AGH

RESEARCH UNIVERSITY
EXCELLENCE INITIATIVE

AGH UNIVERSITY OF KRAKOW

ECAI
50th ANNIVERSARY

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

#M1096 – ECAI 2024 conference paper

Igor Ratajczyk igor@ratajczyk.eu (Master's Student)

Adrian Horzyk horzyk@agh.edu.pl

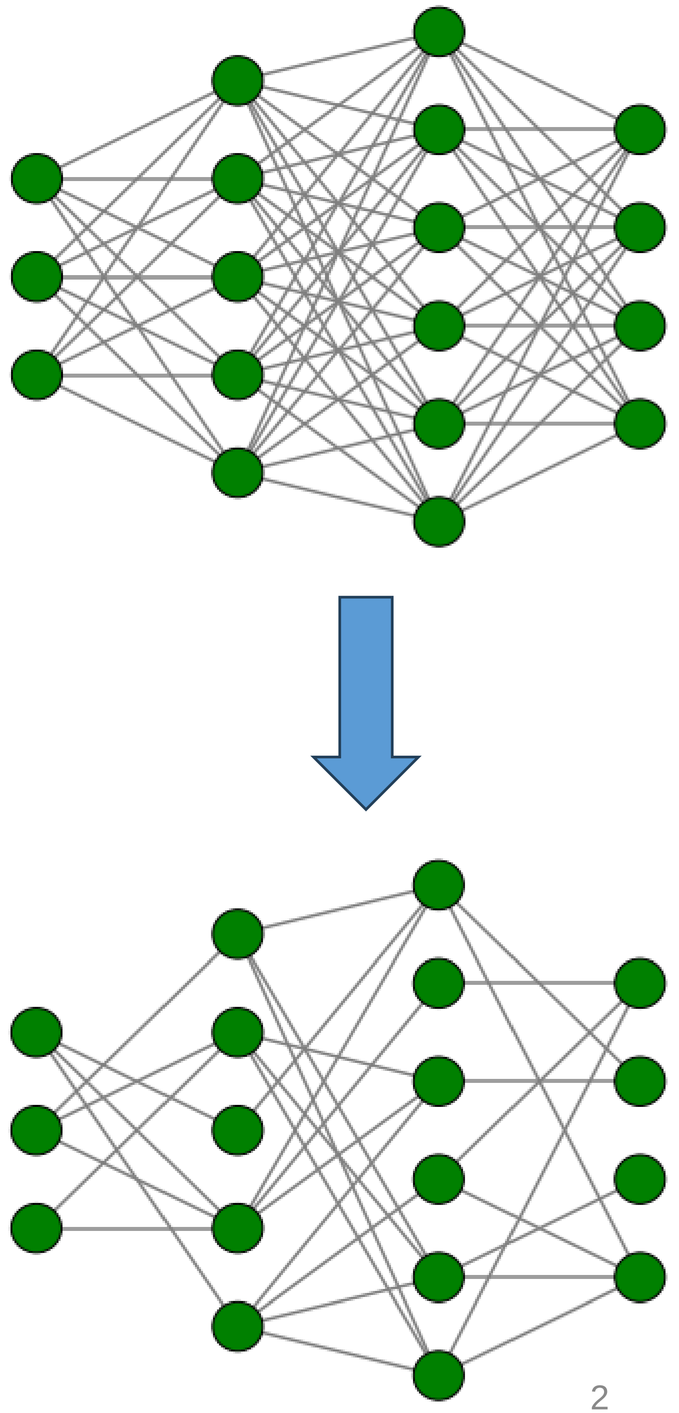
agh.edu.pl

Pruning

Pruning algorithms aim to reduce
the size of the model, the used memory,
and computational complexity
by removing unnecessary parameters.

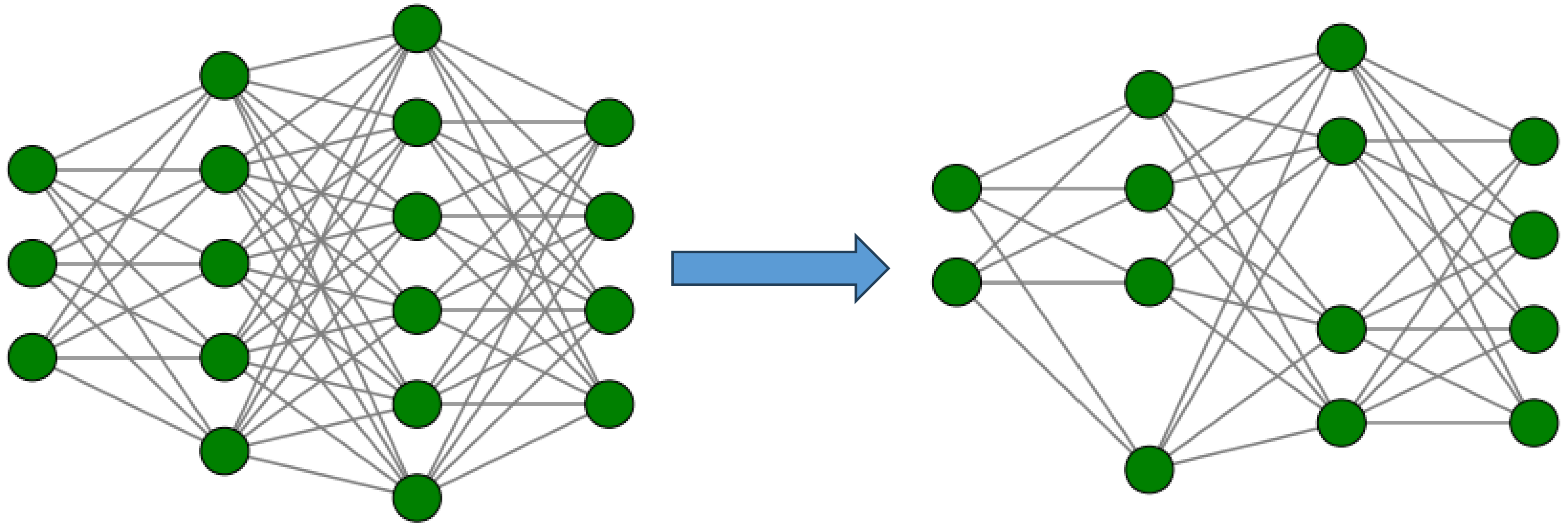
Pruning methods are usually divided into:

- Unstructured pruning
- Structured pruning

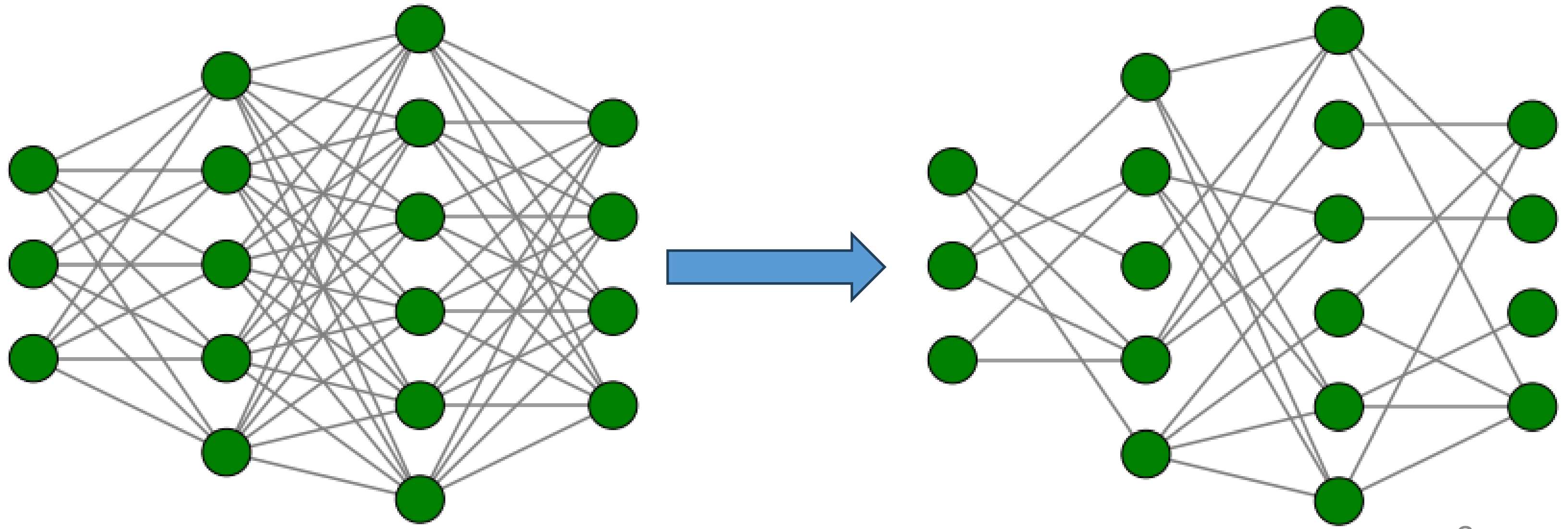


Pruning

Structured Pruning



Unstructured Pruning

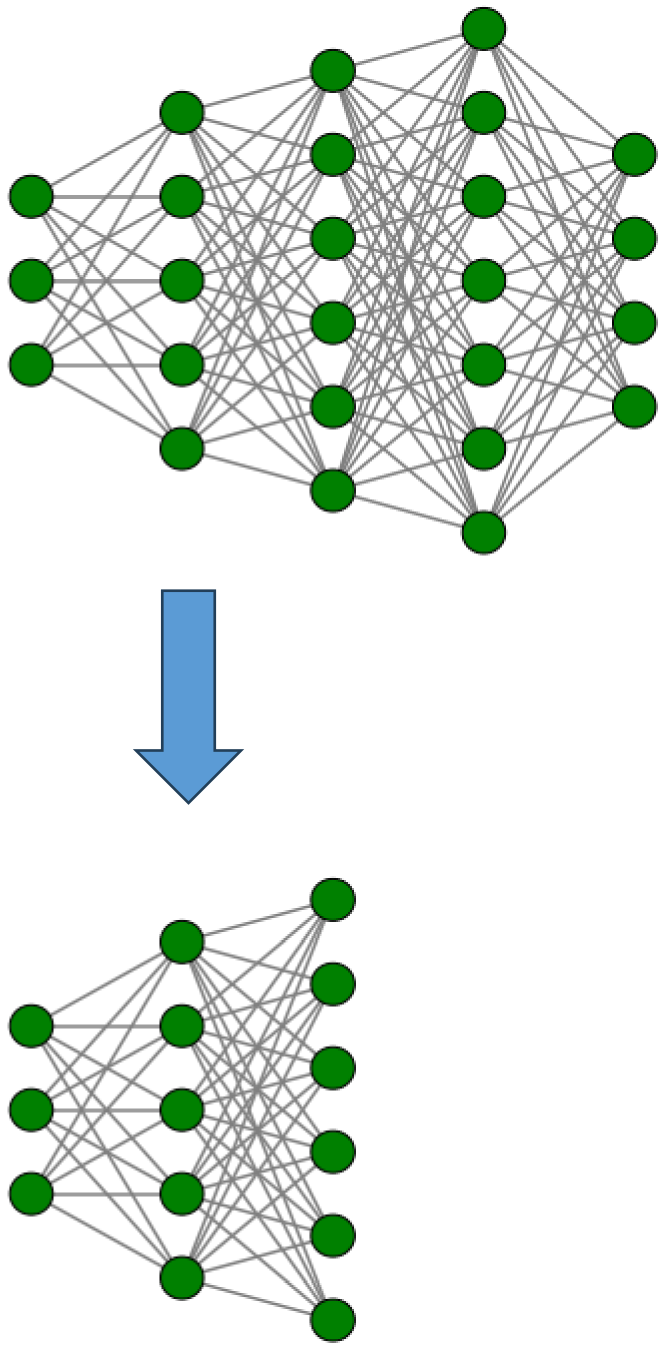


Depth pruning

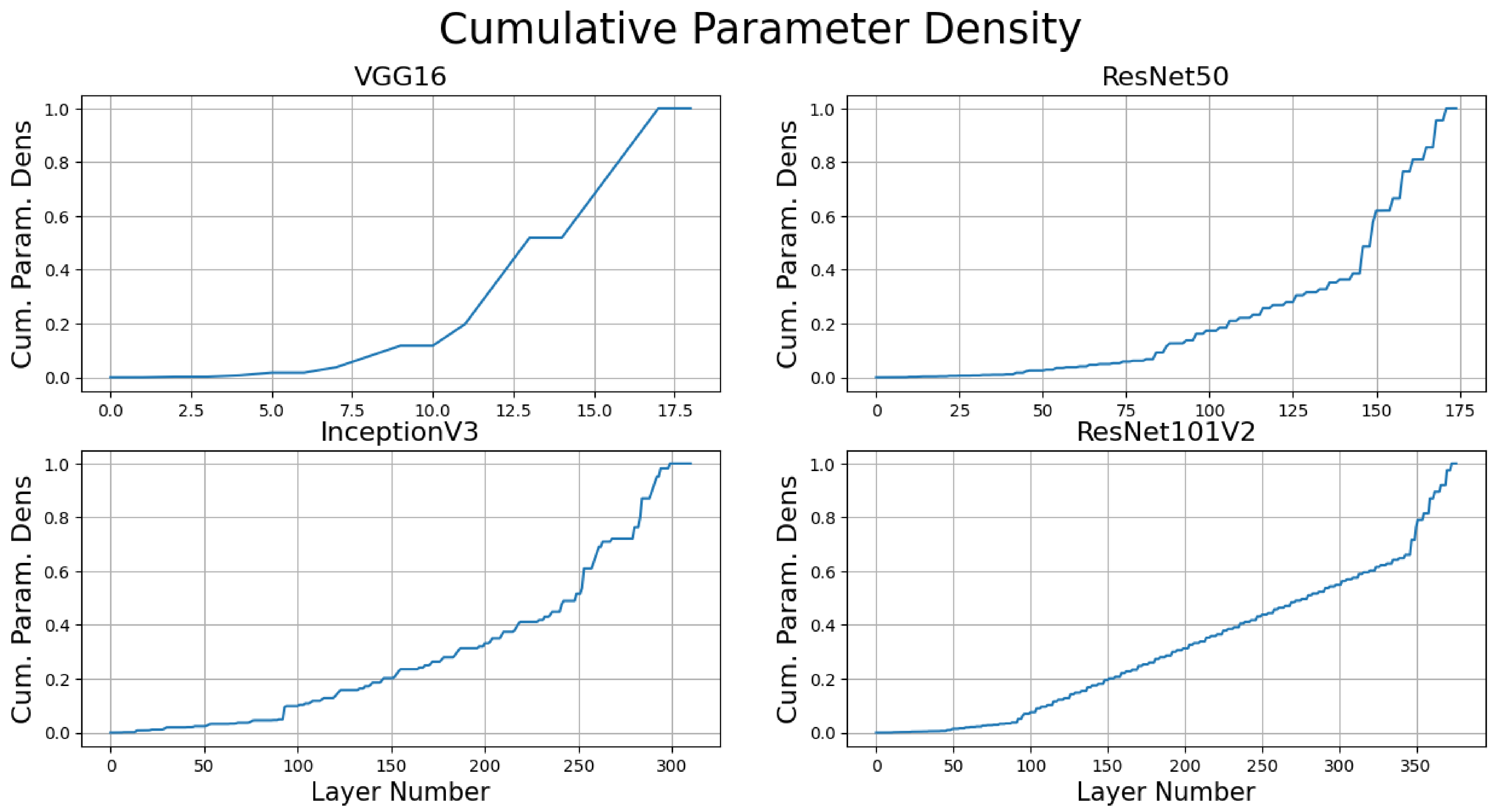
[De Leon, Atienza] proposed a novel pruning method – **depth pruning**, which removes entire layers in ConvNet.

Such an approach is computationally demanding due to the necessity of training multiple neural classifiers.

Moreover, width pruning cannot be performed in parallel in this approach.



Depth pruning – why should we care?



Pruning transfer learning architectures

In transfer learning architectures, **filters** are well formed and represent meaningful patterns, but these **patterns** may just be useless in terms of a particular dataset, i.e., filter detecting fur may not be useful in discrimination between cats and dogs because both have furs.

[Mahendran et al.] have shown, that the deeper the convolutional backbone, the **more abstract features** it creates. **Depth pruning** may be applied to significantly reduce the model size, as simple problems may not need such advanced features, i.e.: to determine whether tomato is ripe or not one needs only color features.

Addressed issues

In this paper, we address the following issues:

- **Difficulty in Simultaneous Depth and Width Pruning** which can be combined and used in parallel**.**
- **Lack of efficient pruning for transfer learning of convnet architectures** because most existing methods focus on **fine-tuning** or **pruning width (filters) but not depth (layers)** or **fail to handle both pruning tasks simultaneously**.
- **Removing interference noise from the inference process** due to over-parameterization of models that carry unnecessary convolutional filters, which leads to generalization improvement of the pruned models.

Contributions

In this paper, we address the following issues:

- **Novel XGB-based Pruning Algorithm** - a new pruning method that uses XGBoost as a surrogate classifier to prune transfer learning models efficiently.
- **Pruning for Transfer Learning Models** addresses the problem of over-parameterization in transfer learning models by pruning unnecessary convolutional filters and reducing the model size without compromising performance and, in some cases, even improves accuracy.
- **XGBoost to evaluate feature importance** in convolutional layers is a novel application, enabling the identification of filters that are less relevant to the solved task.
- **Efficient Model Size Reduction** (up to 50% or even more than 100 times) while maintaining or even improving accuracy.

eXtreme Gradient Boost

XGBoost is a tree-based ensemble machine-learning algorithm that uses **gradient boosting** to train **decision trees**.

I our case, **XGBoost** is used to guide both depth pruning (removing entire layers) and width pruning (removing unnecessary filters within a layer). This process is done by creating a sub-model for each convolutional layer and then using XGBoost to determine the importance of features derived from each filter. Filters that contribute little to the overall decision-making process, based on the XGBoost feature importance scores, are pruned.

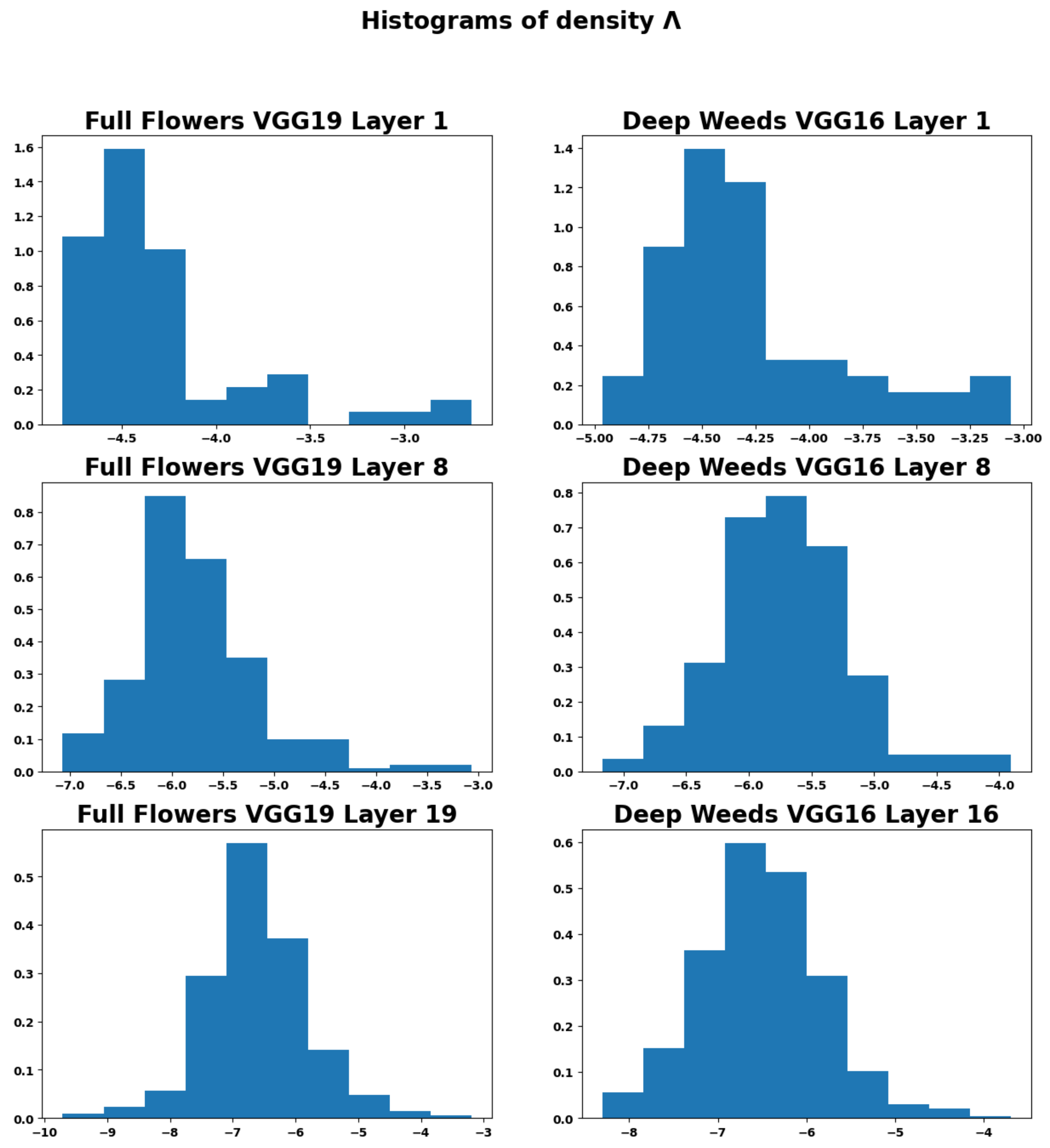
For simple classification and regression tasks, **XGBoost** has a strong track record of producing high-quality results in various machine learning tasks.

XGBoost

Trained XGBoost model can assess the importance of particular features, i.e., measure how much of performance is gained with j-th feature.

$$\mathcal{J}(\varphi(j)|\mathcal{T}) = Gain(\varphi(j))$$

$$\Lambda = \log(\mathcal{J}(\varphi|\mathcal{T}) + \epsilon)$$



XGB-based pruner – big picture

PROPOSED PRUNING ALGORITHM:

1. For each layer i, we build an **XGB classifier $\boldsymbol{\mathcal{T}(i)}$** on top of features generated by the neural network.
2. The classifier then assesses **the importances $\boldsymbol{\mathcal{I}}$ of every feature $\boldsymbol{\varphi(j)}$**.
3. Every j-th feature is associated with j-th filter.
4. Those features with an importance lower than a given threshold ρ are removed.

XGB-based pruner – big picture

Let us consider a classification task with images X and labels Y. CNN consists of convolutional backbone $\mathcal{N}$, reduction layer $\mathcal{R}$, and dense neural classifier $\mathcal{C}$:

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

where

$$\mathcal{N} = \mathcal{N}(L) \circ \cdots \circ \mathcal{N}(1)$$

Reduction layer $\mathcal{R}$ can be implemented as GlobalMaxPool or GlobalMeanPool.

It may be also Flatten layer, but we do not recommend it.

XGB-based pruner – modus operandi

Algorithm: XGB Based Pruner

Parameters:

- XGB parameters Q
- Threshold ρ
- Performance measure μ
- Algorithm mode MODE

Inputs:

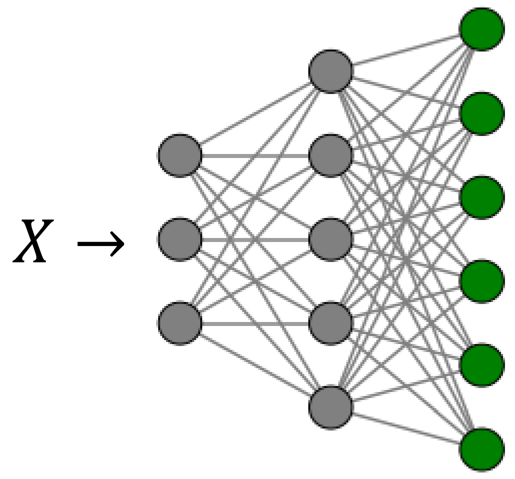
- Training data D
- Reference ConvNet backbone $\mathcal{N}$

Outputs:

- Pruned model $\mathcal{N}^*$
- Performance indices $\{\mu(i): i \in [1; L]\}$

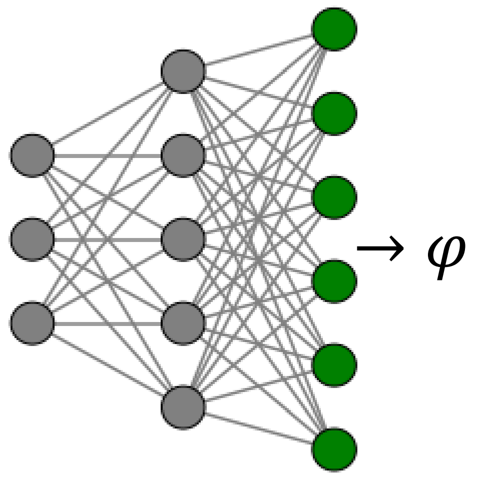
```
• Let $X$ = images
• Let $Y$ = data labels
• For i=1 to $L$:
    • Set $\varphi \leftarrow \mathcal{N}(i)(X)$
    • Create surrogate model $\mathcal{T}(i)$ given parameters $Q$
    • Train model $\mathcal{T}(i)$ using $\mathcal{R}(\varphi_{train})$ and $Y_{train}$
    • Find all indices satisfying the pruning predicate
    • Set $J(i) = \{j: P(\ j \mid \rho, \mathcal{T}(Q)\ )\}$
    • Set $\mu(i) = \mu(\mathcal{T}(i) \mid R(\varphi), Y)$
    If MODE is LAYER-WISE:
        ▪ Proceed pruning of layer $i$ and $\varphi$ according to $J(i)$
        ▪ Set $X \leftarrow \varphi$
    If MODE is ONE-SHOT:
        ▪ Prune the entire model according to J
```

Inference



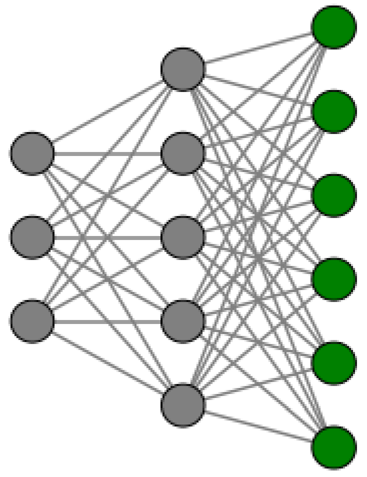
Images are processed by CNN’s convolutional backbone.

Feature Extraction



Features are obtained.

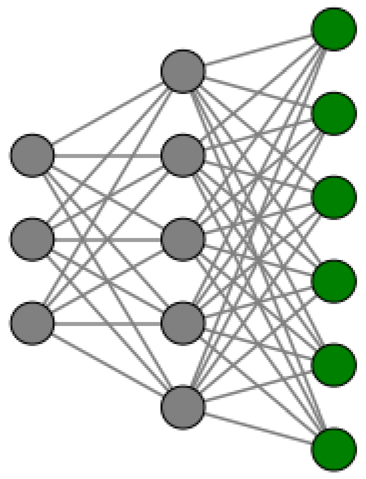
Reduction



$$\varphi \to \mathcal{R}(\varphi)$$

Feature tensor is reduced to a vector.

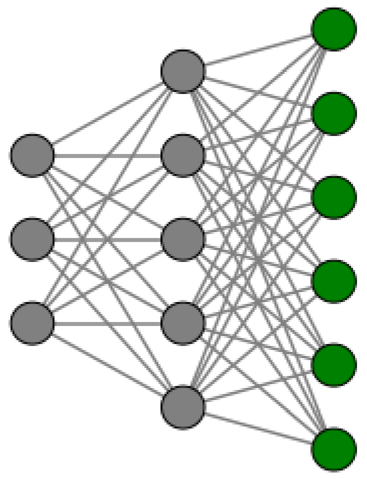
Building surrogate model



$T(i)|\mathcal{R}(\varphi)$

XGB classifier is trained utilizing features.

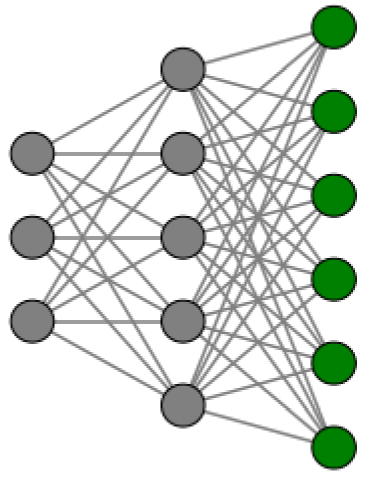
Assessing importances



$\mathcal{I}(T(i)|\mathcal{R}(\varphi))$

Feature importances are assessed.

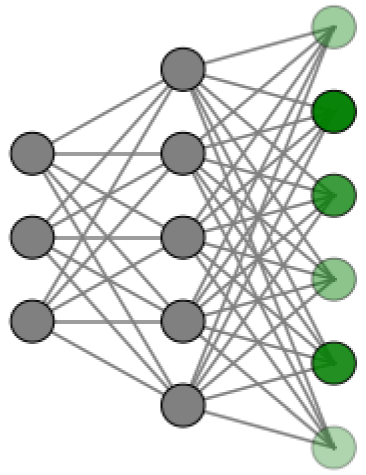
Assessing importances



$\leftarrow \mathcal{I}(T(i)|\mathcal{R}(\varphi))$

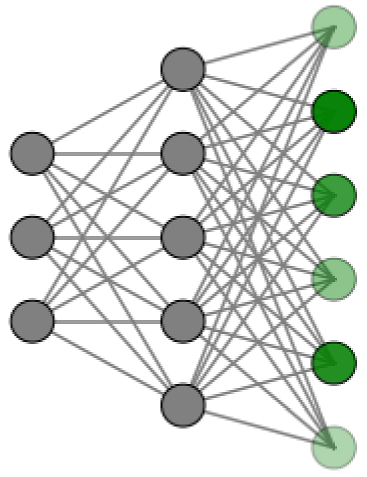
Feature importances are assigned to neurons (filters).

Features Assessed



Feature importances are assessed and assigned to neurons (filters).

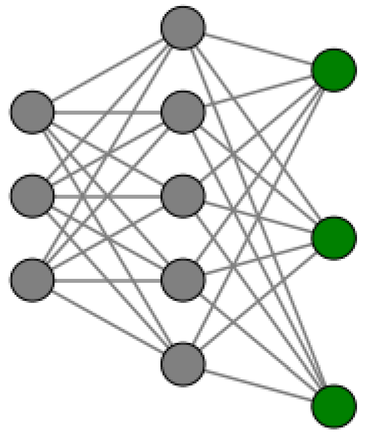
Application of pruning predicate



$\leftarrow P(j|\mathcal{T},\rho) = \mathcal{I}(T(i)|\mathcal{R}(\varphi)) > \rho$

Pruning predicate is applied.
Neurons (filters) with importance lower than set threshold are removed.

Layer Pruned

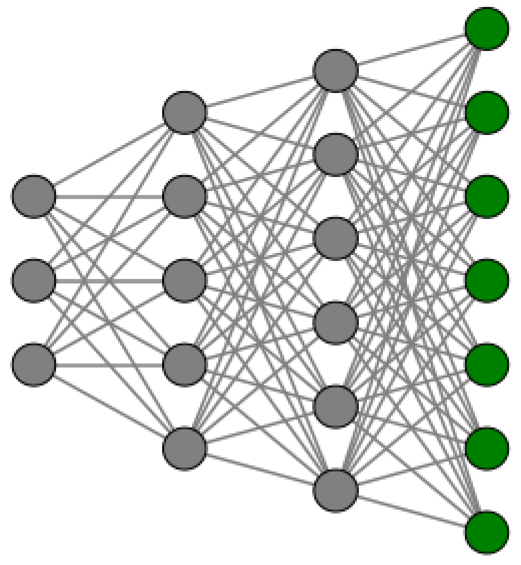


The network structure after pruning.

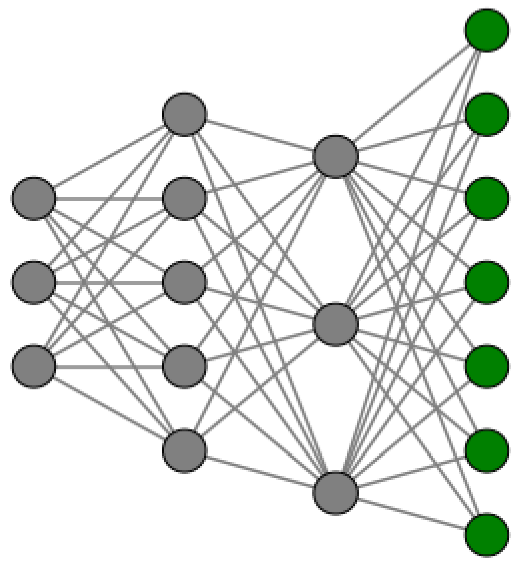
Mode selection

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.



Optimization target

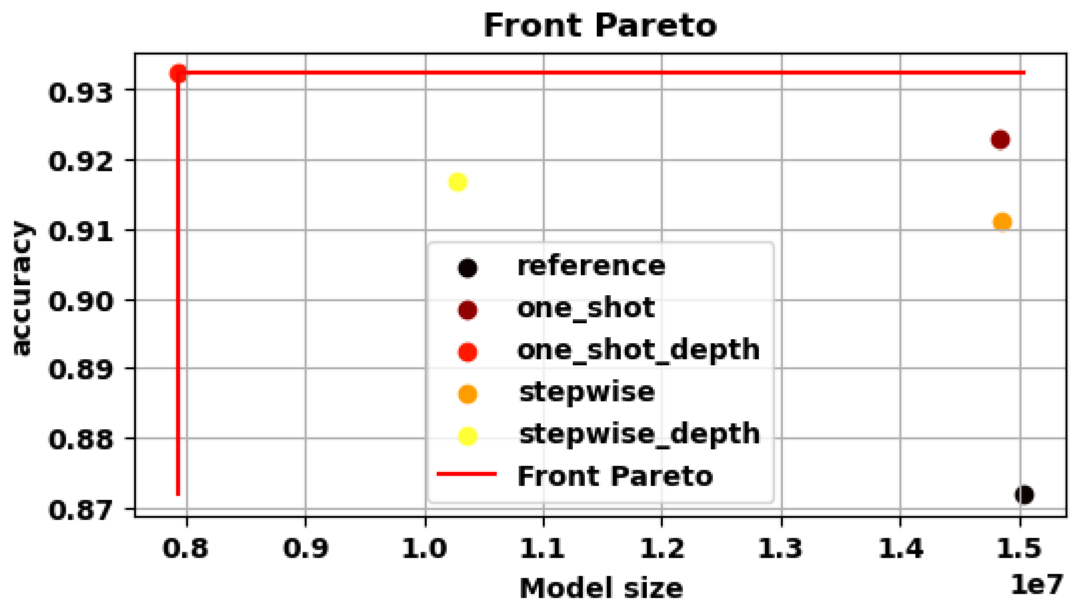
✓The proposed method can be used to reduce size, speed, costs, and improve performance.

✓Different modes usually yield a different model.

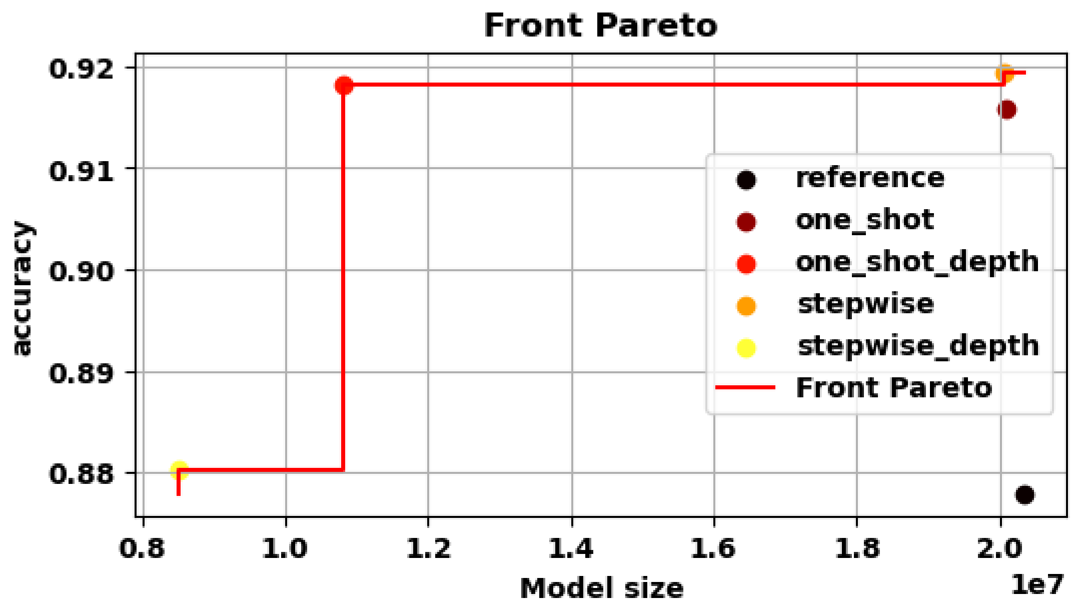
✓This paper considers two targets: **minimize model size**, that is, several parameters in a model that we aim to minimize and simultaneously **maximize model performance**, measured as accuracy over test datasets.

✓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.

Mode selection



VGG16



VGG19

Comparison of different pruning modes for VGG16 i VGG19.

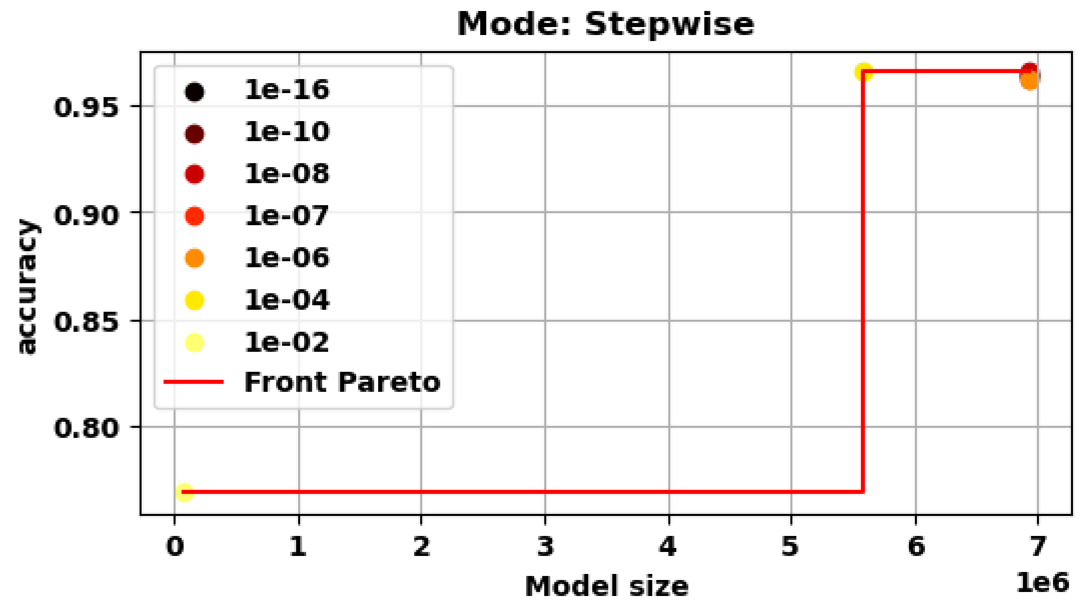
Due to cost-effective selection between all these modes
one can efficiently select best architecture for a particular use-case.

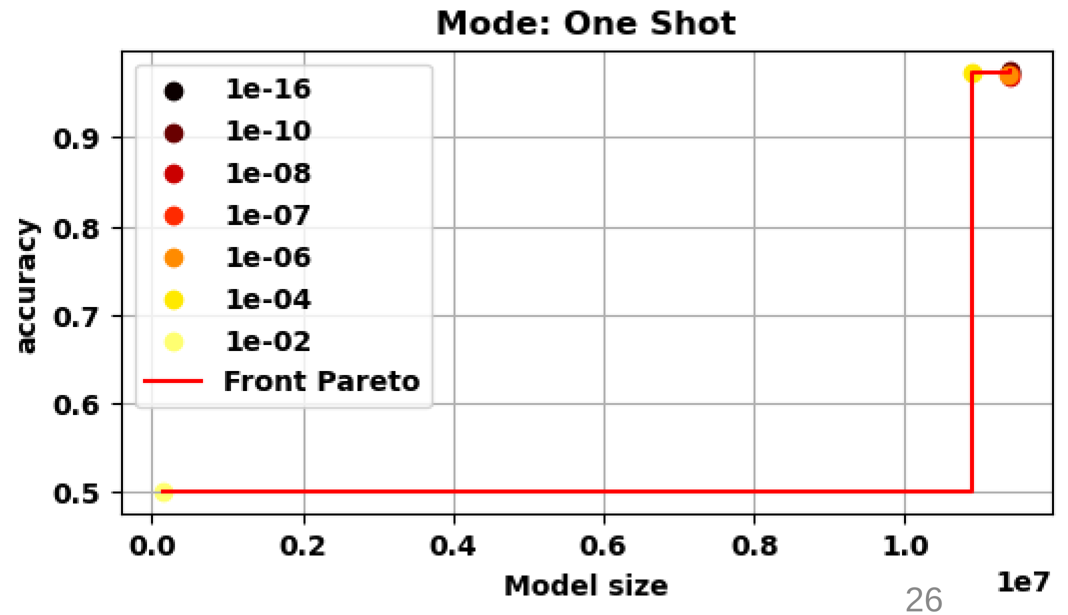
Threshold selection

The higher the threshold ρ, the smaller the output pruned model.

At the same time, the best-performing one has **47%** of the total parameter count of the reference model and **97 %** accuracy on the test dataset (95% was achieved by the reference model).

The most miniature architecture possible reached **<0.5%** of the total parameter count of the ref model and a test set accuracy decreased to **76%.**





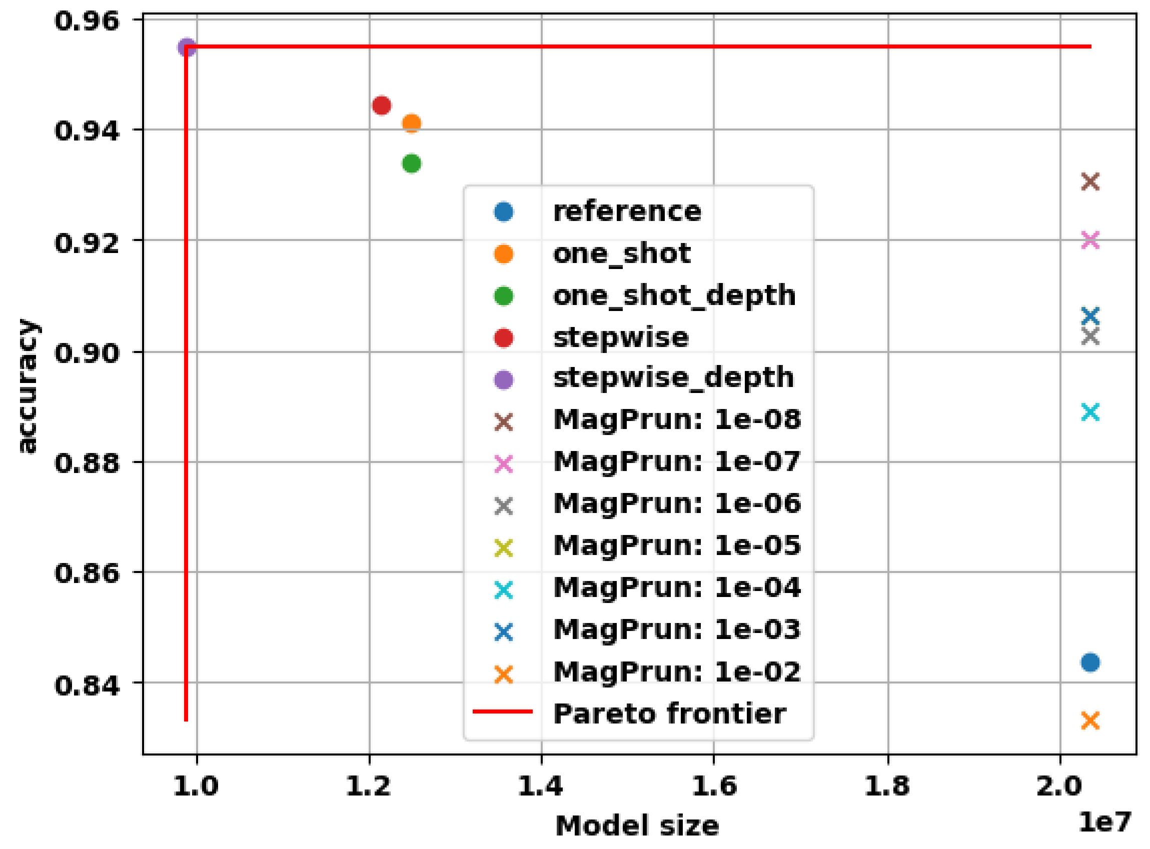
Results

Results of the XGB-based pruner for the different architectures and datasets:

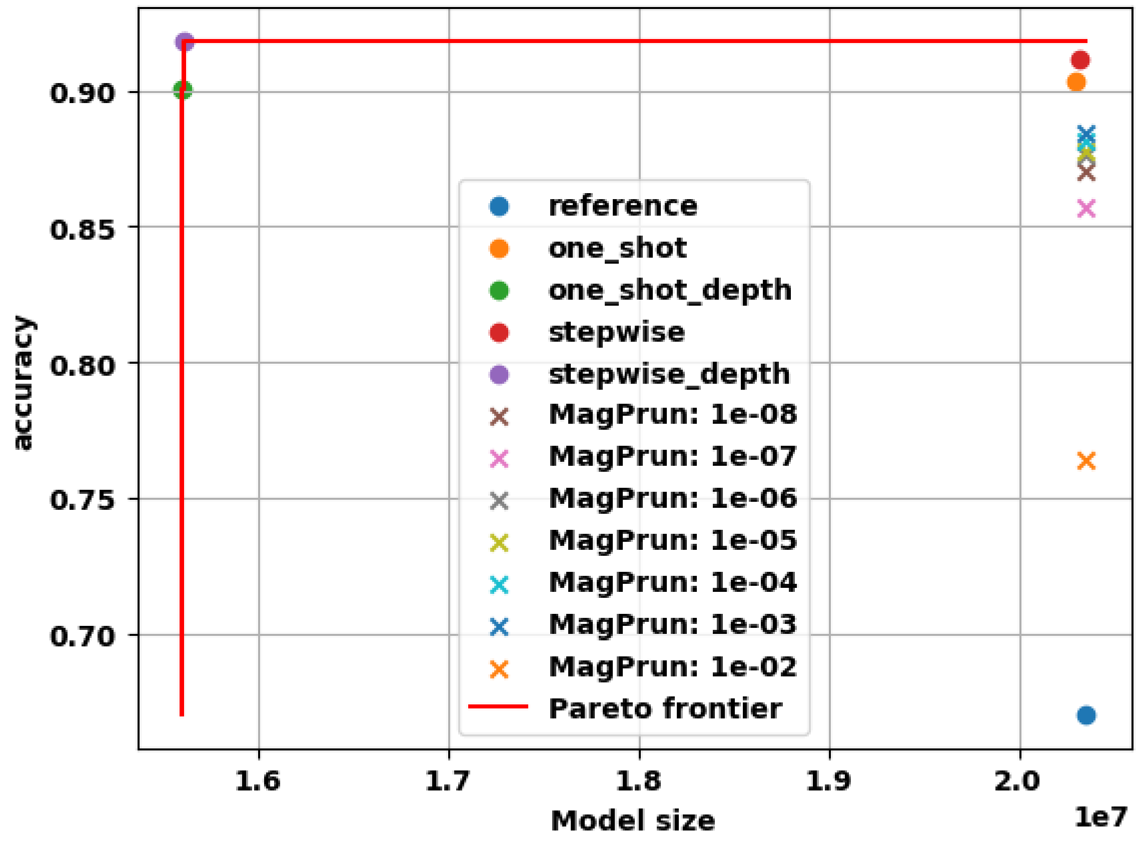
test dataset accuracy (# parameters)

Dataset	Mode	VGG16	VGG19	ResNet50
Binary Flowers	Reference	82% (1.5E7)	84% (2E7)	91% (2.3E7)
	Proposed	**93% (6.6E6)**	**95% (9.9E6)**	**92% (1.2E7)**
Flowers	Reference	75% (1.5E7)	67% (2E7)	90% (2.3E7)
	Proposed	**92% (1.2E7)**	**91% (1.6E7)**	**91% (2.2E7)**
Cats vs Dogs	Reference	74% (1.5E7)	81% (2E7)	98% (2.3E7)
	Proposed	**97% (7.4E6)**	**97% (1.1E7)**	96% **(7.9E6)**
Dr Eyes	Reference	95% (1.5E7)	95% (2E7)	96% (2.3E7)
	Proposed	**97% (4.1E6)**	**98% (9.3E6)**	95% **(8.2E6)**
Eye Disease	Reference	87% (1.5E7)	88% (2E7)	90% (2.3E7)
	Proposed	**93% (7.9E6)**	**92% (1E7)**	90% **(1.2E7)**

Comparison to the state-of-the-art model



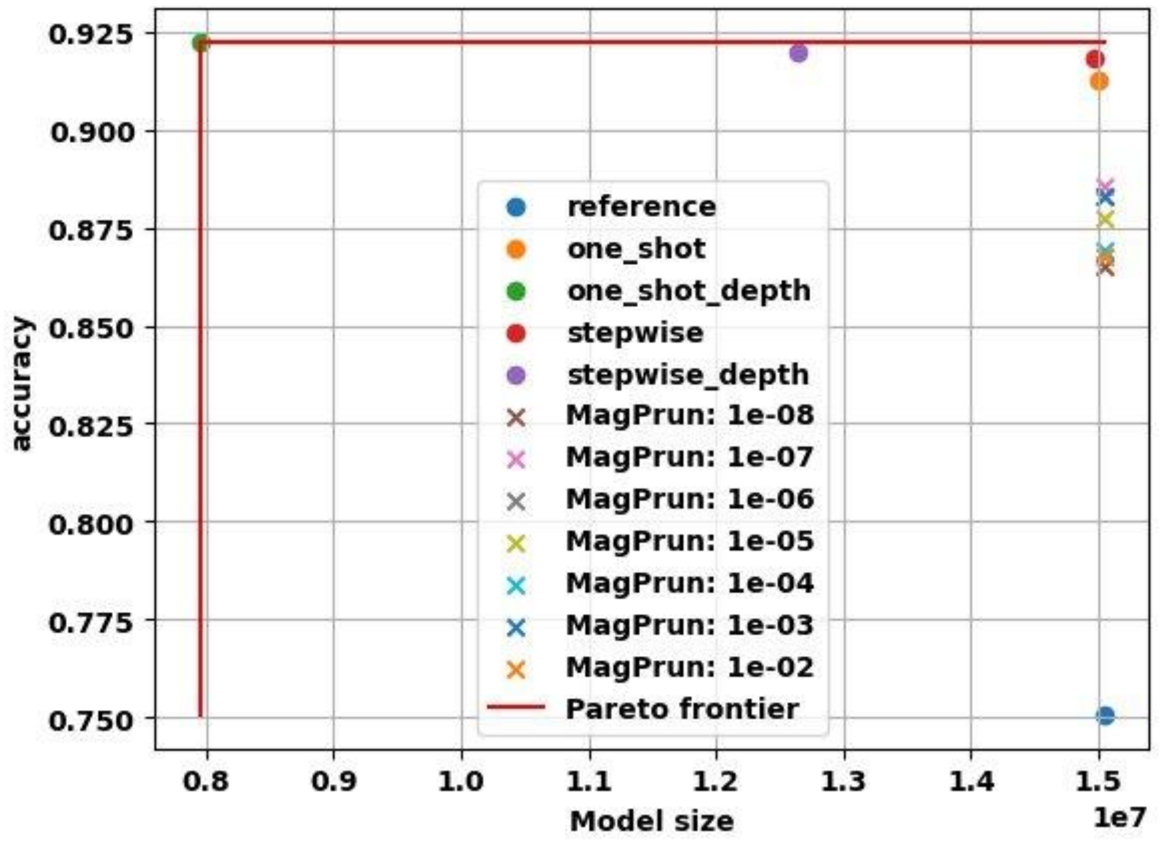
VGG19, Binary Flowers



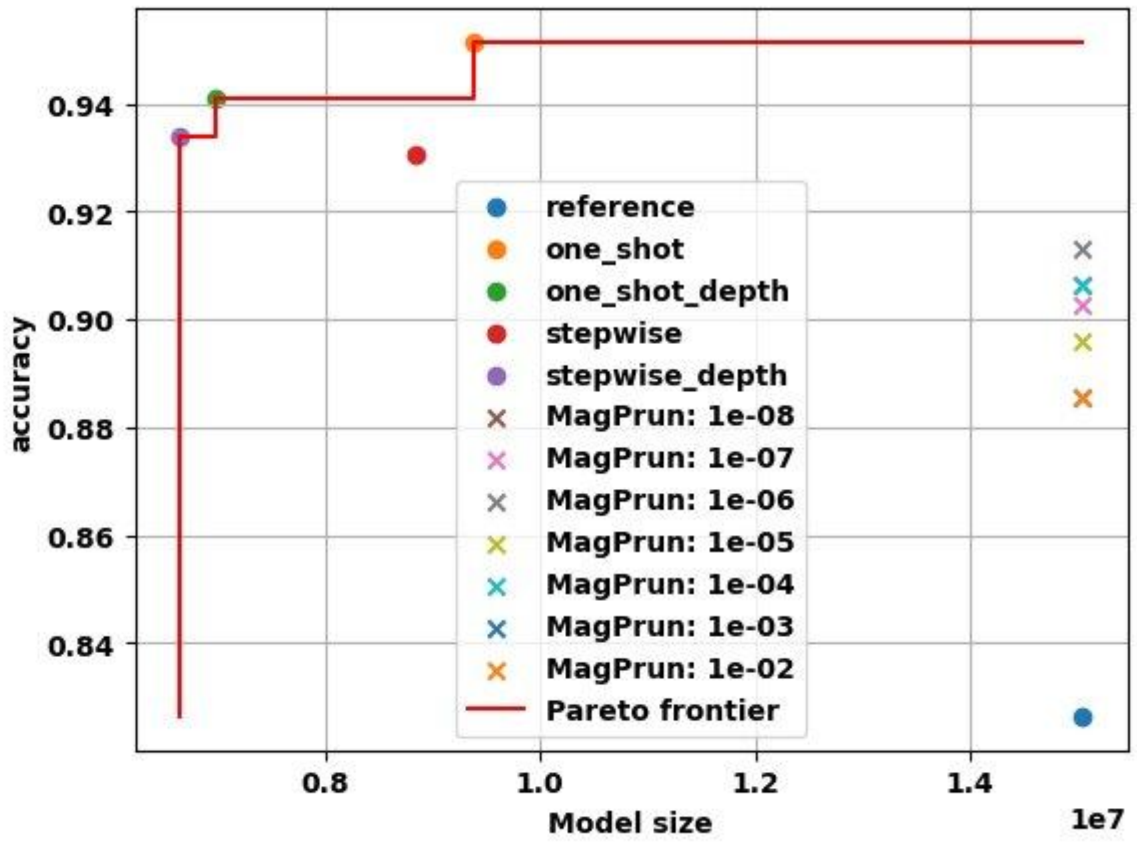
VGG19, Flowers

Comparisons of the proposed models with a **magnitude-based pruner**.

Comparison



VGG16, Binary Flowers



VGG16, Flowers

Comparisons of the proposed models with the **magnitude-based pruner**.

Speed Comparison

Comparison of the **XGB-based pruner** with the pruning algorithm proposed by [De Leon, Atienza]:

Dataset	# train samples	NN train time [s]	XGB train time[s]	Saved 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%

Conclusions

- **XBG-based pruner** solves the pruning problem for the different transfer learning models;
- **Depth pruning** mode can be performed with no additional computational overhead;
- **XGB-based pruner** is pretty robust and works well for different model architectures and computer vision datasets;
- Proposed method has superior results to most common pruning method.

Not always bigger is better ☺

Igor Ratajczyk

LinkedIn: linkedin.com/in/igor-ratajczyk/

Email: igor@ratajczyk.eu

Adrian Horzyk

LinkedIn: linkedin.com/in/adrian-horzyk-30658844/

Email: horzyk@agh.edu.pl

Web: https://home.agh.edu.pl/~horzyk/index-eng.php

Bibliography

- Igor Ratajczyk, Adrian Horzyk, Advancing ConvNet Architectures: A Novel XGB-based Pruning Algorithm for Transfer Learning Efficiency, Proc. of ECAI 2024, Volume 392, Frontiers in Artificial Intelligence and Applications, 2024, pp. 2114–2121.
- 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.
- 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