# Pruning Strategies and Their Role in Transfer Learning

Aakash Agarwal
University of Pennsylvania
Philadelphia, USA
aakash24@seas.upenn.edu

Saaransh Pandey
University of Pennsylvania
Philadelphia, USA
saaransh@seas.upenn.edu

## Keywords

Pruning, Iterative Pruning, Model Compression, Transfer Learning

**ACM Reference Format:**
Aakash Agarwal and Saaransh Pandey. 2024. Pruning Strategies and Their Role in Transfer Learning. In *Proceedings of Hardware Software Co-Design for Machine Learning (ESE 539).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Project Summary

This study explores and evaluates various pruning techniques to optimize deep learning models for resource efficiency while maintaining performance. Using a pretrained VGG16 model on the ImageNet validation dataset, we analyze global pruning, layer-wise unstructured pruning, layer-wise channel pruning, and hardened channel pruning. Each method employs the L1 norm to identify and prune the least important weights or channels at sparsity levels ranging from 5% to 45%. These methods are compared in terms of top-1 accuracy, runtime, and model size to understand the trade-offs between computational efficiency and performance.

The study further investigates iterative pruning, where pruning and fine-tuning are repeated across multiple cycles to maximize sparsity while recovering lost accuracy. This iterative approach aims to push model compression limits without significant accuracy degradation.

Additionally, we extend the analysis to transfer learning by fine-tuning pruned models on the melanoma cancer dataset. This demonstrates the adaptability models to new tasks with iterative pruning, assessing their potential for real-world applications in computationally constrained settings.

The overall study highlights the balance between sparsity, accuracy, and efficiency, offering insights into pruning strategies for deployment in diverse scenarios, including transfer learning.

## 2 Methodology

### 2.1 Global Weight Pruning

Global weight pruning is a model compression technique that reduces the size and complexity of neural networks by removing the least important weights across the entire network. It evaluates weight importance using a metric such as the $L1$ norm (absolute magnitude) and globally identifies the least significant weights to prune, regardless of their layer. These weights are initially masked, meaning their contributions are set to zero during computations while the original weights are temporarily stored.

In PyTorch, *prune.global_unstructured* applies binary masks to the specified layers based on a pruning criterion like $L1Unstructured$. To finalize the process, $prune.remove()$ is used to remove the mask and hooks, leaving the pruned weights as permanent zeros in the weight tensor and eliminating additional computational overhead.

Global pruning has shown promise in achieving high levels of sparsity without significantly sacrificing performance, making it particularly useful for deploying deep learning models in resource-constrained environments. In this study, our aim is to evaluate the impact of global weight pruning on a pretrained VGG16 model across various sparsity targets, analyze the trade-off between sparsity and top-1 accuracy, and identify the thresholds at which performance begins to degrade significantly.

### 2.2 Layer-wise Weight Pruning

Layer-wise weight pruning is a model compression technique that removes the least important weights independently within each layer of a neural network. By evaluating weights in each layer based on their significance, typically determined using metrics like the $L1$ norm (absolute magnitude), the method ensures that sparsity is distributed uniformly across the network. This approach is particularly effective for preserving the proportional contribution of each layer to the overall performance, as it avoids over-pruning smaller layers.

The pruning process involves iterating through the layers of the network, applying a specified pruning strategy, and removing the pruned weights. For example, in PyTorch, the $prune.l1\_unstructured$ function is applied to each layer, targeting a specified percentage of weights to prune ($e.g.$, 5%, 10%). After pruning, $prune.remove()$ is used to finalize the process by removing the pruning masks and permanently setting the pruned weights to zero. This ensures that the pruned model is ready for evaluation without additional computational overhead.

Layer-wise pruning allows for precise control over sparsity in individual layers, which can result in better retention of performance for certain network architectures. It ensures that all layers retain sufficient capacity to contribute to the overall task, reducing the risk of performance degradation that may occur with unbalanced sparsity. In this study, the aim is to analyze the impact of layer-wise pruning on the accuracy of a pretrained VGG16 model, evaluate its performance across various sparsity levels.


*ESE 539, 2024, Philadelphia, PA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 2.3 Layer-wise Output Channel Pruning

Layer-wise output channel pruning is a structured model compression technique that removes entire output channels from convolutional layers or dimensions from fully connected layers, rather than individual weights. By pruning output channels, this method reduces both the computation and memory requirements of the network while maintaining the structure of the weight tensors. Unlike unstructured pruning, which sparsifies individual weights within a layer, channel pruning focuses on removing redundant feature maps (channels) in a systematic manner.

The process involves applying pruning algorithms like PyTorch's *prune.ln_structured* to target specific output channels. For convolutional layers, pruning is performed along the output channel dimension, while for fully connected layers, it targets the output dimensions. To ensure compatibility, the input channels of each subsequent layer are automatically adjusted to match the pruned output channels of the preceding layer. The last linear layer, responsible for predicting classes, is excluded from pruning to preserve the output dimensions of the network. The pruning percentages can be incrementally increased ($e.g.$, 5%, 10%) to evaluate the effect of sparsity on model performance. After pruning, the masks are removed to finalize the channel pruning.

To further optimize the pruned model, the channel pruning can be "hardened" by instantiating a new model architecture with the reduced number of output channels in each layer. The non-zero weights from the pruned model are copied over to the new hardened model, ensuring a compact architecture without unnecessary computations. Special attention is required to align the input and output channels between consecutive layers, particularly at the interface between the last convolutional layer and the first fully connected layer. Hardened models provide an accurate representation of the pruned network, enabling direct comparison of runtime and size with the original model.

Layer-wise output channel pruning has the advantage of structured sparsity, which often results in more efficient computation compared to unstructured methods. By eliminating entire channels, it directly reduces the computational cost and memory footprint, making the network suitable for resource-constrained environments. In this study, the aim is twofold: first, to evaluate the impact of channel pruning on the accuracy of a pretrained VGG16 model and compare it to unstructured pruning, and second, to analyze the trade-offs between accuracy, runtime, and model size for the hardened channel-pruned models across various sparsity levels.

## 2.4 Iterative Pruning

Iterative pruning is a technique that is widely employed for model compression and optimization in deep learning. It involves progressively reducing the size of the neural network by identifying and removing less significant parameters in a series of iteration while maintaining the overall performance of the network. At each iteration a fraction of the least important parameters, which are determined by their magnitude are set to zero. After each pruning step the model is fine-tuned on the training dataset to recover any loss in performance caused by the removal of parameters. This way the network is able to learn the underlying distribution of the data using less parameters than before. Finally this process is repeated iteratively until a target sparsity level is achieved or the network's performance begins to degrade significantly. This methodology achieves significant model compression and at the same time preserves the critical features of the original network, making iterative pruning an effective tool for deploying deep learning models in resource-constrained environments.

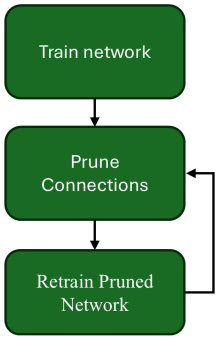


**Figure 1: Three Stage Iterative Training/Pruning Pipeline**

## 2.5 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is repurposed as the foundation for a second task. Leveraging learned features from the initial task, the model can adapt more efficiently to the new task, accelerating learning and improving performance. The need for training from scratch is eliminated in this scenario which shortens training times and conserves resources by utilizing existing models.

Transfer learning involves a structured process to leverage existing knowledge from a pre-trained model for new tasks:

- Begin with a model that has already been trained on a large dataset for a particular task. This pre-trained model has acquired general features and patterns that are applicable to similar tasks. .
- The pre-trained model, referred to as the base model, contains layers that have processed data to learn hierarchical representations, capturing everything from basic to more complex features.
- A new classification layer, Fully Connected (FC) layer, is added at the end of the pre-trained model.
- This last layer is then fine-tuned with the data from the new task. This process helps retain the pre-trained knowledge while adjusting parameters to meet the specific requirements of the new task, improving accuracy and adaptability.

The aim of this work is to explore the applicability of iterative pruning in the context of transfer learning. We experiment with two kinds of setup. In Setup 1, as shown in Fig. 2 , we use the baseline model as the VGG16 model, we add a FC layer to it and then run iterative pruning on our local task (transfer learning). The drawback of this method is that we will need to iteratively retrain/prune on the local task every time. This could be computationally expensive if the number of downstream tasks are high. Another setup that we experiment is shown in Fig. 3. In this setup we create a pruned

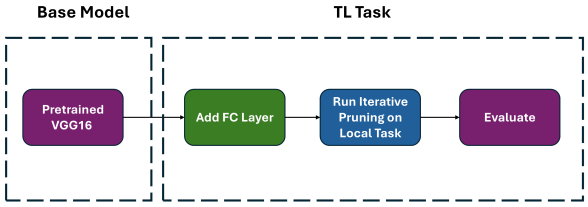


**Figure 2: Setup 1 for Transfer Learning**

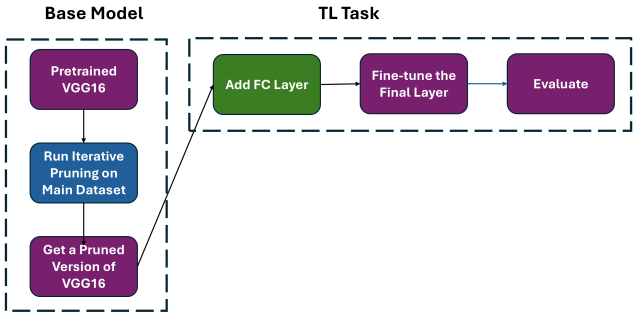


**Figure 3: Setup 2 for Transfer Learning**

version of the main baseline model by iteratively pruning/retraining on the main dataset. We then directly use this pruned version of the baseline model, add a FC layer to it, fine-tune this Final Layer for some epochs, and evaluate the model. We just need to do iterative pruning once on the main model. This alleviates the computational complexity of doing iterative pruning for each local task. We present the dataset that we use for transfer learning and the results associated with it in the next sections.

## 3 Dataset Preparation

### 3.1 Evaluation of Pruning Techniques

Initially, for evaluating and comparing different pruning techniques, we utilize the validation set of the ImageNet dataset [1]. This set comprises 50,000 images spanning 1,000 diverse categories, providing a robust benchmark for assessing model performance across various pruning methods.

### 3.2 Iterative Pruning

Due to lack of computational resources, we use just the validation subset of the complete Imagenet Dataset. We split this dataset into two parts into 90% and 10%, for training and testing purposes.

### 3.3 Transfer Learning.

For transfer learning we choose the open source Melanoma Skin Cancer Dataset [2]. The dataset consists of 10605 images with 3 colour channels belonging to two classes, benign and malignant. Out of these 10605 images, we use 9605 images (4605 - malignant, 5000 - benign ) for training and 1000 images (500 - malignant, 500 - benign) for testing purpose. Fig. 4 shows some samples from the dataset. The first row shows benign class and the second row shows the malignant class.

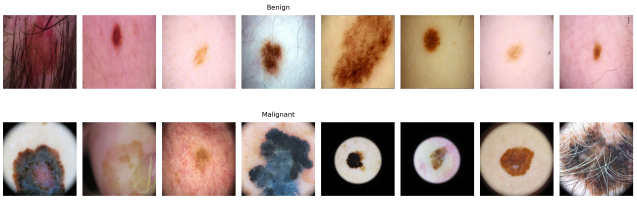


**Figure 4: Melanoma Cancer Dataset**
a) Benign b) Malignant

## 4 Experimental Setup

All the implementations are done in a Kaggle environment with 16 Gb T4 GPU and 30 Gb CPU RAM. The fine tuning for transfer learning is done on an Apple M2 Macbook Pro with 16 GB of unified memory. For training the VGG16 we make use of Stochastic Gradient Descent optimizer with a learning rate of 0.001 and momentum of 0.9, unless otherwise stated. Training one epoch on Imagenet Dataset takes about 15 minutes, whereas inferencing on validation dataset takes approximately a minute. We use L1-norm based pruning method for pruning of weights. In iterative pruning section, we do both layer-wise pruning and layer-wise channel pruning. In transfer learning section, we only do layer-wise pruning.

## 5 Evaluation

In the following subsections we discuss about the results that we got by implementing various methods menetioned talked about in the paper.

### 5.1 Global Weight Pruning

Here, we apply global weight pruning to a pretrained VGG16 model using the L1 norm to identify and prune the least important weights across all convolutional layers. The pruning is performed at sparsity levels ranging from 5% to 45%, incremented in steps of 5%. The pruned models are evaluated on the ImageNet validation dataset, and the performance is measured using the top-1 accuracy metric. The original VGG16 model serves as the baseline for 0% pruning to compare the impact of pruning on model accuracy.

In Fig 5, we can see the top-1 accuracy remains close to the original VGG16 model's accuracy upto 30% pruning. This suggests that less significant weights are successfully removed without affecting model performance significantly.

Beyond 35% pruning, although the accuracy drops noticeably, with a decline at 40% and 45%, it still maintains a relatively high accuracy around 70% at 45% pruning. This indicates that while some critical weights contributing to the model's performance are being pruned, the loss in accuracy remains moderate, demonstrating the model's resilience to pruning even at higher sparsity levels.

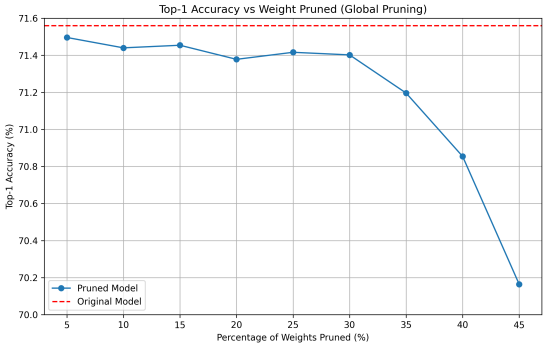


**Figure 5: Plot for Top-1 Accuracy vs Weights Pruned via Global Weight Pruning Method**

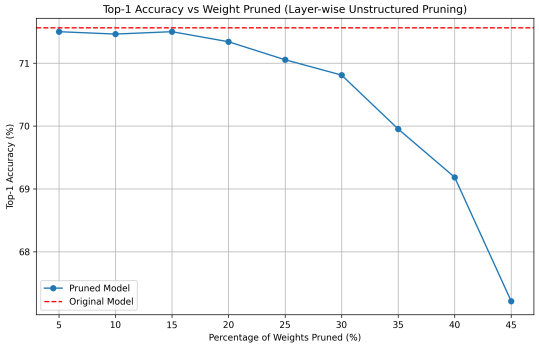


**Figure 6: Plot for Top-1 Accuracy vs Weights Pruned via Layer-wise Weight Pruning Method**

## 5.2 Layer-wise Weight Pruning

Here, we apply layer-wise unstructured pruning to a pretrained VGG16 model using the L1 norm to prune the least important weights within each layer independently. The pruning is performed at sparsity levels ranging from 5% to 45%, incremented in steps of 5%. The pruned models are evaluated on the ImageNet validation dataset, and the performance is measured using the top-1 accuracy metric. The original VGG16 model serves as the baseline for 0% pruning, allowing for a comparison of the impact of layer-wise unstructured pruning on model accuracy.

In Fig 6, we can see the top-1 accuracy remains close to the original VGG16 model's accuracy upto 15% pruning. This indicates that layer-wise pruning successfully removes less significant weights within each layer without affecting performance noticeably.

From 20% to 30% pruning, the accuracy begins to drop steadily. This suggests that as more weights are pruned within each layer, the layers start losing their representational capacity, impacting the model's overall performance.

Beyond 30% pruning, the accuracy decline becomes more pronounced, with the top-1 accuracy dropping at 45% to around 67%. This shows that layer-wise unstructured pruning can impact critical weights within individual layers, leading to reduced performance.

Layer-wise unstructured pruning retains accuracy better than global pruning at lower sparsity levels (up to 15%), as it ensures balanced pruning across layers without over-pruning smaller or sensitive layers. However, as pruning increases beyond 20%, accuracy begins to drop steadily, and the decline becomes more pronounced at higher levels (above 35%), with significant performance loss at 45%. This is because critical weights within individual layers may still be pruned. In contrast, global pruning achieves a more balanced sparsity across the entire model, making it more effective at higher pruning rates. Overall, layer-wise unstructured pruning performs better at low sparsity but worse than global pruning at higher levels due to its lack of global weight importance consideration.

## 5.3 Layer-wise Output Channel Pruning

Here, we perform layer-wise output channel pruning on a pretrained VGG16 model using the L1 norm to prune entire output channels from convolutional layers and output dimensions from linear layers. The pruning is applied at sparsity levels ranging from 5% to 45% in increments of 5%, while preserving the last linear layer to maintain the number of predicted classes. The pruned models are evaluated on the ImageNet validation dataset using the top-1 accuracy metric, with the original VGG16 serving as the baseline for 0% pruning. This allows us to analyze the effect of layer-wise channel pruning on model accuracy and compare its performance to layer-wise unstructured pruning.

In Fig 7, we can see the relative top-1 accuracy drops sharply even at low pruning percentages (5% – 10%) compared to the original model. Beyond 10% pruning, the accuracy declines drastically, and the model loses most of its predictive capacity by 25%, with negligible accuracy at higher pruning levels (30% – 45%).

Layer-wise channel pruning performs worse than layer-wise unstructured pruning because removing entire output channels severely impacts the network's representational capacity compared to removing individual weights. Output channels correspond to entire feature maps, and their removal eliminates critical information processed by the layer. While channel pruning provides structured sparsity, improving model efficiency and inference speed, it sacrifices accuracy significantly, especially without retraining, as key features essential for decision-making are lost. In conclusion, layer-wise channel pruning negatively affects performance by aggressively reducing the model's ability to extract meaningful features.

Later, we harden the layer-wise channel pruning by instantiating a new model with updated architecture based on the number of non-zero output channels in the pruned model. Adjustments are made to ensure compatibility between the output channels of one layer and the input channels of the next. The non-zero weights from the pruned model are copied into the hardened model. We then evaluate the runtime, model size, and relative top-1 accuracy of the hardened models and compare them to the original VGG16. This allows us to explore the trade-offs between accuracy, runtime, and model size at different levels of pruning.

In Fig 8, we can see:

- The relative top-1 accuracy drops sharply even at low pruning rates (5% – 10%) and becomes negligible beyond 15%

pruning. This indicates that hardened channel pruning significantly affects the model's ability to retain meaningful features, as entire channels are permanently removed without retraining.

- The relative runtime decreases gradually as pruning increases, showing that hardened channel pruning improves inference efficiency. At 45% pruning, the runtime reduces to nearly 76% of the original, demonstrating the benefits of structured sparsity.
- The relative model size decreases linearly with the pruning rate, reaching less than 60% of the original size at 45% pruning. This reflects the direct removal of pruned channels, which reduces storage requirements effectively.

Hardened channel pruning significantly reduces model size and runtime, offering substantial efficiency gains that make it well-suited for deployment in resource-constrained environments. However, this comes at the cost of a sharp decline in accuracy, particularly beyond 15% pruning, making the model unsuitable for tasks requiring high predictive performance unless retraining is performed. This highlights a clear trade-off between computational efficiency and model performance, where the acceptability of accuracy loss depends on the specific use case.

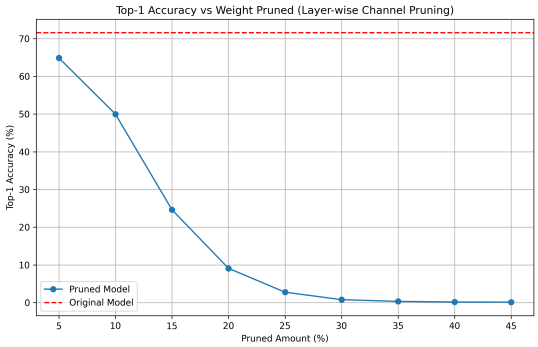


**Figure 7: Plots for Top-1 Accuracy vs Weights Pruned via Layer-wise Channel Pruning Method**

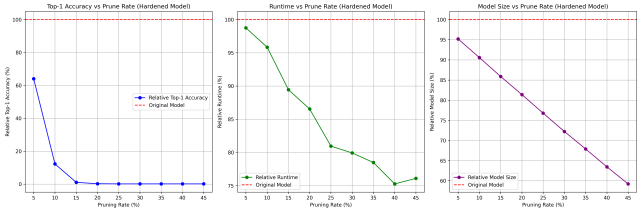


**Figure 8: Plots for evaluating (a) Top-1 Accuracy (b) Runtime (c) Model size vs Weights Pruned via Layer-wise Channel Hardened Pruning Method**

## 5.4 Comparison of Pruning Techniques

This analysis demonstrates the effectiveness and limitations of different pruning techniques for model compression, as illustrated in the Fig. 9. Global weight pruning preserves accuracy even at higher pruning levels due to its balanced weight removal strategy. Layer-wise pruning shows improved accuracy retention at lower sparsity levels by respecting individual layer dynamics but becomes less effective as pruning increases. In contrast, layer-wise channel pruning and hardened pruning sacrifice accuracy significantly, especially without retraining, as entire feature maps are removed, impacting the model's representational capacity.

Hardened pruning achieves notable reductions in model size and runtime, making it ideal for resource-constrained environments where computational efficiency outweighs accuracy. However, it highlights the importance of retraining to restore performance.

The findings underscore that the choice of pruning technique depends on the specific application. For high accuracy retention, global pruning is preferable, while for efficiency in deployment, hardened pruning offers substantial benefits. These insights pave the way for future exploration of retraining methods and transfer learning to maximize the utility of pruned models across diverse tasks.

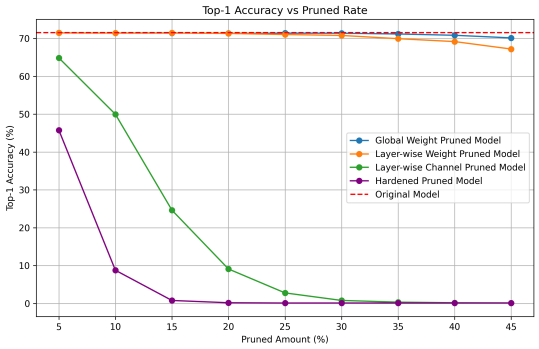


**Figure 9: Plots for comparing Top-1 Accuracy of all the pruning methods**

## 5.5 Iterative Pruning

In this section we present the detailed results from iterative pruning on VGG16 model on Imagenet dataset.

We prune 25% of the weights of each layer based on their L1 norm, in every iteration. This is called layer-wise pruning. After each prune iteration, we train the network again for 7 epochs. This way the network redistributes the weights values and tries to learn the underlying pattern in the dataset with fewer number of weights.

Fig. 10 shows the variation in validation accuracy with percentage of weights pruned in each iteration. The red dotted line shows the baseline validation accuracy (71.78%) which is calculated using the pretrained VGG16 model. We can observe (purple line) that we can throw away 58% of weights for virtually no loss in accuracy. The green line shows the validation accuracy when we do no retrain the network. The gap between retrained and not retrained

network starts to widen after we prune 58% of weights which proves retraining is necessary to regain accuracy.

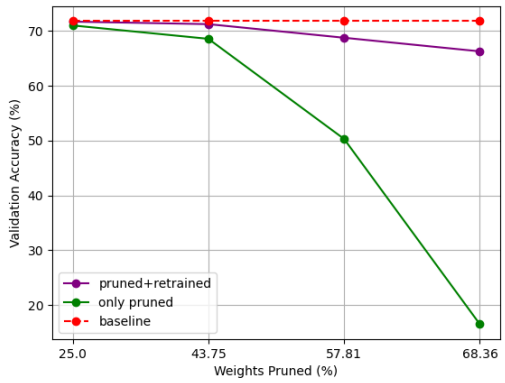


**Figure 10: Validation Accuracy vs Percentage of Layer-Wise Pruned Weights for VGG16 on ImageNet dataset.**

We also implement layer-wise channel pruning where we prune away the whole channel of the layer. In each iteration based prune away 10% of the channel that have already not been pruned. We choose these channels based on the L1-norm. The results are presented in Fig 11. We see (green line) that after 10% the model loses all of its predictive capabilities with validation accuracy dropping to almost zero. We see that retraining such highly pruned model help a lot in recovering the validation accuracy. As seen by the purple curve, with retraining model can restore a lot of its predictive capability back till 47.8%. This shows that retraining is absolutely necessary on such highly pruned network.

Figure 12 shows the distribution of weights of original VGG16 pretrained model and our iteratively pruned model using Layer-wise Pruning. We see that the model weights before and after retraining are quite similar. This is due to the fact that we have already trained the model enough that it has learnt the underlying distribution of the data. We will see in the next subsection that

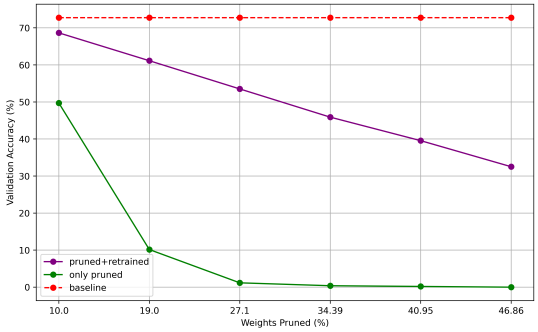


**Figure 11: Validation Accuracy vs Percentage of Layer Channel Wise Pruned Weights for VGG16 on ImageNet dataset.**

weight distribution changes after retraining when the weights are initialized randomly.

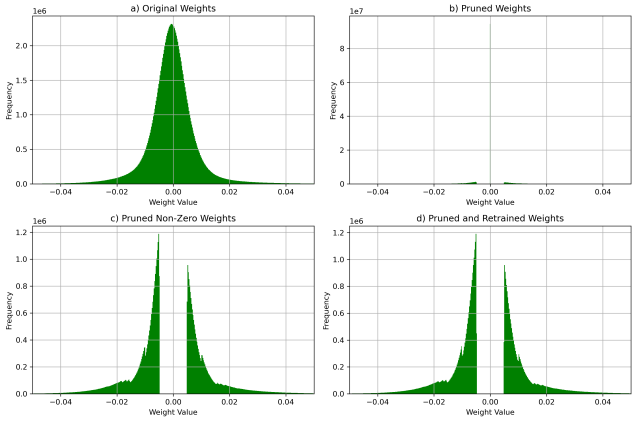


**Figure 12: Weight Distribution of, (a) Original VGG16 model (b) Pruned VGG16 (c) Non-Zero Weights of Pruned VGG16 (d) Pruned and Retrained VGG16, on ImageNet Dataset with Layer-wise Pruning**

Fig. 13 shows the weight distribution of the original model and our iteratively pruned model using Layer-wise Channel Pruning.

## 5.6 Transfer Learning

In this section we present the results of iterative pruning in transfer learning. In transfer learning, we freeze the weights of original model and add a fully connected (FC) layer in the end of the network which acts as a classifier layer. This last layers is then fine tuned on the melanoma dataset.

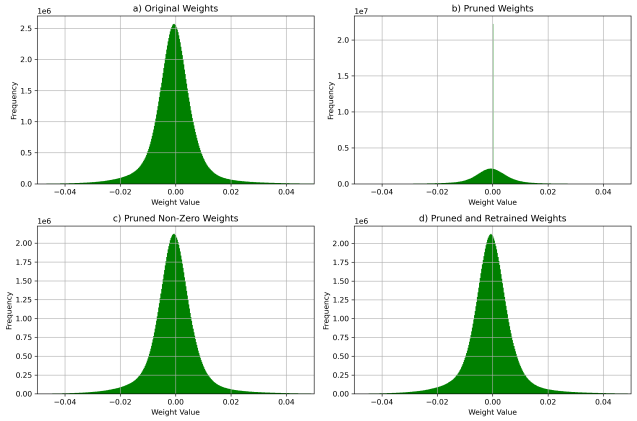


**Figure 13: Weight Distribution of, (a) Original VGG16 model (b) Pruned VGG16 (c) Non-Zero Weights of Pruned VGG16 (d) Pruned and Retrained VGG16, on ImageNet Dataset with Layer-wise Channel Pruning.**

## 5.7 Setup 1

In this setup, iterative pruning is done by pruning away 10% weights in every iteration and then retraining the model for 10 epochs. We use Stochastic Gradient Descent Optimizer with a learning rate of 0.001. These 10% weights are selected such that their L1 norm is the lowest. We keep doing this until the validation accuracy is within 5% of the baseline VGG16 validation accuracy.

Fig. 14 shows the trend in the validation accuracy as we increase pruning. The red line is the baseline accuracy (84.6%) that we get without pruning of the network. We see (purple line) that the model is able to recover accuracy after each iteration till almost 80% weights are pruned. This shows that we can use over five times (x5) lesser weights with minimal loss in validation accuracy. We also plot the validation accuracy achieved using only a pruned model. We see that till 68.62% pruning, both retrained and not retrained models perform equally well. This suggests that the model is overly parametrized and 68.62% of the weights are not even useful. The need to retrain begins after that as the gap widens, which suggests that pruning further removes meaningful connections as well and the neural network needs to be retrained to learn those patterns.

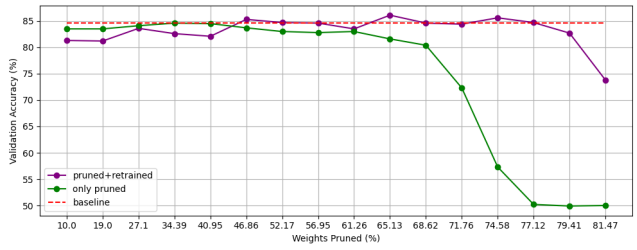


**Figure 14: Validation Accuracy vs Number of Weights Pruned in setup 1.**

Fig. 15 (a) shows the weight distribution of all the layers of the original VGG16 model. Fig. 15 (b) shows the weight distribution of the pruned network. We can notice there is a high density of weights whose values are zero. This becomes clear from Fig. 15 (c) where almost 80% of weights whose values are closer to zero are pruned away (i.e., made zero). After retraining the weights are redistributed to gain any loss in accuracy due to pruning, as seen in Fig 15 (d).

We also investigate the weight distribution of the last fully connected layer before and after training/pruning. Fig. 16 (a) shows that the weights of the last FC layer are initialized uniformly. This is a randomly chosen initialization and hence the weights contain no information about the patterns in the dataset. In the first iteration we prune 25% of the weights which is shown in Fig. 16 (b), (c). Specifically, (b) also shows the number of weights which have values 0. These are the weights that we have pruned. Since the weights are not trained on the dataset, they are still almost uniformly distributed as in (c). After training, as seen is Fig. 16 (d) the weights are distributed in a Gaussian-like distribution which shows that they now learn the underlying pattern of the data.

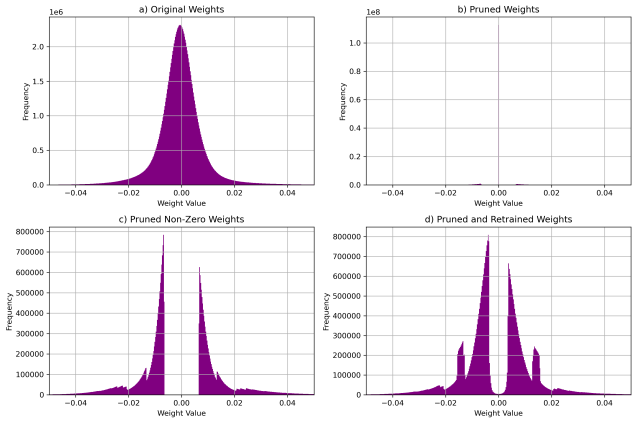


**Figure 15: Weight Distribution of (a) Original VGG16 model (b) Pruned VGG16 (c) Non-Zero Weights of Pruned VGG16 (d) Pruned and Retrained VGG16**

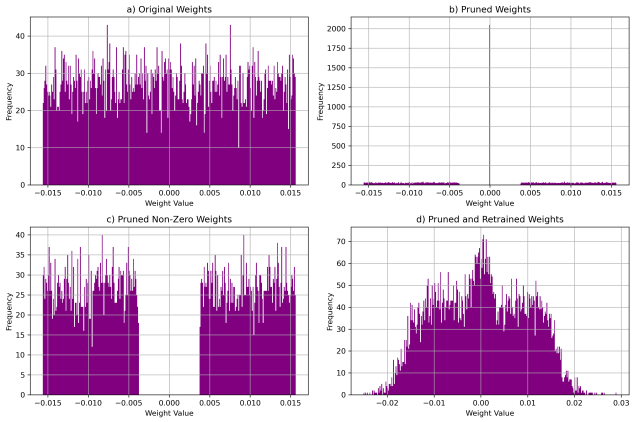


**Figure 16: Weight Distribution of Last Layer of, (a) Original VGG16 model (b) Pruned VGG16 (c) Non-Zero Weights of Pruned VGG16 (d) Pruned and Retrained VGG16, on Cancer Dataset**

## 5.8 Setup 2

In this setup we use an already pruned version of the VGG16 model, add a FC layer at the end of it and fine tune that on the melanoma dataset. We present the results for different percentage of pruning. Fig. 17 shows the trend in the validation accuracy as we increase the prune percentage of the baseline model. the green curve shows the accuracy when we do not retrain the model. This accuracy is very low and random due to the random nature of the final fully connected layer. After retraining this layer we see we can get an accuracy boost which matches the baseline accuracy (red dotted line). The reason behind that is even thought there are pruned weights in the baseline part of the model, the last FC layer tries to learn the underlying pattern that is lost due to pruning. Hence this is a better technique to learn better.

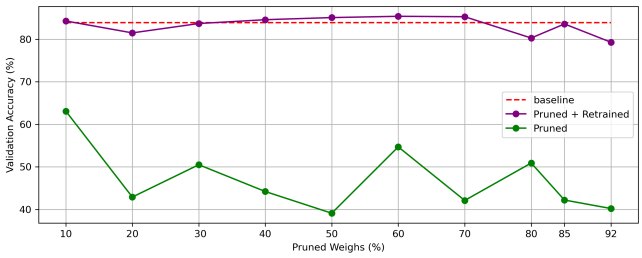


**Figure 17: Validation Accuracy vs Number of weights Pruned in Setup 2.**

## 6 Conclusion & Discussion

In this study, initially, we explored global pruning, layer-wise unstructured pruning, layer-wise channel pruning, and hardened models on a pretrained VGG16 to evaluate their impact on accuracy, runtime, and model size at sparsity levels ranging from 5% to 45%. At lower pruning percentages (up to 15%), layer-wise unstructured pruning retained accuracy the best, while global pruning showed slightly higher accuracy drops. At higher pruning percentages (beyond 30%), global pruning outperformed layer-wise methods in preserving accuracy due to its balanced sparsity distribution. Layer-wise channel pruning suffered the most significant accuracy loss even at low sparsity levels due to the removal of entire feature maps. The hardened model demonstrated substantial improvements in runtime and model size, making it suitable for resource-constrained environments, but at the cost of sharp accuracy declines. This highlights the trade-offs between accuracy and efficiency, with the choice of pruning method depending on the specific application requirements.

We then explored iterative pruning and demonstrated its effectiveness. Iterative pruning allows the model to retain accuracy even after substantial weight reductions. Specifically, pruning 43.75% of weights resulted in virtually no loss of accuracy, and 58% pruning led to only a minor 3% accuracy drop. This highlights the method's capability to efficiently compress models while maintaining performance. We also applied iterative pruning in a transfer learning scenario, where the pruned model adapted to the melanoma dataset. Remarkably, this approach allowed us to prune up to 80% of the weights with minimal impact on validation accuracy. These results underscore the potential of iterative pruning not only to reduce model size but also to enhance transfer learning tasks, providing a scalable solution for resource-constrained environments.

## References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[2] Muhammad Hasnain Javid. 2022. Melanoma Skin Cancer Dataset of 10000 Images. https://doi.org/10.34740/KAGGLE/DSV/3376422