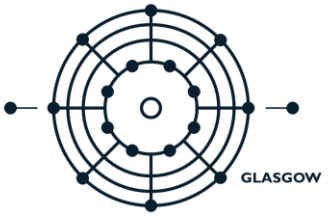

EG-ICE 2025
GLASGOW

Towards Compact AI Models For Efficient Machining Feature Recognition

KONSTANTINOS GKRISPANIS[1,2] STAVROS NOUSIAS[1,2] ANDRÉ BORRMANN[1,2]

[1]Chair of Computing in Civil and Building Engineering, Technical University of Munich, Munich, Germany
[2]TUM Georg Nemetschek Institute – AI for the Built World, Munich, Germany

ABSTRACT:
Machining feature recognition is the first step in the automation of the design and production pipeline. Currently, this process relies on manual annotation by human experts, which is time-consuming and prone to errors. Computer Numerical Control (CNC) machines are automated tools that use pre-programmed computer software to control machining processes with high precision and efficiency. Enhancing CNC machines with an AI-based approach for the recognition of machining features in the CAD (Computer Aided Design) input models eliminates the need for manual annotation and enables seamless integration of design and production workflows for optimized machining strategies. CNC controllers often operate in resource-constrained environments with limited computational capabilities. Therefore, there is a pressing demand for machining feature recognition models that can operate efficiently across these devices. In recent years, network pruning algorithms have gained significant attention from researchers due to the growing size and complexity of deep learning models, which often require considerable computational resources for training and development. Network pruning is a technique that reduces the size of deep learning models by removing unnecessary weights or entire structures (e.g., filters, channels). Despite their growing adoption in other domains, pruning strategies have not been explored in machining-specific AI models. In this paper, we evaluate four different scoring criteria combined with the Soft Pruning iterative procedure on BRepNet, a machining feature recognition model. Our experiments demonstrate that pruning not only preserves performance but can also lead to slight accuracy improvements for small pruning rates. Remarkably, when removing 90% of the model's parameters, one pruning criterion results in only 2% loss in accuracy. These findings highlight the potential of pruning as a practical approach to developing efficient and compact AI models for deployment in manufacturing and robotized construction environments.

KEYWORDS:
Machining Feature Recognition, Network Pruning, Deep Learning Optimization

1. INTRODUCTION

The first step towards the automation of the production pipeline involves identifying machining-relevant features within CAD models, which subsequently enables the generation of operations required to manufacture the desired object. Until

recent years, human expertise was essential for this step. This process has been completely transformed by the development of AI algorithms, which allow intelligent systems to be directly integrated with CNC machines. Despite these technological breakthroughs, a significant challenge persists; CNC machines and similar industrial computing environments that are used in production, are inherently resource-constrained and they were not originally designed to efficiently execute deep learning models (Manikanta *et al.*, 2024). These computational limitations create a bottleneck in the integration of AI technologies in manufacturing settings. With the continuous growth of machine learning models in depth and complexity, there is an increased need for sophisticated optimization techniques to ensure practical, sustainable and cost-effective optimization.

Network pruning is a field that has emerged in the recent years to tackle the problem of deep neural networks, aiming at reducing their size without a significant drop in their performance. Various pruning algorithms have been proposed, mainly in Computer Vision tasks (Li *et al.*, 2017; Luo, Wu and Lin, 2017; He *et al.*, 2019). Yet, their application and potential benefits in the domain of manufacturing domain remain unexplored.

Considering the above, we aim to examine the application of network pruning to a novel machining feature recognition model, named BRepNet (Lambourne *et al.*, 2021). BRepNet can be configured in multiple ways, depending on the usage of UV-grids (Jayaraman *et al.*, 2021) and the selection of input features. In this work, we evaluate two configurations. One with the use of UV-grids (surface encoder) and another that represents the most compact version of the network architecture while still yielding competitive performance.

We evaluate four different scoring criteria combined with the Soft Pruning (He *et al.*, 2018) iterative procedure on BRepNet (Lambourne *et al.*, 2021). The scoring criteria are namely: the Geometric Median (He *et al.*, 2019), the L2-Norm, the Activation Importance (Ardakani, Condo and Gross, 2017) and the Group Taylor importance criterion (Fang *et al.*, 2023). Experimental results on the 360Fusion dataset show that the proposed approach has the potential to generate even more compact networks with state-of-the-art performance, especially when small pruning rates are used. Overall, we provide a comprehensive analysis of the above criteria and determine which of them -if any- offers a notable advantage in manufacturing domain. The paper contributions can be summarized as follows:

- To the best of our knowledge, this is the first study to investigate the effects of network pruning in the domain of machining feature recognition.
- The proposed approach yields a family of compact models suitable for deployment on resource-constrained environments.
- We present a comprehensive comparison of multiple pruning criteria, offering insights into the effectiveness of them and giving the opportunity for applications in other domains.
- We observe performance improvement for the baseline BRepNet segmentation model.

The paper is organized as follows: Section 2 presents the related work on Segmentation of CAD-Derived representations and Network Pruning. Section 3 explains the methodology used in this paper. Section 4 presents the experimental settings used, Section 5 the results of the experiments and finally, Section 6 concludes our work.

2. RELATED WORK

2.1 Segmentation of CAD-Derived Representations

In the literature, many models have been proposed for segmenting representations derived from CAD models, such as point clouds and meshes. Prior approaches like PointNet++ (Qi *et al.*, 2017), DGCNN (Wang *et al.*, 2019) and MeshCNN (Hanocka *et al.*, 2019) have demonstrated strong performance, but these methods require transformations that may lead to information loss. Moreover, these models tend to have a large number of parameters, making them less suitable for resource-constrained environments such as CNC machines.

To address these challenges, CADNet (Colligan *et al.*, 2022) and BRepNet (Lambourne *et al.*, 2021) were specifically designed for machining feature recognition. CADNet represents CAD models as hierarchical graph structures, capturing surface geometry and face topology. BRepNet, on the other hand, operates on BRep structures and is already a relative compact model compared to other approaches.

While BRepNet is more lightweight than the other models, further optimization is needed for real-time deployment. This motivates the use of pruning algorithms to further reduce the size while preserving -or even enhancing- performance.

2.2 BRepNet

BRepNet (Lambourne *et al.*, 2021) is a neural network specifically designed for machining feature recognition in CAD models. It operates directly on boundary representation (BRep) data structures, aiming to improve the segmentation of BRep

models. The core innovation of BRepNet lies in its convolutional kernels, which are defined with respect to oriented edges within the BRep structure.

To extract meaningful features, BRepNet identifies localized neighborhoods of faces, edges and coedges around each coedge, utilizing learnable parameters to detect patterns in these feature vectors. A key concept introduced by BRepNet is topological walks, which enable the model to navigate the BRep structure. These walks are guided by adjacency relationships - such as next, previous and mating connections - allowing the network to traverse coedges, edges and faces effectively.

BRepNet supports different kernel configurations, which determine how geometric and topological information is captured and processed. Each configuration defines a distinct set of topological walks, aggregating information from adjacent elements.

In this work, we evaluate two versions of the BRepNet architecture. The first is a larger configuration with a surface encoder, which includes UV-grid embeddings to enhance face representations. It contains a total of 603,532 trainable parameters. The second configuration is the most compact version of BRepNet. It excludes the surface encoder entirely and does not use the topological convolution kernel. Instead, it relies solely on a lightweight MLP. This version contains only 181,196 parameters, making it a robust foundation for assessing the effectiveness of pruning strategies. Tables 1 and 2 provide the architectural breakdown of both model variants.

Table 1: BRepNet with Surface Encoder

Module	*Parameters*
Surface Encoder	390K
Module List	211.9K
Classification layer	680

Table 2: MLP-only BRepNet

Module	*Parameters*
Module List	180.5K
Classification layer	680

2.3 Network Pruning

Network Pruning can be roughly categorized into unstructured and structured pruning. Unstructured pruning removes individual weights inside a weight matrix and structured pruning removes whole network structures like filters or channels (He *et al.*, 2019). Unstructured pruning creates a model with irregular sparsities, while structured pruning yields models that can be easily deployed. For the rest of the paper, we will focus on structured pruning.

There are various criteria for determining which filters or channels to prune, ranging from simple approaches, like removing components based on their magnitude (e.g., L2-norm of weights), to more advanced techniques that consider the impact of each component on the network's output, such as Group Taylor importance criterion (Fang *et al.*, 2023).

3. METHODOLOGY

3.1 Scoring Criteria

Consider an individual layer in a neural network with weight parameters,

$$F = [F_1, \dots, F_n], (1)$$

where $F_j \in R^{k \times k \times c}$ is the j-th filter with spatial size $k \times k$ and depth c, and n is the total number of filters in the layer. Based on this formulation, the goal of the proposed approach is: given a pruning rate θ, prune the $n\theta$ filters in each layer of the model.

Redundancy-based pruning algorithms, i.e. algorithms that identify and remove filters in a layer with the most similar characteristics, have been shown to outperform other criteria in various contexts (He *et al.*, 2019; Gkrispanis, Gkalelis and Mezaris, 2024). To this end, we adapt the Geometric Median (He *et al.*, 2019) algorithm, a method that has demonstrated strong performance across different architectures and application domains by removing filters closest to the geometric centre of the layer.

For comparison purposes, we additionally evaluate three alternative importance criteria: the GroupTaylor (Molchanov *et al.*, 2017), which estimates the impact of each group of weights on the loss function using the first-order Talyor expansion; gradient-based importance (Fang *et al.*, 2023), which ranks parameter sensitivity based on backpropagation gradients; and the well-known L2 Norm, which evaluates parameters based on their Euclidean magnitude. All four pruning criteria are described briefly in the following:

- **L2 Norm:** The L2 Norm pruning algorithm evaluates the significance of groups of weights (such as filters or channels) in model's layers based on their L2 Norm. It is based on the same assumption as the L1 Norm algorithm, which has been tested in various works (Li *et al.*, 2017;

Kumar *et al.*, 2021). It is based on the traditional "smaller-norm-less-important" criterion. Given filter F_j, its L2 norm is computed as:

$$norm(F_j, 2) = \sum_{i=1}^{ck^2} |f_{i,j}| \quad (2),$$

where $f_{i,j}$ is the i-th element of F_j. Filters with smaller L2 norm values, which indicate lower overall importance, are pruned. This methodology is inspired by the intuition that smaller-norm weight filters contribute less to model's final prediction.

- **Geometric Median:** This algorithm was proposed by (He *et al.*, 2019). It is based on Geometric Median (GM), the classic robust estimator of centrality for data in Euclidean spaces, to prune redundant filters in a layer. Unlike magnitude-based methods such as the L2 Norm, GM focuses on minimizing information redundancy by pruning filters that are closer to the geometric center of the filter set. As defined in (He *et al.*, 2019), the GM of a set of filters, denoted as x^{GM}, is mathematically expressed as:

$$x^{\mathrm{GM}} = \arg\min_{x \in R^{k \times k \times c}} \sum_{j' \in [1,n]} ||x - F_{j'}||_2, \qquad (3)$$

where x represents a filter in a layer and is used as a placeholder to denote any filter from the set of filters in that layer. Subsequently, the filter in the layer closest to this geometric median is identified by:

$$F_{j^*} = \arg\min_{F_{j'}} |F_{j'} - x^{\mathrm{GM}}|_2, \text{ s.t. } j' \in [1, n]. \quad (4)$$

To avoid the high computational cost of solving for the geometric median, an efficient approximation in employed. Specifically, the algorithm directly selects the filter that minimizes the sum of pairwise distances to all other filters:

$$F_{x^*} = \arg\min_{x} g(x), \text{ s.t. } x \in \{F_1, \ldots, F_n\} \quad (5)$$

where the function $g(x)$ is defined as:

$$g(x) = \sum_{j'=1}^{n} |x - F_{j'}|_2. \quad (6)$$

The filter F_{x^*} that minimizes the aggregate distance $g(x)$ is considered the most redundant and can be pruned, resulting in minimal impact on the network's representational capacity.

- **Group Taylor:** Group Taylor importance criterion estimates the contribution of each convolutional filter to the network's loss using a first-order Taylor expansion. It evaluates how the removal of a filter is expected to affect the loss function by leveraging both the filter's weight and its gradient. Let $W \in R^{C_{\mathrm{out}} \times C_{\mathrm{in}} K_h \times K_w}$, denote the 4D convolutional weight tensor, where C_{out}, is the number of output channels (i.e., filters). The importance of the i-th filter $W_i \in R^{C_{\mathrm{in}} \times K_h \times K_w}$ is:

$$I(W_i) = \left| W_i \cdot \frac{\partial \mathcal{L}}{\partial W_i} \right|, \quad \forall i \in [1, C_{\mathrm{out}}], (9)$$

where $\mathcal{L}$ is the loss function, w_i is the i-th filter (i.e., output channel), and $\frac{\partial \mathcal{L}}{\partial W_i}$ is its gradient. The dot product is element-wise and summed over all dimensions. If the layer includes biases $b \in \mathbb{R}^{C_{out}}$, their importance is similarly computed as:

$$I(b_i) = \left| b_i \cdot \frac{\partial \mathcal{L}}{\partial b_i} \right|, \quad \forall i \in [1, C_{\mathrm{out}}]. (10)$$

Filters and biases with the lowest Taylor importance values are considered the least critical and are pruned accordingly.

- **Gradient based:** Gradient based pruning computes the importance of each convolutional filter based on the norm of its gradient with respect to the loss function, assuming that filters with larger gradients contribute more significantly to loss minimization. For a convolutional weight tensor $W_i \in R^{C_{\mathrm{out}} \times C_{\mathrm{in}} K_h \times K_w}$, the importance of the i-th output filter is:

$$I(W_i) = \left| \frac{\partial \mathcal{L}}{\partial W_i} \right|_p, \quad \forall i \in [1, C_{\mathrm{out}}], (11)$$

where $\frac{\partial \mathcal{L}}{\partial W_i}$ is the gradient of the loss with respect to the i-th filter, and $|\cdot|_p$ denotes the p-norm (e.g., $p = 2$ for the Euclidean norm). Filters with smaller gradient norms are considered less important and are pruned.

When pruning is applied to linear layers, it corresponds to removing entire neurons, either from the input or output of the layer. In this context, the

pruning process removes columns or rows of the weight matrix depending on whether the goal is to reduce input or output dimensionality. For a linear layer with weight matrix $W \in R^{m \times n}$, where m is the number of output features and n is the number of input features. Pruning input neurons corresponds to removing columns $w_j \in R^m$, for $j = 1, \ldots, n$ and pruning output neurons corresponds to removing rows. Similar to convolutional filters, various criteria (e.g., L2 norm, Geometric Median) can be applied to these column or row vectors to evaluate their importance and determine which ones to prune. In our experiments, we apply the same scoring criteria used in convolutional layers on the corresponding dimensions of linear layers, allowing a unified pruning strategy across all model components.

3.2 Soft Pruning

Table 3: Soft Pruning Algorithm

Input: Pre-trained model **M**, example input **X**, labels **Y**, pruning ratio **r**, pruning steps **N**, criterion (e.g., FPGM), loss and forward functions

1. Train BRepNet to 360Fusion
2. Initialize pruner P with M, X, Y, r, criterion
3. Define ignored layers (e.g., final classification layer)

---

4. **Repeat N times:**
5. Compute outputs and gradients on M(X)
6. Apply soft pruning (zero weights)
7. Fine-tune model
8. Print sparsity

---

9. Apply hard pruning to M
10. Adjust model architecture post-pruning
11. Final fine-tuning with lower LR
12. Test pruned model

Output: Pruned model

We combine all the algorithms with the Soft Pruning (He *et al.*, 2018) procedure to prune BRepNet in an iterative manner. Soft pruning is an iterative method to prune deep networks and is designed to address limitations in traditional pruning techniques, which permanently prune filters or channels and thereby reduce the model's representational capacity. Soft Pruning allows pruned filters to be updated during subsequent model training. One main advantage offered by this approach is that it preserves a larger model capacity since updating previously pruned filters provides a broader optimization space compared to permanently setting filters to zero. This larger optimization space allows the network to better learn from training data (He *et al.*, 2019). The soft pruning algorithm used in our experiments is summarized in Table 3.

4. EXPERIMENTS

4.1 Experimental Settings

Experiments were conducted using the following pruning rates $\theta = \{0.1, 0.2, 0.4, 0.8\}$. The pruning rate refers to sparsity pre pruned layer. All network layers, except for the last one that is used to output the predictions for the faces, are pruned. For the employment of the algorithms, torch-pruning (Fang *et al.*, 2023) library was utilized.

4.2 Dataset and Metrics

The dataset in which the BRepNet was originally trained and evaluated is the 360Fusion Dataset. It is a big dataset containing around 36k 3D CAD models in BRep format. The CAD models are segmented to 8 classes related to machining operations. The classes are namely: ExtrudeSide, ExtrudeEnd, Chamfer, Fillet, RevolveEnd, RevolveSide, CutEnd and CutSide.

The metrics used to evaluate model's performance are the typical metrics used for evaluating segmentation models. In the following equations, let *K* represent the total number of classes, p_{ii} denote the correct predictions for class i, p_{ij} represent the misclassifications from class i to class j, and so on. The equations for the most important metrics are as follows:

- **Accurary:** computes the ratio between the number of truly classified samples and the total number of samples.

$$\mathrm{Acc} = K \sum_{i=0}^{p} \frac{p_{ii}}{\sum_{j=0}^{K} p_{ij}} \quad (12)$$

- **Mean Intersection over Union:** computes the intersection ration between ground truth and predicted value averaged over the total number of classes *K*.

$$mIoU = \frac{1}{K+1} \sum_{i=0}^{K} \frac{p_{ii}}{\sum_{j=0}^{K} p_{ij} + \sum_{j=0}^{K} p_{ji} - p_{ii}} \quad (13)$$

5. RESULTS

As mentioned in Section 2.2, we evaluate the pruning performance on two setups of BRepNet: (1) the version with the surface encoder, and (2) a minimal version consisting only of linear layers (MLPs). The models are tested under various pruning rates ranging from 10% to 90% and we report Accuracy and Mean IoU to assess performance. Our goal is to determine which pruning criteria preserve – or even improve - performance while significantly reducing model size.

5.1 BRepNet with Surface encoder

We first apply pruning to the BRepNet with the surface encoder at a modest pruning rate of 10%. This will give us insights into how each pruning criterion performs under light compression, where model capacity is still largely retained.

Table 4: Results for 10% Pruning Rate – BRepNet with Surface Encoder

Criterion	*Parameters*	*Model Size*	*Accuracy*	*Mean IoU*
Original	**603.532**	**2.414MB**	**0.9534**	**0.8318**
L2-Norm	587.711	2.351MB	**0.9545**	**0.8390**
FPGM	587.711	2.351MB	**0.9571**	**0.8405**
Taylor	587.711	2.351MB	**0.9579**	**0.8461**
Gradient	587.711	2.351MB	**0.9578**	**0.8447**

At 10% pruning rate, all criteria retain and even slightly improve performance. This complies with findings from Computer Vision experiments which have shown that pruning may act as a form of regularization, allowing the model to generalize better. All methods maintain model size reductions while preserving high accuracy.

Table 5: Results for 20% Pruning Rate – BRepNet with Surface Encoder

Criterion	*Parameters*	*Model Size*	*Accuracy*	*Mean IoU*
Original	**603.532**	**2.414MB**	**0.9534**	**0.8318**
L2-Norm	549.668	2.199MB	0.9504	0.8135
FPGM	549.668	2.199MB	**0.9540**	0.8237
Taylor	549.668	2.199MB	0.9483	0.8036
Gradient	549.668	2.199MB	**0.9540**	0.8274

While we increase the pruning rate, FPGM and Group Taylor criteria show a small increase in accuracy but not in mean IoU, as it can be seen in Table 5.

We then test an extreme pruning scenario, reducing 90% of the parameters from the BRepNet. This setup helps evaluate the pruning criteria in an aggressive compression scenario.

Table 6: Results for 90% Pruning Rate – BRepNet with Surface Encoder

Criterion	*Parameters*	*Model Size*	*Accuracy*	*Mean IoU*
Original	**603.532**	**2.414MB**	**0.9534**	**0.8318**
L2-Norm	29.883	0.120MB	0.8435	0.5983
FPGM	29.883	0.120MB	0.9398	0.7887
Taylor	29.883	0.120MB	0.9084	0.6876
Gradient	29.883	0.120MB	0.8988	0.6479

As expected, L2 Norm degrades significantly, with a drop in both accuracy and mean IoU. However, the remaining algorithms maintain high accuracy despite the extreme compression. FPGM achieves 0.9398 accuracy – corresponding to less than 2% loss compared to the original model – while also retaining a strong mean IoU. Group Taylor and Gradient criteria also preserve performance, reaching 0.9084 and 0.8988 accuracy, respectively. The results in Table 6, indicate that redundancy still exists even in high-capacity models like BRepNet, and careful pruning can exploit it.

5.2 MLP – only Compact BRepNet

For the second setup of experiments, we test the algorithms to the most compact version of the BRepNet, the model with only MLP layers was used. We decided to evaluate the pruning ratios $\theta = \{0.2, 0.4\}$. Pruning ratios bigger than 0.4 removed critical parts and the network was not able to recover, so we only tested these 2 pruning ratios.

Table 7: Results for 20% Pruning Rate – BRepNet only with MLP layers

Criterion	*Parameters*	*Model Size*	*Accuracy*	*Mean IoU*
Original	**181.196**	**0.725MB**	**0.9390**	**0.8076**
L2-Norm	148.573	0.594MB	0.9338	0.8052
GM	148.573	0.594MB	**0.9407**	**0.8079**
Taylor	148.573	0.594MB	0.9341	0.7806
Gradient	148.573	0.594MB	0.9373	0.8015

Table 8: Results for 40% Pruning Rate – BRepNet only with MLP layers

Criterion	*Parameters*	*Model Size*	*Accuracy*	*Mean IoU*
Original	**181.196**	**0.725MB**	**0.9390**	**0.8076**
L2-Norm	107.875	0.431MB	0.9335	0.7903
GM	107.875	0.431MB	0.9320	0.7837
Taylor	107.875	0.431MB	0.9335	0.7855
Gradient	107.875	0.431MB	0.9330	0.7859

From Table 8, we notice that the Geometric Median criterion is the only one to yield a minor accuracy improvement at 20% pruning rate. This

observation highlights GM's ability to capture redundancy even in models that are already compact.

Figure 1: BRepNet with Surface Encoder

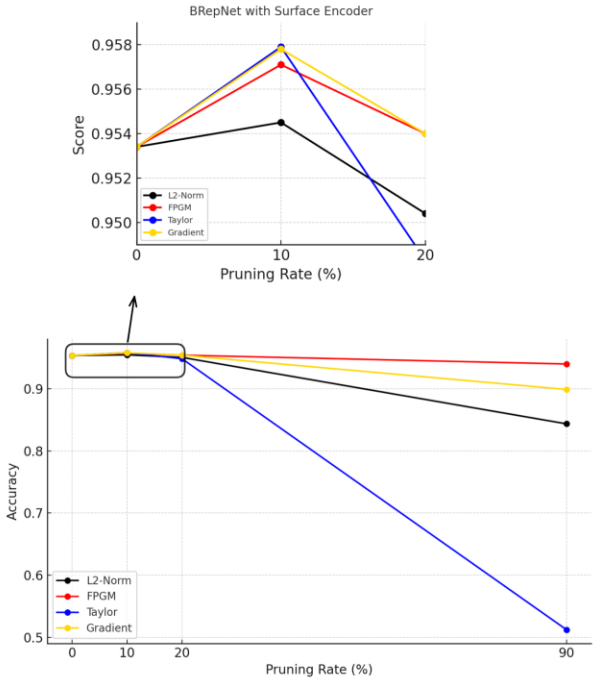

Figure 2: MPL BRepNet

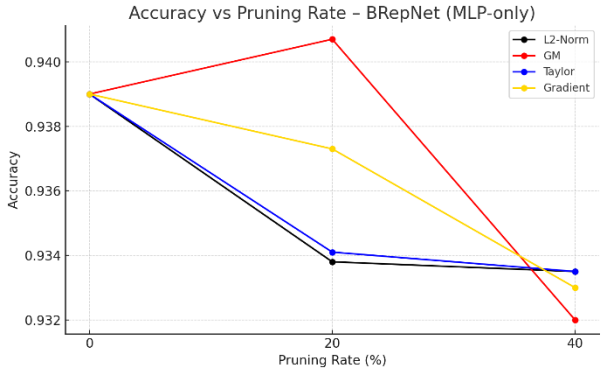

Figure 1 shows the accuracy trends for all pruning methods applied to BRepNet with the surface encoder, while Figure 2 presents the results of the experiments for the compact version of the model.

6. LIMITATIONS AND FUTURE WORK

The scope of our study was limited to structured pruning on a specific architecture (BRepNet) and dataset (360Fusion). While this setup provided a controlled environment for evaluating pruning effectiveness, it does not yet account for the performance of such approaches across different datasets, industrial conditions, or other CAD model representations.

Future work could extend this approach to other models designed for CAD-based reasoning, such as CADNet and UV-Net, or explore combinations with quantization and knowledge distillation for even greater efficiency. Evaluating generalization under real-world deployment scenarios and developing adaptive pruning schedules may further support robust deployment on embedded industrial systems.

7. CONCLUSION

In this work, we explored the application of structured pruning on BRepNet, a model designed for machining feature recognition in CAD models. We evaluated four pruning strategies across two configurations – a high-capacity version with a surface encoder and an extremely compact MLP-only version of the model. Our results demonstrated that pruning can act as a form of regularization, maintaining or even improving accuracy at low pruning rates. Notably, we achieved less than 2% accuracy loss after removing over 90% of the model parameters, with the Geometric Median. These findings showcased the potential of structured pruning to enable efficient deployment of AI models in resource-constrained manufacturing environments. To the best of our knowledge, this was the first work to explore such techniques in the context of BRepNet and machining-specific AI systems.

REFERENCES

Ardakani, A., Condo, C. and Gross, W.J. (2017) 'Activation pruning of deep convolutional neural networks', in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Montreal, QC: IEEE, pp. 1325–1329. Available at: https://doi.org/10.1109/GlobalSIP.2017.8309176.

Colligan, A.R. *et al.* (2022) 'Hierarchical CADNet: Learning from B-Reps for Machining Feature Recognition', *Computer-Aided Design*, 147, p. 103226. Available at: https://doi.org/10.1016/j.cad.2022.103226.

Fang, G. *et al.* (2023) 'DepGraph: Towards Any Structural Pruning', in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada: IEEE, pp. 16091–16101. Available at: https://doi.org/10.1109/CVPR52729.2023.01544.

Gkrispanis, K., Gkalelis, N. and Mezaris, V. (2024) 'Filter-Pruning of Lightweight Face Detectors

Using a Geometric Median Criterion', in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, Waikoloa, HI, USA: IEEE, pp. 280–289. Available at: https://doi.org/10.1109/WACVW60836.2024.00037.

Hanocka, R. *et al.* (2019) 'MeshCNN: a network with an edge', *ACM Transactions on Graphics*, 38(4), pp. 1–12. Available at: https://doi.org/10.1145/3306346.3322959.

He, Y. *et al.* (2018) 'Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks'. arXiv. Available at: https://doi.org/10.48550/arXiv.1808.06866.

He, Y. *et al.* (2019) 'Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration', in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, pp. 4335–4344. Available at: https://doi.org/10.1109/CVPR.2019.00447.

Jayaraman, P.K. *et al.* (2021) 'UV-Net: Learning from Boundary Representations', in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, pp. 11698–11707. Available at: https://doi.org/10.1109/CVPR46437.2021.01153.

Kumar, A. *et al.* (2021) 'Pruning filters with L1-norm and capped L1-norm for CNN compression', *Applied Intelligence*, 51(2), pp. 1152–1160. Available at: https://doi.org/10.1007/s10489-020-01894-y.

Lambourne, J.G. *et al.* (2021) 'BRepNet: A topological message passing system for solid models'. arXiv. Available at: https://doi.org/10.48550/arXiv.2104.00706.

Li, H. *et al.* (2017) 'Pruning Filters for Efficient ConvNets'. arXiv. Available at: https://doi.org/10.48550/arXiv.1608.08710.

Luo, J.-H., Wu, J. and Lin, W. (2017) 'ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression', in *2017 IEEE International Conference on Computer Vision (ICCV)*. *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice: IEEE, pp. 5068–5076. Available at: https://doi.org/10.1109/ICCV.2017.541.

Manikanta, J.E. *et al.* (2024) 'Machine Learning and Artificial Intelligence Supported Machining: A Review and Insights for Future Research', *Journal of The Institution of Engineers (India): Series C*, 105(6), pp. 1653–1663. Available at: https://doi.org/10.1007/s40032-024-01118-z.

Molchanov, P. *et al.* (2017) 'PRUNING CONVOLUTIONAL NEURAL NETWORKS FOR RESOURCE EFFICIENT INFERENCE'.

Qi, C.R. *et al.* (2017) 'PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space'. arXiv. Available at: https://doi.org/10.48550/arXiv.1706.02413.

Wang, Y. *et al.* (2019) 'Dynamic Graph CNN for Learning on Point Clouds', *ACM Transactions on Graphics*, 38(5), pp. 1–12. Available at: https://doi.org/10.1145/3326362.