RESEARCH ARTICLE

COMPARATIVE STUDY OF STRUCTURED PRUNING ON A CUSTOM UNET MODEL USING UNIFORM VS VARIABLE PRUNING RATE

Sangita S. Modi

Department of Computer Science,

N.E.S. Science College, Nanded, Maharashtra, India

Corresponding author E-mail: sangita19modi@gmail.com

DOI: https://doi.org/10.5281/zenodo.18069726

Abstract:

Numerous researchers have developed advanced deep learning algorithms achieving outstanding performance in medical detection, classification, and diagnostic support. UNet [2] is one of the most widely used deep learning models, specifically designed for segmentation tasks [3]. Its architecture is tailored to deliver high accuracy, making it an effective and reliable tool for image segmentation. Its encoder-decoder architecture with skip connections allows it to capture both low-level and high-level features, making it highly effective for precise image segmentation. However, due to its large number of parameters, UNet can be computationally intensive, which makes model pruning—particularly structured pruning [5]—a valuable approach to reduce model size and inference time while maintaining segmentation performance. Simple Structure pruning can remove an entire channel/filter or even layer based on some pruning ratio and rebuild a narrow model with the regular model structure. Proposed experiment performs a comparison study of impact of structured pruning with uniform rate of 20% and structured pruning with custom pruning rate on the proposed UNet model designed and trained to detect infection regions with lung CT and corresponding infection masks, obtained from COVID-19 dataset. The model acquired very good training dice value 91% and testing dice impressive dice coefficient of 90%. The trained model is then pruned to remove less important filters using a one-shot method based on L1 norm value. The filters to prune is determined by calculating L1 norm [6] for all the layers. Two strategies were used in pruning: 1) using uniform pruned ratio of 20% across all the layers removes 20% of filters with the lowest L1 norm and obtained training and testing dice value little higher than original model of 92.63%, 91.04% respectively and 2) using custom pruning ratio, defined by analyzing pruning impact on each layer in step 1. We observed that result is improved with training dice of 93.43% and testing is 91.41%.

Keywords: UNet, COVID-19, Lung CT, CLACHE, Structured pruning, Custom Pruning, dice coefficient, L1 norm.

1. Introduction:

The worldwide healthcare sector was overrun by the COVID-19 [1] epidemic in 2019, which increased demand for diagnostic instruments. Although lung abnormalities were detected using imaging techniques such as X-ray, CT, and ultrasound, the manual analysis process was time-consuming. It has strongly raised the requirement of precise automated techniques that can be used for the lung segmentation or lesion detection tasks efficiently.

Deep Learning (DL) were the most widely used models in medical imaging and demonstrated remarkable performance in different tasks. The goal of DL models in lesion detection in medical imaging is to locate and identify particular regions in an image that are clinically important. Identification of ROIs accurately can help greatly in the early disease detection and planning of the treatment carefully. UNet is a popular Convolutional Neural Networks (CNNs), a type of deep learning-based technique. It has demonstrated remarkable efficiency in identifying and highlighting ROIs in lung CT scans automatically, improving diagnostic precision and alleviating radiologists' time constraints. As described in the work [2], it follows an encoder-decoder architecture, where the encoder (contracting path) uses convolution and max-pooling layers to reduce the image size and extract important features. The decoder (expanding path) then upsamples the features to restore the original image size and make pixel-wise predictions. By connecting corresponding layers from the encoder to the decoder, U-Net combines detailed low-level and high-level information for accurate segmentation results.

Even though UNet models outperformed in the medical and other areas, it is computationally very costly because UNet uses many feature channels, especially in the upsampling path, which requires a lot of memory and can be slow on large images or limited hardware. Therefore the speed of the entire training process can be affected.

To resolve this issue researchers have introduced different model compression techniques such as neural network pruning [9], low-rank factorizations of the weight matrices [10], quantization [11], knowledge distillation [12], neural architecture search [13], and other techniques. A model pruning is a most widely used technique for the model compression that can efficiently prune a DL model to reduce the number of redundant parameters. Therefore this method can be efficiently used to reduce size, computation time, memory, and other hardware requirements than an original DL model.

A model pruning is a technique of removing individual neurons/filters/channels/layers etc. from the Neural network model thereby reducing size of the model and making it more compact. There are different types of pruning: 1) unstructured pruning where weights of an individual neuron is removed or set to zero based on some criteria for example: remove all neurons whose weight magnitude is close to zero [14]. As a result, the model size may be reduced, but it may not yield actual performance improvements on standard hardware. 2) Structured pruning removes /filters/channels or even a layer. [15][16]. It can rebuild the model while preserving original model structure after pruning. This method does not need special hardware or software while pruning the model. 3) Semi-structured pruning removes small blocks of weights e.g., 2x2 or 4x4 groups, instead of individual weights.[17], more flexibility than structured pruning. However, structured pruning is better than unstructured pruning

because most software frameworks and hardware require real neural network acceleration and compression. The computation of sparse matrices cannot be accelerated.

Pruning process can be implemented as follows: 1) Prune before training the model [18] which is a faster method. But it can affect model performance greatly because it is yet to start learning. 2) Prune during training [19] of the model. With this method a model can achieve great accuracy but is a very complex method. 3) Prune after training [20] where we can first train the full model, then prune, followed by fine-tuning or retraining if needed. It is a very effective technique but costly as well.

While pruning the model there are different criteria available based on which pruning process can be performed: 1) magnitude based pruning where a neuron with smallest absolute value of its magnitude is considered as least important hence pruned. [21]. 2) L1/L2 norm [22] [23] where the smallest value of absolute values of weights in a filter/Square root of sum of squares of weights is the criteria for pruning. L1 norm preferred for structure pruning to remove individual filter/channel 3) saliency/sensitivity based pruning [24] measures change in loss function if a weight is removed making it data-dependent pruning and is computationally costly. 4) Loss change [25] using Taylor expansion method, nut estimate the impact of pruning on loss is costly again.

The remainder of the manuscript is organized as follows: Section II discusses related work found in the literature. Section III discusses the datasets and methods used to implement classification of the COVID-19, Magnitude based pruning. Section IV provides discussion on the results obtained, and Section V concludes the study with a discussion on the merits and limitations of the proposed approach and future work directions.

2. Literature Survey:

In a variety of computer vision and medical imaging tasks, deep learning models have demonstrated impressive performance. However, when implementing on edge devices or in real-time applications, their high memory consumption and computational expense pose difficulties. To overcome these obstacles, model compression strategies such as knowledge distillation, quantization, and pruning have been put forth. The ability to preserve hardware efficiency by pruning entire filters, channels, or layers rather than individual weights has made structured pruning stand out among the others.

This section of the presents a detailed literature review on structured pruning techniques in deep learning, highlighting the evolution of methodologies, pruning criteria, optimization strategies, and their applicability to convolutional neural networks (CNNs), including encoder-decoder architectures such as UNet.

The work proposed by [26] implemented filter-level pruning method where value of L1 norm of filter weights is used as an importance score for pruning. Filters with lower L1 norms are pruned, leading to structured sparsity while maintaining accuracy. He *et al.* [27] have introduced a data-driven approach where channels are pruned by minimizing the loss increase caused by their removal. Liu *et al.* [28] used scaling factors (γ) from batch normalization layers and applied L1 regularization to induce sparsity. Filters with low γ values were pruned, providing a simple yet effective structured pruning strategy. Molchanov *et al.* [29] have referred a Taylor expansion-based criterion for estimating the saliency of filters, focusing on first-order sensitivity to changes in the loss. Gao *et al.* [30] implemented

a dynamic approach where channels are adaptively activated during inference using learnable gates. This method improves runtime efficiency while preserving model capacity. The work done by Yu *et al.* [31] introduced a model that can adaptively adjust width (number of channels) at runtime. Though not traditional pruning, it enables structural compression via shared weights. Luo *et al.* [32] have used output feature map statistics to guide filter pruning, formulating the problem as a minimization of output reconstruction error. Ye *et al.* [33] presented discriminative-aware pruning, which takes into account a filter's significance for classification in addition to its norm, and questioned the L1/L2 norm assumptions. He *et al.* [34] Implemented reinforcement learning to automate the pruning policy search for structured pruning, focusing on mobile and edge deployment. [35] provided a principled framework for structured pruning using control theory to balance performance and efficiency. SCOP outperforms heuristics like magnitude-based pruning.

Structured pruning has advanced significantly. Early works focused on simplicity and ease of implementation and recent research emphasizes data-driven importance estimation and automation. These pruning techniques are highly relevant for large architectures like UNet, where channel or block-level redundancy can be substantial. Structured pruning ensures consistent acceleration, reduced memory usage, and hardware compatibility—making it a practical and powerful tool in real-world deep learning deployments.

The goal of the proposed work is 1) designs implemented as a base model which is then trained on lung CT with lung mask to detect infection region using semantic segmentation method on the dataset obtained from Kaggle [36].2) Apply structure pruning on trained model using uniform pruning rate across all layers and different pruning rate for each layer L1 norm based criteria 3) compare and analyze the impact of both pruning methods .

3. Research Methodology:

This proposed experiment identifies and locates infection regions in the lung CT data for COVID-19 diagnosis with the help of the custom UNet deep learning framework. UNet is a good choice for detecting anomalies connected to COVID-19 and separating lung and infection regions due to its capacity to precisely depict complex structures.

3.1 Data Acquisition

Proposed word is performed on the dataset [36] available nii-formatted containing CT scans with corresponding lung masks and infection masks of 20 patients. These scans of the left and right lungs are obtained, labeled by radiologists manually and confirmed by experts. A patient who tests negative for COVID-19 is indicated by the black mask. Sample CT slices with infection masks are ash shown below in Figure 1

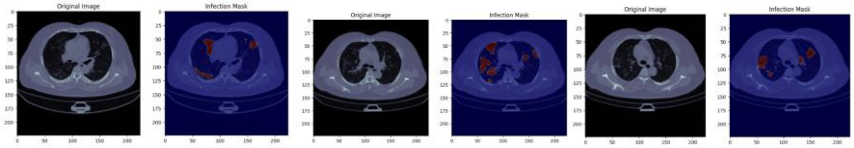

Figure 1: Sample images from the dataset

3.2 Data Preprocessing

All the grayscale images are resized to 224x224x1, some masks with NaN values and some duplicate slices are removed as part of data cleaning. All the CT image slices with their corresponding infection masks are processed to enhance contrast with using the most popular CLACHE method [**37**] that can significantly boost detection of ROI accuracy and cropping helps to remove unwanted portions of the CT like bones and tissues therefore focusing only on required lung region. Since dataset size is small 1613 slices, the conventional data augmentation techniques like flipping the original CT slice and its matching original masks up-down and left-right to increase size of dataset by 2016 slices to solve the overfitting problem during training of the DL model. The dataset is further divided into a training (70%) testing set (30%).

3.4 Model implementation and pruning

The proposed pruning work is divided and implemented into phases as given below:

3.4.1 Define Custom U-Net Model for Lung Infection detection and training: A custom U-Net model is implemented for detecting infection region in the lung CT slices with infection masks, from COVID-19 cases, using the pixel-wise [38]segmentation method. UNET model architecture is customized by incorporating batch normalization layers, dropout layers, attention gates, and increased depth. The output layer is classifying the images using sigmoid activation function. The accuracy of feature extraction and localization is enhanced by these changes in the model design, particularly for tiny or dispersed infection areas.

The model is then trained carefully by using different hyper parameters as given below: epochs = 50, batch_size = 16, Adam(learning rate = 0.00004) with learning scheduler to adjust learning rate [39], **BCE-Dice Loss** (Binary Cross-Entropy + Dice Loss) [40] which is a popular loss function for segmentation tasks, especially when using models like U-Net to monitor the performance of the model on training and testing. The experiment successfully achieved an impressive dice value of 91% and testing dice of 89%, with slight overfitting issue. Trained model is saved for further processing.

3.4 .2 Model Pruning using uniform structure pruning method: Load the trained model. Pruning after training approach is used in this phase to prune Custom UNet model using one shot, L1 norm based criteria with pruning percentage = 0.2 i.e. remove 20% of filters with lowest L1 norm**.** The L1 norm of a weight vector is the sum of the absolute values of the weights. Formula: L1 norm=∑i|wi|L1 norm=∑*i* |*wi*|, It encourages sparsity by driving some weights to exactly zero.

L1 norm is most widely used for structured pruning to prune e.g., filter/channel. A filter with small L1 norms has low overall activation, meaning they contribute less to the output feature map and have less impact on the model's performance, will be pruned. Since backpropagation is not required and one shot method [22] scores once and then prunes the network to a target prune ratio, computation cost will be less while preserving important filters and eliminating the least active ones. Bar graph in Figure 2 depicts layer wise filters before and after pruning and Fig 2 and 3 depicts average L1 norm calculated per layer.

Save and Fine tune the model to regain its performance because after pruning the model can degrade performance of the model. We have achieved the best training dice coefficient of 92.63% and

testing dice of 91.04 % after fine tuning with the same hyperparameter as in phase 3.4.1 after pruning the custom UNnet model.

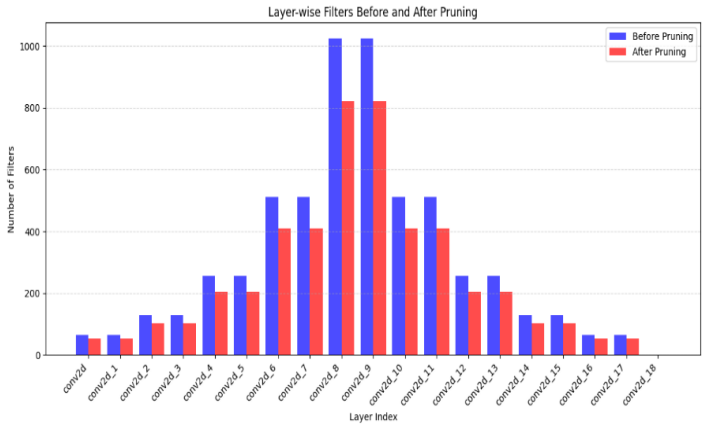

Figure 2: Layer wise comparison of L1 norm before and after pruning

Table 1 summarizes the impact of the uniform structure pruning rate on each layer of the trained custom Unet model.

Table 1: Effect of Pruning on different layers in custom Unet model

Layer(s)	Pruning Impact	Reason
conv2d_9	Most affected (largest L1-norm drop)	Redundant high-level feature filters removed
conv2d_8, conv2d_10	Highly affected	Mid-level layers optimized for efficiency
conv2d_6, conv2d_11	Moderately affected	Some high-level feature extraction preserved
conv2d_1 to conv2d_3	Least affected	Early layers crucial for texture and edge detection
conv2d_16 to conv2d_18	Least affected	Excessive pruning of final layers can degrade accuracy

3.4.3 Model Pruning with custom pruning rate: The analysis in Table 1 shows that when pruning proposed custom Unet model, uniform pruning rate is inappropriate. This analysis is used as a base information and a suitable pruning rate is determined for each layer separately to apply custom pruning on the UNet model trained in phase I. The proposed UNet model is therefore with variable rates as follows:

pruning_rates = { "conv2d_9": 0.50, # Most affected (High pruning)
"conv2d_8": 0.40, "conv2d_10": 0.40, # Highly affected
"conv2d_6": 0.30, "conv2d_11": 0.30, # Moderately affected
"conv2d_1": 0.20, "conv2d_2": 0.20, "conv2d_3": 0.20, # Least affected (early layers)
"conv2d_16": 0.20, "conv2d_17": 0.20, "conv2d_18": 0.20 # Least affected (final layers)}

The graph in figure 3 demonstrated L1 norm for each layer before and after pruning with custom pruning rate. Save and fine tune the pruned model with the same hyperparameters as in 3.4.1 and obtained best dice of 93.29% in training and 91.41% while testing with little overfitting. Complete experiment of segmentation model is implemented and tested on TensorFlow/Keras on google Colab with GPU for optimizing for pixel-wise segmentation accuracy.

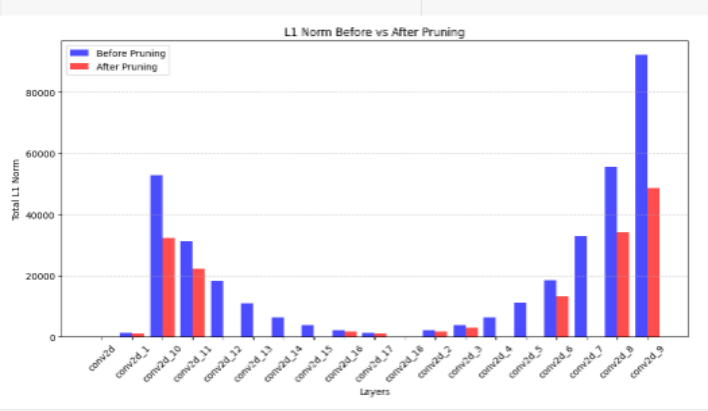

Figure 3: Average L1 Norm per Layer before and after custom Pruning

4. Results and Discussion:

By eliminating unnecessary or insignificant weights and neurons, model pruning is a popular method for lowering the size and complexity of DL models. The proposed experiment of defining a custom UNet model for ROI infection detection in lung CT is planned in 3 phases as follows.

In Phase I Structure pruning is performed on a custom UNet model as a base which is trained initially to recognize infection regions spread in lung CT with the help of lung infection mask. The model training dice is 0.91% and testing is 90% which is very impressive.

In phase II, Pruning After Training approach used in this experiment to have better control on the pruning process. Entire pruning is applied with uniform pruning rate of 20% for all the layers and removed all those filters whose L1 norm value smallest as shown in Figure 4:

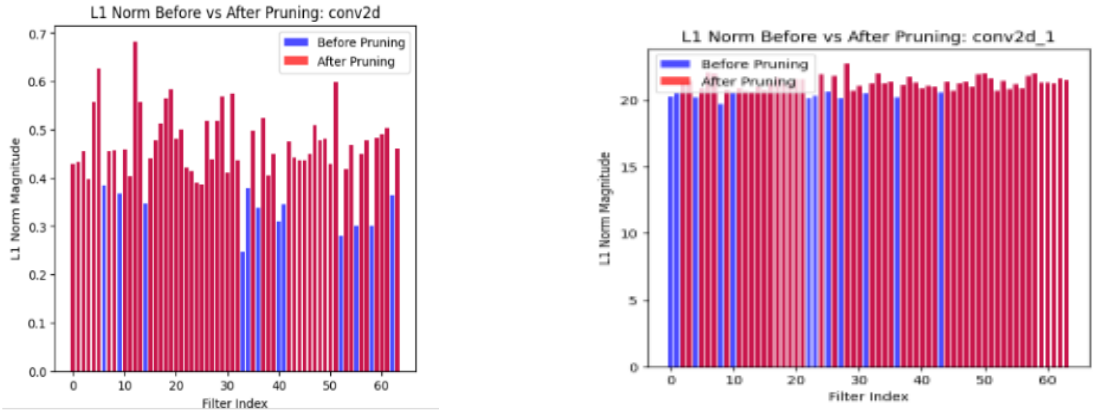

Figure 4: Filters pruned in first and second layer in with uniform pruning rate

Pruned model is retained with fine tuning using the same hyperparameters as in phase I and it was observed that performance of the model is improved with training dice of 92.63% and testing dice of 91.04%. Table 2 describing the impact of the uniform pruning across each layer on the custom UNet model.

The proposed experiment demonstrates the impact of the uniform pruning rate across layer in Custom UNet Model. Layer conv2d_9 is the most affected, which has the largest L1 norm that means more filters had large weights, making them good candidates for pruning. The reduction in L1 norm suggests that several filters were removed but not as aggressively as conv2d_9.

The proposed experiment demonstrates the impact of the uniform pruning rate across layer in Custom Unet Model. Layer conv2d_9 is the most affected, which has the largest L1 norm that means more filters had large weights, making them good candidates for pruning. The reduction in L1 norm suggests that several filters were removed but not as aggressively as conv2d_9.

Table 2: Impact of uniform pruning rate of 20% on each Layer

Layer	Pruning Impact	Observations from Graphs	Conclusion
conv2d_9	Most affected (high drop)	Significant L1-norm reduction; many low-impact filters removed	Successfully removed redundant high-level filters
conv2d_8, conv2d_10	Highly affected (moderate drop)	Filters with low contribution to mid-level feature extraction were pruned	Optimized mid-level layers for efficiency
conv2d_6, conv2d_11	Moderately affected	Some filters retained; moderate L1-norm reduction	Balanced pruning while preserving important features
conv2d_1, conv2d_2, conv2d_3	Least affected	Small L1-norm reduction; key edge and texture filters retained	Preserved early feature extraction layers

Middle layers (conv2d_6 to conv2d_12) contribute to high-level feature extraction and these layers extract abstract patterns rather than simple edges, making some filters redundant. Therefore these layers showed moderate pruning (~15-20% decrease). Structured pruning typically targets mid-network layers since they tend to have redundant filters that contribute less to final predictions. Early layers capture basic features (edges, textures), removing filters can degrade model performance. Also last layers contribute to final feature representations therefore excessive pruning can affect final predictions. As a result, Minimal pruning in conv2d_1 to conv2d_3 and conv2d_16 to conv2d_18. The pruned model is saved and fine-tuned to achieve reduction in the performance. The best dice value options are: training: 92.63% and testing: 91% with little overfitting problem which is little more than the model before pruning. Structured pruning successfully reduced L1 norm values across most layers, especially in the middle of the network. Filters in mid-network layers were pruned the most, potentially optimizing computational efficiency.

Phase III: By analyzing impact of uniform pruning on Custom UNet model with uniform pruning rate as described in the table 1, the experiment in Phase III is further performed to prune toe Custom Unet model defined in Phase I, by carefully planning for the layer wise pruning rate.

The Pruned model is trained again with fine tuning using the same hyperparameters as in phase I. The summary of the impact of the pruning on each layer is analyzed in table 3 and detailed analysis of the number of filters removed in the pruning process layerwise with sample graph is depicted in figure 5.

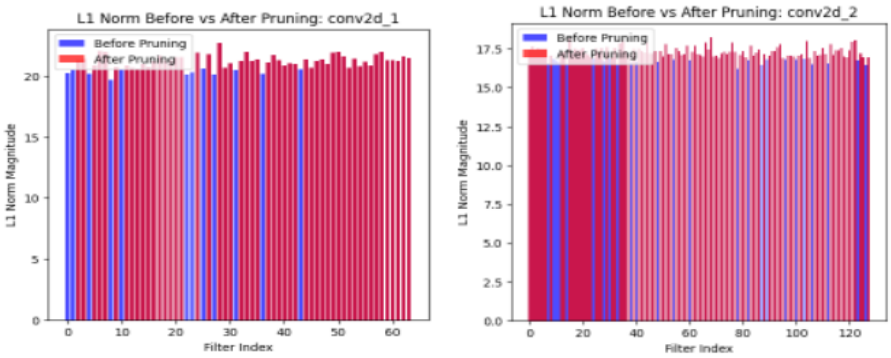

Figure 5: Filters pruned in first and second layer before and after custom Pruning

Table 3: Table 2: Impact of custom pruning rate on each Layer

Layer Group	Pruning Rate	Explanation
conv2d to conv2d_2	10%	Early layers detect basic visual patterns; aggressive pruning may drastically reduce accuracy.
conv2d_3 to conv2d_5	20%	Slightly deeper layers with more filters; can tolerate moderate pruning without major performance loss.
conv2d_6, conv2d_7	30%	Deeper layers encode abstract patterns; moderate capacity allows increased pruning to reduce model size.
conv2d_8, conv2d_9, conv2d_10	40–50%	Bottleneck (U-Net center) layers are over-parameterized and least sensitive; tolerate heavy pruning well.
conv2d_11	30%	Decoder start; benefits from pruning while maintaining balance with skip-connection information.
conv2d_12 to conv2d_14	20%	Output-refining decoder layers become more sensitive; moderate pruning preserves performance.
conv2d_15 to conv2d_18	10%	Final reconstruction layers require precision; light pruning avoids artifacts and poor localization.

All the models in the proposed experiment are implemented and tested using Tensor Flow/Keras on google Colab with GPU for optimizing for pixel-wise segmentation accuracy. It was observed that the result after custom pruning is increased by approximately by 1% in training and 0.41% for testing which can be further enhanced by more careful fine tuning by eliminating little overfitting issues, with results obtained are: dice coeff: 93.43% for training and testing dice_coeff:91.41 % which is little more than the original model. Detailed comparative result analysis of three proposed models is as shown in table 4. Table 5 demonstrates comparison of the prediction of ROI for infection detection by the proposed models in the lung CT both before and after pruning with dice value.

Table 4: Basic Deep Learning Architecture with its important features

dice value	Training	Testing	Dice Coeff. Improvement compared before
before pruning	91%	90%	-
After uniform pruning	92.63%	91.04%	Improved by 1.28%
After uniform pruning	93.43%	91.41%	Improved by 1.03%

Fine tuning the Custom UNet model after Pruning has slightly improved the model's performance in terms of Dice Coefficient on both training and testing datasets, without significantly increasing the inference time. This suggests that pruning can not only reduce the model size but also contribute to regularizing it and therefore improving models generalization ability.

Table 5: Comparison of dice value before, after uniform pruning (20%), and after custom pruning rate

Comparison of prediction of infection mask with dice value obtained from three different model		
Before pruning	**After 20%Pruning**	**After custom Pruning**

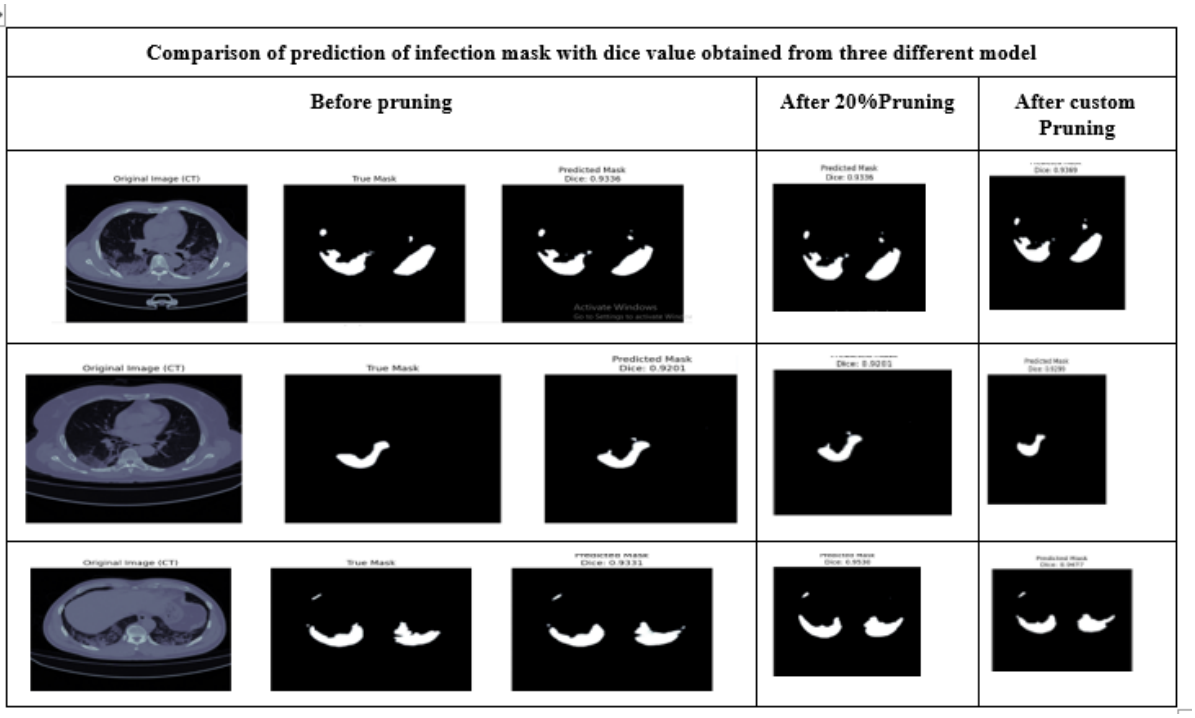

Figure 6: comparison of predicted mask before pruning, after uniform pruning rate of 20%, after custom pruning rate

Conclusion:

Detection of infection region in the lung CT is a crucial step in the treatment of COVID-19 cases. It equips medical workers with vital knowledge for precise diagnosis, treatment coordination, and patient care. The proposed Custom UNet model which is well trained to detect ROI with pruning using uniform structured and custom structured pruning approach has demonstrated improvement in the model's performance, achieving training Dice coefficient of 91 to 91.63 and further 93.43% on testing dice coefficient of from 90% to 91.04 then 91.41% respectively. Result may be further improved by training for more epochs and more careful hyperparameters to eliminate slight overfitting. Structured pruning effectively lowered L1 norm values, particularly in the network's middle layers. The L1 norm was significantly reduced through structured pruning, with the most substantial impact observed in the intermediate layers. Successful structured pruning led to a notable decrease in L1 norm values throughout the network, most prominently in its central sections.

Limitations and Potential Challenges

Despite the promising lung segmentation results, implementation faced limitations. The limited dataset size, primarily COVID-19 scans, potentially restricts generalization to diverse lung pathologies and raises overfitting risks given the model's capacity. Data preprocessing, while helpful, introduced

variability impacting cross-dataset consistency, and pruning could degrade performance without meticulous fine-tuning, demanding extra epochs and hyperparameter optimization. Furthermore, the Dice coefficient-centric evaluation might not fully capture all facets of model performance.

References:

1. World Health Organization. (n.d.). *COVID-19 situation in the WHO European Region*. Retrieved from https://www.who.int/europe/emergencies/situations/covid-19
2. Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional networks for biomedical image segmentation*. Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI).
3. Hu, S., Hoffman, E. A., & Reinhardt, J. M. (2001). Automatic lung segmentation for accurate quantitation of volumetric X-ray CT images. *IEEE Transactions on Medical Imaging, 20*(6), 490–498. https://doi.org/10.1109/42.929615
4. Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *Proceedings of the International Conference on Learning Representations (ICLR).*
5. Cheng, H., Zhang, M., & Shi, J. Q. (2024). A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 46*(12), 10558–10578.
6. Kumari, K. S., Samal, S., Mishra, R., *et al.* (2022). Diagnosing COVID-19 from CT image of lung segmentation and classification with deep learning based on convolutional neural networks. *Wireless Personal Communications, 127*, 2483–2499. https://doi.org/10.1007/s11277-021-09076-w
7. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of MICCAI.*
8. Cheng, H., Zhang, M., & Shi, J. Q. (2024). A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 46*(12), 10558–10578.
9. Ashkboos, S., Croci, M. L., Gennari, M., *et al.* (2024). SliceGPT: Compress large language models by deleting rows and columns. *Proceedings of the International Conference on Learning Representations (ICLR).*
10. Denton, E., Zaremba, W., Bruna, J., LeCun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in Neural Information Processing Systems (NeurIPS), 27*, 1269–1277.
11. Shao, W., Chen, M., Zhang, Z., *et al.* (2024). OmniQuant: Omnidirectionally calibrated quantization for large language models. *Proceedings of the International Conference on Learning Representations (ICLR).*
12. Xu, X., Li, M., Tao, C., *et al.* (2024). A survey on knowledge distillation of large language models. *arXiv preprint*, arXiv:2402.13116.

13. Zhang, M., Su, S., Pan, S., Chang, X., Abbasnejad, E., & Haffari, R. (2021). iDARTS: Differentiable architecture search with stochastic implicit gradients. *Proceedings of the International Conference on Machine Learning (ICML).*
14. Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems (NeurIPS), 28*.
15. Ashkboos, S., Croci, M. L., Gennari, M., *et al.* (2024). SliceGPT: Compress large language models by deleting rows and columns. *Proceedings of the International Conference on Learning Representations (ICLR).*
16. You, Z., Yan, K., Ye, J., Ma, M., & Wang, P. (2019). GateDecorator: Global filter pruning method for accelerating deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS).*
17. NVIDIA Developer Blog. (n.d.). *Accelerating sparse deep neural networks*. Retrieved from https://developer.nvidia.com/blog/accelerating-sparse-deep-neural-networks/
18. Tanaka, H., Kunin, D., Yamins, D. L., & Ganguli, S. (2020). Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems (NeurIPS), 33*, 6377–6389.
19. Liu, S., Chen, T., Atashgahi, Z., *et al.* (2022). Deep ensembling with no overhead for either training or testing: The all-round blessings of dynamic sparsity. *Proceedings of the International Conference on Learning Representations (ICLR).*
20. Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2019). Rethinking the value of network pruning. *Proceedings of the International Conference on Learning Representations (ICLR).*
21. Hanson, S., & Pratt, L. (1988). Comparing biases for minimal network construction with back-propagation. *Advances in Neural Information Processing Systems (NeurIPS), 1*.
22. Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient convnets. *Proceedings of the International Conference on Learning Representations (ICLR).*
23. He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2234–2240.
24. LeCun, Y., Denker, J., & Solla, S. (1989). Optimal brain damage. *Advances in Neural Information Processing Systems (NeurIPS)*, 598–605.
25. Santacroce, M., Wen, Z., Shen, Y., *et al.* (2023). What matters in the structured pruning of generative language models? *arXiv preprint*, arXiv:2302.03773.
26. Li, Y., Kadav, S., Duranovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient ConvNets. *Proceedings of the International Conference on Learning Representations (ICLR).*
27. He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1398–1406.
28. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2736–2744.

29. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., & Kautz, J. (2019). Importance estimation for neural network pruning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 11264–11272.
30. Gao, H., Zhao, H., Liao, S., Li, Z., Luo, H., & Gao, Y. (2019). Dynamic channel pruning: Feature boosting and suppression. *Proceedings of the International Conference on Learning Representations (ICLR).*
31. Yu, J., Yang, L., Xu, N., Yang, J., & Huang, T. (2019). Slimmable neural networks. *Proceedings of the International Conference on Learning Representations (ICLR).*
32. Luo, J., Wu, J., & Lin, W. (2017). ThiNet: A filter-level pruning method for deep neural network compression. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 5058–5066.
33. Ye, S., Zhan, K., Zhang, Y., & Loy, C. C. (2019). Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 6656–6664.
34. He, Y., Lin, J., Liu, Z., Wang, H., Li, L., & Han, S. (2018). AMC: AutoML for model compression and acceleration on mobile devices. *Proceedings of the European Conference on Computer Vision (ECCV)*, 815–832.
35. Tang, Y., Hua, G., & Wang, Y. (2020). SCOP: Scientific control for reliable neural network pruning. *Advances in Neural Information Processing Systems (NeurIPS).*
36. Andrewmvd. (n.d.). *COVID-19 CT scans dataset*. Kaggle. Retrieved from https://www.kaggle.com/datasets/andrewmvd/covid19-ct-scans
37. Zuiderveld, K. (1994). Contrast limited adaptive histogram equalization. In *Graphics Gems IV* (pp. 474–485). Academic Press.
38. Fan, D.-P., Zhou, T., Ji, G.-P., *et al.* (2020). Inf-Net: Automatic COVID-19 lung infection segmentation from CT scans. *IEEE Transactions on Medical Imaging, 39*(8), 2626–2637.
39. Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.-M., & Larochelle, H. (2017). Brain tumor segmentation with deep neural networks. *Medical Image Analysis, 35*, 18–31.
40. Konar, J., Khandelwal, P., & Tripathi, R. (2020). Comparison of various learning rate scheduling techniques on convolutional neural networks. *Proceedings of the IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 1–5. https://doi.org/10.1109/SCEECS48394.2020.94
41. Xue, Y., Xu, T., Ma, L., Zhang, T., Gao, S., & Zhang, C. (2018). Liver tumor segmentation in CT volumes using an adversarial image-to-image network. *IEEE Access, 6*, 57999–58009.
42. Kang, J. M., Jang, J., & Ro, Y. M. (2021). Automatic liver and tumor segmentation of CT and MRI volumes using cascaded fully convolutional neural networks. *Sensors, 21*(4), Article 1162.