eu
[1] 1st author contribution 60%, 2nd author contribution 40%.

setting selected weights to 0. Unstructured pruning generally reduces neither computational complexity nor memory consumption. Structured pruning aims to prune entire filters and respective channels by removing them from the tensors. It also reduces the model size by decreasing the convolutional tensor's width (filter dimension), i.e., structured pruning may be considered width pruning. Structured pruning may be constrained when preserving specific layer shapes, e.g. in residual connections.

In [5], a novel pruning approach was proposed in which the authors prune models by removing entire layers. It was achieved by training the dense neural classifier on top of the n-first convolutional layers and then assessing its performance. The best sub-model, which consists of an optimal depth backbone, is then chosen as a final solution. Such an approach is computationally demanding due to the necessity of training multiple neural classifiers. Moreover, width pruning cannot be performed in parallel.

2.2 Model-based and data-based pruning

Pruning methods differ in the heuristics on which the pruning decision is made. Thus, they can be divided into model-based pruning heuristics and data-based heuristics. Common model-based pruning heuristics utilize the magnitude of the weight or filter to determine the utility of the weight or filter, respectively. This approach can be implemented as unstructured [12] and structured [17] pruning. This predicate assumes that the magnitude of useless filters is decreased during training. These methods assume that the poorly formed filter can only be identified by considering the filter itself. Their utility for transfer-learning backbones is limited as they do not hold genuine filters inside pre-trained models, which are well-formed due to long training processes. Still, they may proceed with an inference based on redundant features. It is not the filter itself that performs poorly, but it does not match the problem to be solved.

Another class of heuristics may be described as data-based. Using data sets, it performs pruning. The recent proposition of such a method was described in [20], which estimates the importance score through an averaged gradient. However, these methods may be problematic in the case of massive datasets due to the long inference times and the extensive use of memory. Although data-based pruning utilizes the information available in the data, it may be unable to proceed with pruning properly for pre-trained architectures as this approach assumes that only filters that represent patterns absent in training data are redundant. This statement may also not hold - consider the problem of discriminating between cats and dogs: a filter representing fur is present in both cases but has no meaning in the inference process because it is not discriminative for these classes. In fact, it is redundant indeed.

3 XGB-Based Pruner

The fundamental assumption of the proposed algorithm is that the utility of a filter can be measured by its presence and utility in the decision process for the solved problem defined by a given dataset.

We also assume that the neural classifier $\mathcal{N}$ can be mimicked by a surrogate XGB classifier $\mathcal{T}$. We take the importance of features, calculated by one classifier, that reflect the nature of the problem instead of the decision process of the algorithm itself, that is, if one classifier $\mathcal{C}_1$ tends to assess one feature as important in the decision process, another classifier $\mathcal{C}_2$ should similarly process the information. Considering the exemplary problem of discriminating between triangles and circles, the "sharpness" of the edges should be an essential feature independent of the algorithm chosen to distinguish between them instead of, e.g., the background color.

3.1 Extreme Gradient Boosting

XGBoost is a tree-based ensemble machine-learning algorithm that uses gradient boosting to train decision trees. For simple classification and regression tasks, XGBoost [4] has a strong track record of producing high-quality results in various machine learning tasks, especially in Kaggle competitions, where it has been a popular choice for winning solutions.

3.2 Modus operandi

Consider an image classification task with data $\mathcal{D}$ (1) consisting of images X and respective labels Y. To solve this problem, we utilize the neural network (2) consisting of L convolutional layers in the feature extraction part $\mathcal{N}$, defined as (3), followed by the reduction layer $\mathcal{R}$ (4) and the neural classifier $\mathcal{C}$ (5). The reduction layer (4) transforms the data tensor into a vector of features and can be performed by the global max pooling layer or global average pooling. The reduced feature must assess each filter scalar real value, which makes the flattened layer unsuitable. Variable φ represents intermediate results for every layer (6) and $\varphi_0 = X$.

$$\mathcal{D} \subset X \times Y \tag{1}$$

$$\mathcal{C} \circ \mathcal{R} \circ \mathcal{N} : X \to Y \tag{2}$$

$$\mathcal{N} = \mathcal{N}(L) \circ ... \circ \mathcal{N}(1) \tag{3}$$

$$\mathcal{R} : \mathbb{R}^{w \times h \times c \times n} \to \mathbb{R}^n \tag{4}$$

$$\mathcal{C} : \mathbb{R}^n \to Y \tag{5}$$

$$\mathcal{N}(l) : \varphi_{l-1} \to \varphi_l \tag{6}$$

The algorithm works as follows: for every layer $l \in [1, L]$ in the feature extractor $\mathcal{N}$ part, we create a submodel $\mathcal{S}(l)$ as described by (7).

$$\mathcal{S}(l) = \mathcal{T} \circ \mathcal{R} \circ \mathcal{N}(l) \circ ... \circ \mathcal{N}(1) \tag{7}$$

Each of the n filters in the final convolutional layer results in a scalar feature. These features are then used as training data in the XGB classifier $\mathcal{T}$. The trained surrogate classifier $\mathcal{T}$ can assess the importance $\mathcal{I}$ of each feature, which is calculated for every filter n after (8), i.e., the improvement in precision brought about by a feature to model averaged over all trees. This refers to the default method of estimating feature importances in XGBoost named "Gain".

$$\forall j \in (1, n)\ \mathcal{I}(\varphi(j)|\mathcal{T}) = Gain(\varphi(j)) \tag{8}$$

The pruning predicate P with the threshold ρ is defined as (9). The proposed method works structurally as the pruning predicate assesses results per filter. Therefore, it can efficiently reduce the size of the output model.

$$P(\ i \mid \rho, \mathcal{T}) = \mathcal{I}(\varphi(i)|\mathcal{T}) \leq \rho \tag{9}$$

The pseudocode for the proposed pruning algorithm can be found in Algorithm 1. L refers to the number of convolutional layers in the reference model $\mathcal{N}$.

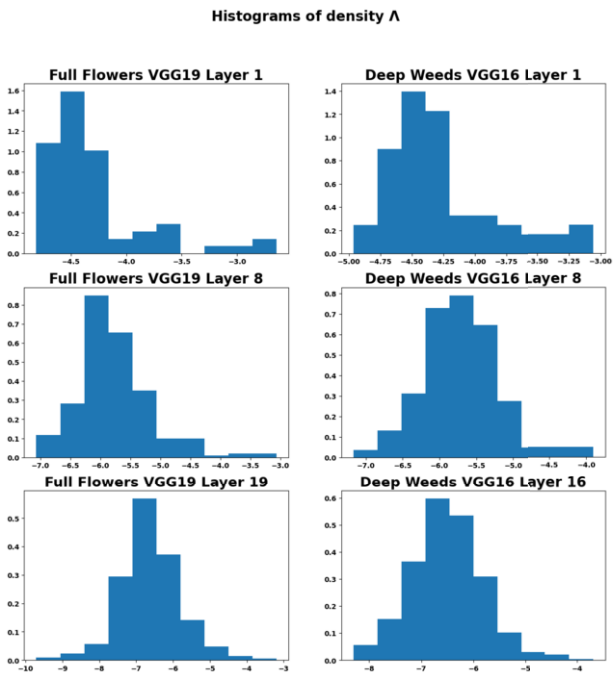

Figure 1. Distribuiton of Λ across chosen architectures (VGG16 and VGG19) and datasets (Flowers and Deep Weeds).

3.3 *Feature importance analysis*

Figure 1 shows the distribution of Λ defined as (10), that is, the natural logarithm (ln) of the importance of features normalized with $\epsilon > 0$ used to ensure a positive argument of the logarithm. Different layers are displayed for VGG19 on the Flowers dataset and VGG16 on the Deep Weeds [21] dataset. In deeper layers, the distribution of Λ seems to follow a normal distribution. For these plots, ϵ was chosen as 1e-16. The x-axis scales differ in each plot because the number of convolutional filters varies across the layers.

$$\Lambda(i) = ln(\mathcal{I}(\varphi(i)|\mathcal{T}) + \epsilon) \tag{10}$$

Algorithm 1 XGB Based Pruner

Parameters:
XGB parameters Q
threshold ρ
performance measure μ
algorithm mode MODE
Inputs:
training data $\mathcal{D}$
reference ConvNet backbone $\mathcal{N}$
Outputs:
pruned model $\mathcal{N}^*$
performance indices $\{\mu(i) : i \in [1; L]\}$

X = images
Y = data labels
for $i = 1$ **to** L **do**
  $\varphi \leftarrow \mathcal{N}(i)(X)$
  Create surrogate model $\mathcal{T}(i)$ given parameters Q
  Train model $\mathcal{T}(i)$ using $\mathcal{R}(\varphi_{train})$ and Y_{train}
  Find all indices satisfying the pruning predicate (9)
  $J(i) = \{j : P(\, j \mid \rho, \mathcal{T}(Q))\}$
  $\mu(i) = \mu(\mathcal{T}(i) \mid \mathcal{R}(\varphi_{val}), Y_{val})$
  if MODE is LAYER-WISE **then**
    Proceed pruning of layer i and φ according to $J(i)$
  end if
  $X \leftarrow \varphi$
end for
if MODE is ONE-SHOT **then**
  Prune the entire model according to J
end if

3.4 *Hyperparameter selection*

The algorithm parameters cover threshold ρ for the pruning predicate, mode, and parameters for the XGB surrogate model, such as the number of tree limits, the tree max depth, the function to measure the feature importance, and the loss function. XGB-based Pruner works quite robustly regarding parameter selection of the XGB surrogate. However, challenging computer vision tasks may require parameter adjustment. The authors find adjusting the number of trees and tree max depth to be the most promising, where prior knowledge: the harder the problem, the deeper trees can be utilized. Threshold ρ can be adjusted by analyzing Λ histograms.

3.5 *Mode selection*

The proposed algorithm can work in different modes:

- One-Shot: Feature importance assessment is performed for every layer, and then pruning is done for every layer. In this context, one-shot refers to processing all layers simultaneously, not sequentially.
- Layer-Wise: Feature importance assessment followed by pruning is done for each layer sequentially: layer l is assessed, then pruned, after that layer $l + 1$ is assessed and pruned, etc.

The layer-wise mode should yield more aggressive pruning, as it assumes the same data flow during inference as in the pruned model. On the other hand, the One-Shot model can proceed with pruning based on information that may no longer be available in the inference of a model (as it may have been pruned). However, the experimental results presented in the following sections have shown that the One-Shot mode may yield models that are larger but perform better because of relatively mild pruning.

3.6 *Optimization target*

The proposed method can be used to reduce size and improve performance. Different modes usually yield a different model. This paper considers two targets: model size, that is, several parameters in a model that we aim to minimize and simultaneously maximize model performance, measured as accuracy over test datasets.

The proposed solution optimization target is defined utilizing Pareto optimality, i.e., getting a better solution measured in one criterion is impossible without worsening the second one. We aim to dominate the reference backbone solution (Transfer learning model before pruning), i.e., exclude the reference model from Pareto Frontier. We strive to achieve comparable performance for the pruned models as for the unpruned models, minimizing the model size and the number of parameters.

4 Accelerated depth pruning

4.1 Motivation

For multiple SOTA architectures, the number of parameters in a layer is expected to grow as one moves from input to output of a ConvNet backbone. The comparison of the cumulative weight density for the VGG19, ResNet50 [13], ResNet101v2 [14], and InceptionV2 [25] backbones is shown in Figure 2. The steep incline of the posts on their right part indicates that most of the model weights are located in deeper layers of the models. None of the models presented at half their depth has more than 36% of their weights.

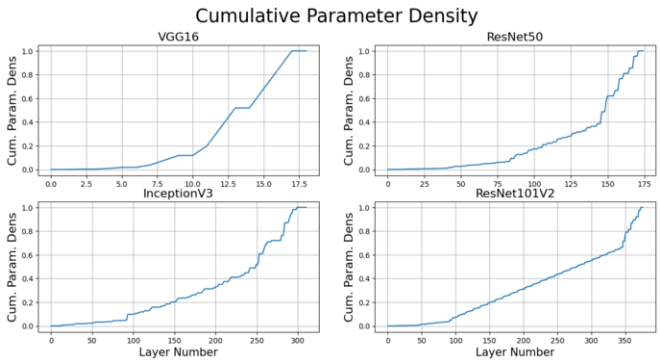

Figure 2. Comparison of cumulative parameter densities for the chosen ConvNet backbones. Most weights in SOTA architectures are located at the end of the convolutional backbone.

[19] has shown via the activation maximization method that the first layers depict simple patterns, while the deeper convolutional layers depict more abstract features. We conclude that for some problems, there is no need for such complex features to be utilized and that a more straightforward representation is sufficient for solving a task correctly. This allows us to reduce the size of the model significantly by pruning entire deeper layers where the pruning is more efficient than in other layers as they contain the major part of all trainable parameters.

4.2 Heuristic

Each sub-model is assessed by the performance measure $\mu(l)$ defined as (11). Sub-model $\mathcal{S}(l)$ is assessed over data $\mathcal{D}$. This paper uses the accuracy score as a reference performance score μ.

$$\mu(l) = \mu(\mathcal{S}(l) \mid \mathcal{D}) \tag{11}$$

Depending on the depth heuristics $\mathcal{H}_D$, one may expect μ to satisfy certain constraints. Here we assume that normalized measure μ^*, defined as (12), satisfies (13), i.e., the normalized performance of the XGB model $\mathcal{T}$ is comparable to the normalized performance achieved by the neural classifier.

$$\mu^*(l) = \frac{\mu(l)}{\max_{i \in [1;L]} \mu(i)} \tag{12}$$

$$\mu^*(l \mid \mathcal{C}) = \mu^*(l \mid \mathcal{T}) \tag{13}$$

4.3 Analyses of Experiments

The results of the validation of (13) are shown in Figure 3, where the neural classifier has been trained on the same backbone sub-models as the surrogate XGB classifier. The reference model was chosen as VGG19 [24] in two classes (roses and tulips) of the TensorFlow Flowers Dataset [26] and the VGG16 model [24] over the full TensorFlow Flowers Dataset. Both models were pre-trained on the ImageNet dataset [6]. Values of $\mu^*(l)$ are close to each other, and the general trend is preserved for both classifiers.

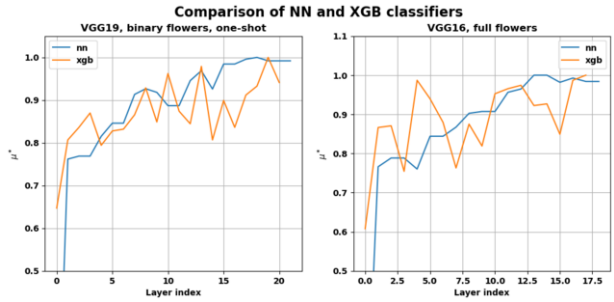

Figure 3. Comparison of μ^* for experimental data for the commonly used VGG19 transfer learning architectures.

The equality (13) does not always hold, and it is not trivial to define the conditions under which it has. However, as shown in the following sections, the proposed constraint does not need to hold to work sufficiently well. It is sufficient that the values of the neural classifier and its surrogate are approximately the same.

Heuristics $\mathcal{H}_D$ to estimate the optimal depth of the backbone is defined as (14). Such an approach focuses only on the performance of the model.

$$\mathcal{H}_D : l^* = \arg \max_{l \in (1,L)} \mu^*(l) \tag{14}$$

As the XGB-based Pruner algorithm defined in Section 1 already gives a set of performance indices mapping $\{\mu(i) : i \in [1; L]\}$, it is not necessary to rerun the calculations. This is a significant improvement over existing methods for depth pruning of ConvNets.

Depth pruning cannot result in a larger model than pruning in a depth-preserving manner. This makes depth pruning the go-to solution for applications where the model size is critical for optimization.

5 Results of Experiments

5.1 Experimental setup

The general training scheme for XGB-based Pruner has been described in Algorithm 2. The accuracy of a model refers to the maximum value of accuracy in the test dataset achieved during the training process. The entire source code, experimental setup details and default parameters, together with the README materials, have been attached to the paper in the supplementary materials [23] for complete reproducibility.

5.2 Tests for common architectures

Experiments have been conducted for Flowers [26], Cats vs Dogs [9], and eye diseases classification datasets: Diabetic Retinopathy binary classification (DR Eye) and Eye diseases dataset for three different diseases (Diabetic Retinopathy, Cataract, Glaucoma) and healthy reference. The results of the numerical experiments are covered in Table 1. However, the proposed algorithm allows the VGG architectures to achieve similar or better results than ResNet50. For majority of the datasets, the proposed algorithm removed up to 50% of all weights while maintaining nearly the same level of accuracy. For Diabetic Retinopathy dataset pruned model with 27% of reference model parameters gained 2% better accuracy on the test dataset.

Algorithm 2 Training Scheme

Parameters:
Pruner parameters $Params$
Flag indicating depth pruning DEPTH-PRUNING
Number of epochs for freeze training e_f
Number of epochs for unfreezing training e_u
Inputs:
training data $\mathcal{D}$
reference ConvNet backbone $\mathcal{N}$
Outputs:
finetuned model $\mathcal{C} \circ \mathcal{R} \circ \mathcal{N}^*$

$l \leftarrow L$
get $\mathcal{N}^*$ through XGB-based Pruner according to $Params$
if DEPTH-PRUNING **then**
  infer optimal depth l^* according to (14)
  remove layers after l^*th layer from $\mathcal{N}^*$ model
  $l \leftarrow l^*$
end if
freeze entire convolutional backbone
add reduction layer $\mathcal{R}$ and neural classifier $\mathcal{C}$
train model $\mathcal{C} \circ \mathcal{R} \circ \mathcal{N}^*$ for e_f epochs
for $i = 1$ **to** l **do**
  train model $\mathcal{C} \circ \mathcal{R} \circ \mathcal{N}^*$ for e_u epochs
  unfreeze the last frozen convolutional layer
end for

Table 1. Comparisons of the model sizes and performances μ across different architectures and datasets with the reference models.

Dataset		VGG16	VGG19	ResNet50
Binary Flowers				
Reference	size	1.5e7	2e7	2.3e7
	μ	82%	84%	91%
Proposed	size	**6.6e6**	**9.9e6**	**1.2e7**
	μ	**93%**	**95%**	**92%**
Flowers				
Reference	size	1.5e7	2e7	2.3e7
	μ	75%	67%	90%
Proposed	size	**1.2e7**	**1.6e7**	**2.2e7**
	μ	**92%**	**91%**	**91%**
Cats vs dogs				
Reference	size	1.5e7	2e7	2.3e7
	μ	74%	81%	98%
Proposed	size	**7.4e6**	**1.1e7**	**7.9e6**
	μ	**97%**	**97%**	96%
DR Eyes				
Reference	size	1.5e7	2e7	2.3e7
	μ	95%	95%	96%
Proposed	size	**4.1e6**	**9.3e6**	**8.2e6**
	μ	**97%**	**98%**	95%
Eye diseases				
Reference	size	1.5e7	2e7	2.3e7
	μ	87%	88%	90%
Proposed	size	**7.9e6**	**1e7**	**1.2e7**
	μ	**93%**	**92%**	90%

5.3 Mode selection

The proposed pruning algorithm has been applied to multiple reference backbones and datasets. Figure 4 presents one model (one shot depth) that spans the Pareto Frontier for the VGG16 model and the Eye diseases dataset. Figure 5 shows three models spanning the Pareto Frontier for the VGG19 model for the same dataset - depending on the objective, one can select a model with half of the parameter (stepwise depth mode), one with +4% accuracy score (stepwise) or intermediate solution (one shot depth).

Notably, for both experiments, every proposed model excludes the reference backbone model from Pareto Frontier; the proposed solution is always better than the reference model, independent of the mode choice. Usually, the proposed depth pruning approach yields the best results.

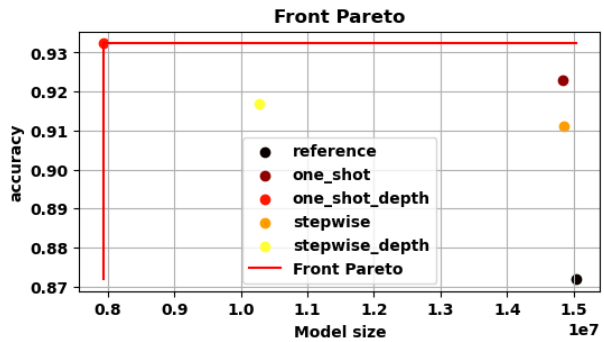

Figure 4. Comparison of the different modes of the algorithm for the VGG16 model over the Eye diseases dataset.

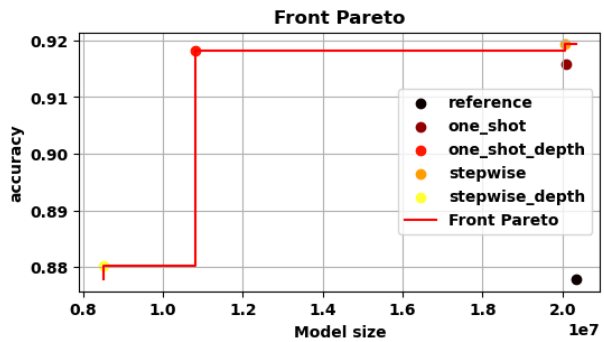

Figure 5. Comparison of the different modes of the proposed XGB-based Pruner for the VGG19 model over the Eye diseases dataset.

5.4 Model size and problem complexity

The definition of the complexity γ of computer vision tasks can be challenging to express formally. Therefore, we assume that complexity is monotonous in the mean of inclusion; that is, if for two sets of classes (and thus their instances) A and B, the formula $A \subset B$ is satisfied. The complexity of a sub-problem of discriminating between sub-classes A is lower than or equal to the complexity of distinguishing between sub-classes B. This is formally stated by (15).

$$\gamma : A \subset B \implies \gamma(A) \leq \gamma(B) \tag{15}$$

Given the constraint for relative complexity (15), we expect an easier problem to require a smaller model than a more complex problem.

Table 1 shows the model sizes measured as the number of parameters for the Flowers Dataset [26] and the Binary Flowers, i.e., the images that depict Tulips and Roses from the same dataset as well as for the DR Eyes and Eye diseases datasets. All models achieve accuracies greater than 90%. For simpler sub-problems, model sizes are reduced compared to the base problems for all common architectures.

5.5 Parameter choice

Figure 6 shows the threshold ρ's impact on the output model size and accuracy. The reference model for this experiment is VGG16, and the DR Eyes dataset. All the pruning was performed in a depth pruning manner, too. The higher the threshold ρ, the smaller the output pruned model, where the differences in the resulting models are negligible for low thresholds. The reference VGG16 model achieves 95% accuracy and 14,714,688 parameters. The most miniature architecture possible reached 71,165 total parameters (which is less than 0.5% of the total parameter count of the reference model) and a validation set accuracy of 76%. At the same time, the best-performing one has 6,934,695 real parameters (47% of the total parameter count of the reference model) and 97 % accuracy on the validation dataset.

Figure 7 shows the threshold ρ's impact on the output model size and accuracy. The reference model for this experiment is VGG19 and the Cats vs. Dogs dataset. All the pruning was performed in a depth pruning manner, too. The higher the threshold ρ, the smaller the output pruned model, where the differences in the resulting models are negligible for lower thresholds. For higher thresholds, lack of subtlety in the pruning process, i.e., inference with the wrong assumption of data availability, results in a model just above the entirely random classifier. Lower thresholds result in models with comparable accuracy.

5.6 Discussion

The results may vary due to the threshold ρ choice. Although ρ may be poorly chosen, the proposed pruning algorithm still delivers Pareto-optimal solutions due to the significant influence of depth pruning. The results of the chosen architectures across different datasets show that the proposed method can perform pruning efficiently. It may be surprising that transfer learning architectures can achieve much better results with proper pruning. Such gains are possible as even properly formed filters across model layers may introduce interference because they do not represent any feature crucial for a decision process. Such features introduce noise to an inference process and reduce the efficiency of a model.

6 Comparisons

6.1 Performance comparison

For comparisons with the proposed method, we have chosen the popular Magnitude-based Pruner, i.e., unstructured pruning, which sets all weights with a magnitude lower than the given threshold to 0. We use it due to its popularity in frameworks like PyTorch [22]. We focused on comparing the ability of interference reduction between

Figure 6. Comparison of different thresholds ρ for layer-wise mode on VGG16 model for DR Eyes dataset.

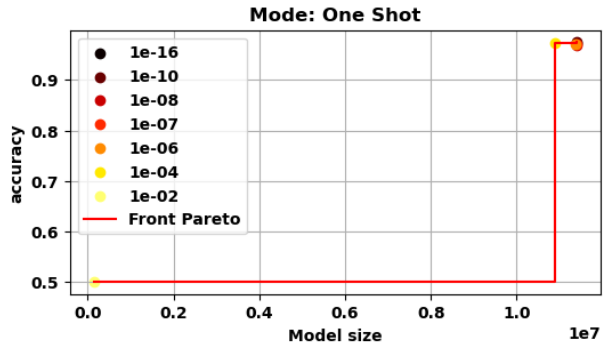

Figure 7. Comparison of the different ρ thresholds for the one-shot mode on the VGG19 model for the Cats vs Dogs dataset.

these two pruning methods, as the second one, due to its unstructured approach, is not restricted to removing entire filters, just particular weights. Comparisons were conducted for multiple pruning thresholds of the Magnitude-based Pruner. The comparison for the VGG16 model and Binary Flowers dataset is depicted in Figure 8. Every mode of the proposed method is better in size and accuracy than even the best Magnitude-based Pruner. The same holds for the comparisons on the entire Flowers dataset shown in Figure 9.

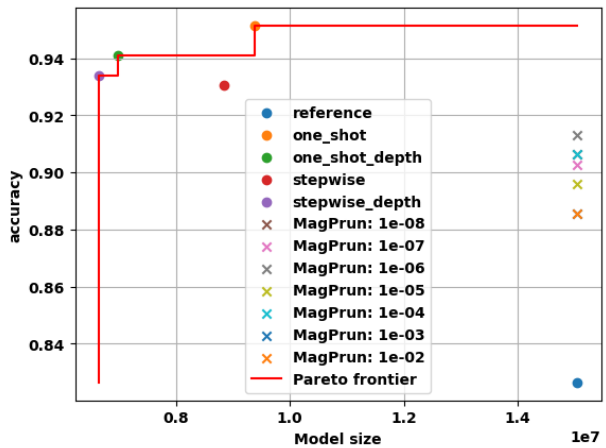

Figure 8. Comparison of the proposed XGB-based Pruner and the Magnitude-based Pruner over the Binary Flowers dataset with the VGG16 as a reference architecture.

For the adequately chosen parameters and modes, the resulting architecture of the XGB-based Pruner is much smaller than the one produced by Magnitude-based Pruner. Moreover, the proposed algorithm for all modes performs better than the Magnitude-based Pruner, as presented in Figures 10 and 11. XGB-based Pruner eliminates filters, i.e., reduces interference in the model more efficiently than Magnitude-based Pruner eliminates weights. This may be considered as a step toward solving the Lottery Ticket Hypothesis problem.

6.2 Speed comparison

For speed comparison solution, we utilize the depth pruning proposed by De Leon and Atienza [5]. For tests, XGB has been selected. For comparison, neural networks were trained for 10 epochs, which is usually not sufficient to achieve convergence. As the literature lacks theoretical analysis of the time complexity of neural networks and XGB training, the explicit time is compared. Table 2 shows the comparison of the training speed for multiple datasets.

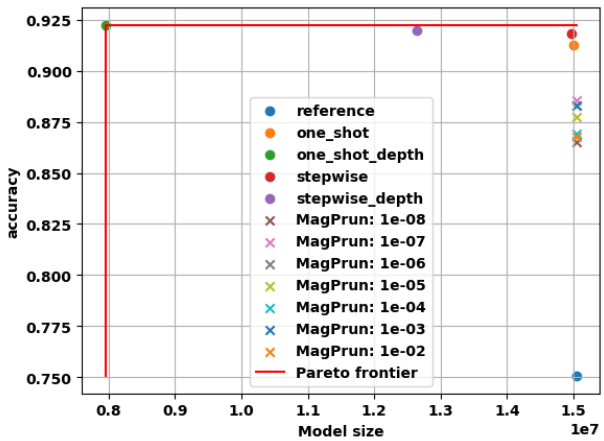

Figure 9. Comparison of the proposed XGB-based Pruner and the Magnitude-based Pruner over the Flowers dataset with the VGG16 as a reference architecture.

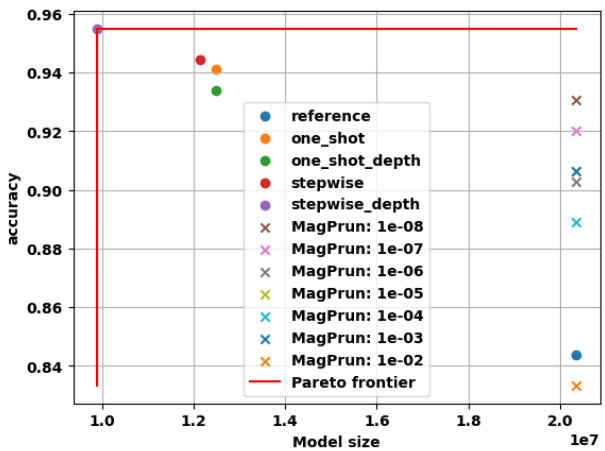

Figure 10. Comparison of the proposed XGB-based Pruner and the Magnitude-based Pruner over the Binary Flowers dataset with the VGG19 as a reference architecture.

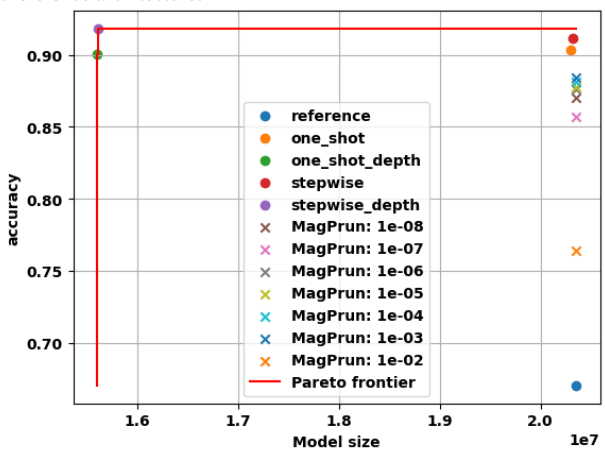

Figure 11. Comparison of the proposed XGB-based Pruner and the Magnitude-based Pruner over the Flowers dataset with the VGG19 as a reference architecture.

Gradient boosted trees are much faster to train, although neural surrogate was trained only for 10 epochs. For neural surrogates, the JIT compilation takes up most of the time.

Table 2. Speed comparison of the trained XGB and the neural surrogates for the VGG16 architecture.

Dataset	Train samples	Time NN[s]	Time XGB[s]	Time
Binary Flowers	1152	465	73	-84%
Flowers	2936	1117	521	-53%
Deep Weeds	2431	1139	646	-43%
Cats vs Dogs	2000	770	114	-85%
DR Eyes	2076	776	64	-91%
Eye Diseases	3374	1256	471	-62%

7 Limitations

The proposed algorithm has been introduced for pruning convolutional backbones used in classification tasks 1 as they are the most popular tasks in computer vision based on transfer-learning architectures. This is because most ConvNet parameters are located in the convolutional backbones.

The XGBoost algorithm also supports regression tasks, so the XGB pruner may be applied to such problems, as well. However, regression tasks in computer vision are not very common, so we do not provide such experiments in this paper. Although the proposed algorithm utilizes XGBoost as a surrogate classifier, the proposed approach allows for the use of other surrogates, too.

The proposed algorithm especially aims to reduce the complexity of transfer-learning architectures to improve their efficiency. XGB Pruner aims to remove filters that do not represent meaningful features in certain datasets. Therefore, it is not expected to introduce significantly better results for datasets comparable to datasets like ImageNet [6] or COCO [18], which are used for training these transfer-learning architectures.

Further work is required for the adaptation of the proposed methods for detection tasks (composition of classification and bounding box regression) and segmentation tasks (per-pixel classification).

8 Conclusions

We proposed the XGB-based Pruner, which can efficiently address the problem of pruning transfer-learning architectures. Moreover, the XGB-based Pruner approach allows the estimation of optimal depth for depth pruning simultaneously proceeding with width pruning, which significantly enlarges the pruning ratio even with poorly chosen parameters for width pruning. In detail, we have:

- proposed a fast pruning method that solves the pruning problem for the different transfer learning models;
- analyzed the depth pruning mode and showed that it can be performed with no additional computational overhead;
- shown that the proposed pruning method is pretty robust and works well for different model architectures and computer vision datasets;
- explored the proposed method experimentally and showed its superior results to other common pruning methods.

Finally, we have demonstrated that we can reduce the model size by nearly 50% while increasing the accuracy on the test dataset by 11% or reduce the model size by nearly 75%, increasing the accuracy on the test dataset by 2%. Alternatively, we can reduce the model size by more than two hundred times, slightly reducing the model accuracy. The selection of the final reduction rate depends on the application and the computing resources of the system.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] P. Angeline, G. Saunders, and J. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65, 1994. doi: 10.1109/72.265960.

[3] D. W. Blalock, J. J. G. Ortiz, J. Frankle, and J. V. Guttag. What is the state of neural network pruning? In I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, editors, *MLSys*. mlsys.org, 2020. URL http://dblp.uni-trier.de/db/conf/mlsys/mlsys2020.html#BlalockOFG20.

[4] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

[5] J. D. De Leon and R. Atienza. Depth pruning with auxiliary networks for tinyml. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3963–3967, 2022. doi: 10.1109/ICASSP43922.2022.9746843.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] M. Denil, B. Shakibi, L. Dinh, M. A. Ranzato, and N. de Freitas. Predicting parameters in deep learning. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/7fec306d1e665bc9c748b5d2b99a6e97-Paper.pdf.

[8] X. Ding, G. Ding, J. Han, and S. Tang. Auto-balanced filter pruning for efficient convolutional neural networks. In *AAAI*, Palo Alto, California, USA, 2018. AAAI Press, AAAI Press. URL https://ojs.aaai.org/index.php/AAAI/article/view/11709.

[9] J. Elson, J. J. Douceur, J. Howell, and J. Saul. Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, Inc., October 2007. URL https://www.microsoft.com/en-us/research/publication/asirra-a-captcha-that-exploits-interest-aligned-manual-image-categorization/.

[10] J. Frankle and M. Carbin. The lottery ticket hypothesis: Training pruned neural networks. 03 2018.

[11] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[12] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

[15] G. Jocher. Ultralytics yolov5, 2020. URL https://github.com/ultralytics/yolov5.

[16] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics yolov8, 2023. URL https://github.com/ultralytics/ultralytics.

[17] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710, 2016.

[18] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[19] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120, 12 2016. doi: 10.1007/s11263-016-0911-8.

[20] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning, 2019.

[21] A. Olsen, D. A. Konovalov, B. Philippa, P. Ridd, J. C. Wood, J. Johns, W. Banks, B. Girgenti, O. Kenny, J. Whinney, B. Calvert, M. Rahimi Azghadi, and R. D. White. DeepWeeds: A Multiclass Weed Species Image Dataset for Deep Learning. *Scientific Reports*, 9(2058), 2 2019. doi: 10.1038/s41598-018-38343-3. URL https://doi.org/10.1038/s41598-018-38343-3.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[23] I. Ratajczyk. Xgb-based-pruner, Aug. 2024. URL https://doi.org/10.5281/zenodo.13340567.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

[26] T. T. Team. Flowers, jan 2019. URL http://download.tensorflow.org/example_images/flower_photos.tgz.

[27] B. Zoph and Q. Le. Neural architecture search with reinforcement learning. 11 2016.