Trade-Offs Between Compression and Accuracy for Skin Disease Classification in Edge Devices

Derek Montanez
Engineering Data Science
University of Houston

April 30, 2024

Abstract

Deep Neural Networks (DNNs) have demonstrated remarkable results in various domains, particularly in medical applications, significantly impacting the healthcare industry. However, these advancements may not be accessible to individuals residing in rural or socioeconomically disadvantaged areas. Bridging this gap is crucial to ensure equal access to cutting-edge medical technology. In this study, I investigate the trade-offs between model accuracy and compression in a fine-tuned DNN model designed for Skin Disease Classification. Utilizing the Skin Disease Classification Dataset [1] along with the neural network pruning library from Torch Pruning [0] to analyze these trade-offs.

1 Introduction

DNNs have made remarkable advancements across various sectors, particularly in the medical industry. However, equal access to advanced medical technologies remains a significant challenge, especially in rural and socioeconomically disadvantaged areas. Factors such as distance, transportation limitations, workforce shortages, lack of broadband access, and social stigmas contribute to inadequate healthcare in these communities, as reported by the Rural Health Information Hub [2].

Addressing these healthcare disparities necessitates the availability of affordable and accessible medical devices. One promising solution is the integration of DNNs into edge devices. DNNs have exhibited exceptional performance in areas such as medical imaging [3], disease diagnosis [4], and helps in risk management. Nonetheless, the increasing complexity of DNN models presents a common issue, which is an increase in model size, specifically the number of parameters in a model [5]. This complexity poses challenges when deploying DNNs on edge devices. The issue arises from the computational and storage restrictions in edge devices.

An area of research that aims to mitigate this challenge is neural network compression, which has been explored since the early 1990s. The objective of neural network compression is to create a subset of the original base model that is less computationally expensive and uses less space while maintaining similar inference performance to the base model. Various methods have been proposed in the past, including weight sharing [6], knowledge distillation [7], quantization [8], and pruning [9]. Pruning has two different kinds of methods, unstructured pruning [10] and structured pruning [13]. The former, pushes values of weights in a neural network to zero, while not changing the structure of the network. The disadvantage of this method is the need for hardware or software that is made for sparse matrix multiplication, if unstructured pruning is not used with these specialized tools you will not be able to see any advancements in terms of compression. Structural pruning on the other hand aims to change the actual structure of the network by removing unimportant structures of the model. This method does not need any specialized hardware or software since the size of the network is being reduced physically. In this work, I focus on structural pruning.

The model in this paper has been fine-tuned specifically for skin disease classification. The goal is to create an efficient model that may be suitable on edge devices dependendent on the specific use case. I would like to ensure both computational efficiency and reliable inference performance, and to analyze the compression and accuracy at different pruning levels.

2 Methods

2.1 Data

There are two different datasets that were used to develop the fine-tuned ResNet 18 model [11]. The original ResNet18 model was trained on the ImageNet dataset [12], an extremely large computer vision dataset for image classification. The ImageNet dataset contains roughly 1.4 million images, and classified across 1000 different classes.

For the fine-tuning process, the Skin Disease Classification [1] dataset curated by Riya Eliza Shaju was utilized. This dataset sourced from Kaggle, comprises 900 images depicting various skin diseases. These skin diseases were classified into 9 distinct classes, including Actinic Keratosis, Atopic Dermatitis, Benign Keratosis, Dermatofibroma, Melanocytic Nevus, Melanoma, Squamous Cell Carcinoma, Tinea Ringworm Candidiasis, and Vascular Lesion.

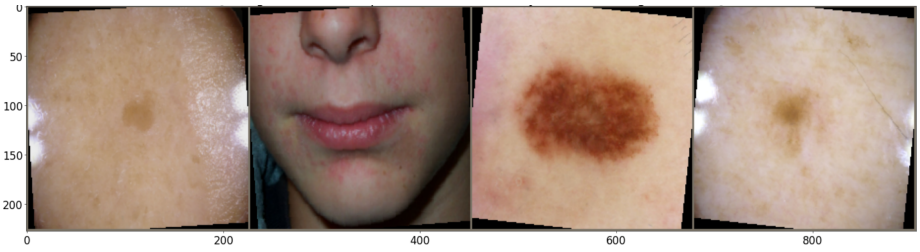

Figure 1: This figure shows some data from our dataset, that has been transformed. The images include the diseases Bengin Keratosis, Atopic Dermatitis, Melanocytic Nevus, and another case of Bengin Keratosis, respecitvely.

2.2 DepGraph: Towards Any Structural Pruning

This study focuses on employing structural pruning using the Torch Pruning library [0] within a ResNet18 model [11]. The developers of Torch Pruning are the authors of the paper 'DepGraph: Towards Any Structural Pruning' [0]. Prior to this study, a significant challenge in compression methods was their architectural specificity, lacking a pruning method or algorithm capable of efficiently and effectively identifying model dependencies across different architectures. Torch Pruning addresses this challenge by constructing a 'Dependency Graph' that introduces inter-layer and intra-layer dependencies within a model. Subsequently, this 'Dependency Graph' serves as input to a grouping/regularization algorithm, which drives grouped weights towards zero during training. After training, weights nearing or reaching zero are removed. Numerous variables are involved in the pruning process that need to be defined to build a MetaPruner. A MetaPruner is essentially an object that holds all the pruning variables provided by the user to determine how to prune the model. Starting with the importance criterion. This criterion is essential in determining the significance of grouped weights during pruning, options like Norm Importance, Hessian Importance, and Taylor Importance are provided by the author's library. For this study, the Taylor Group Importance criterion was utilized for all tests. The next step involves determining which layers to exclude from pruning within the network. In the testing phase, the decision was made to retain the last layer, which is the fully connected layer, for two primary reasons. First, it must maintain the correct number of output neurons corresponding to the number of classes (9) in our dataset. Second, it plays a crucial role in our network's functionality and performance. Finally, defining a pruning ratio, and this pruning ratio is the amount of group parameters to be pruned in our ResNet18 architecture [11], in the work I picked multiple values for testing purposes.

2.3 Fine Tuning

Initially, preprocessing steps were applied to both the training and test data. Emphasizing the need for robust generalization, data augmentation techniques

were applied. These included random rotations, random flips, random crops, and ColorJitter, which I believe accounts for variations in lighting and environments where users might capture images. The images were consistently resized to 224x224 RGB format across the dataset. Additionally, normalization was applied to align the mean and standard deviation of the images with the values from the ImageNet dataset [12], specifically set to [0.485, 0.456, 0.406] for mean and [0.229, 0.224, 0.225] for standard deviation. The decision to retain color information in the images was deliberate, I believe that color could significantly aid in disease classification tasks, especially on skin. During training, the loss function used was PyTorch's CrossEntropyLoss, which is a common choice for multi-class classification problems. The optimizer selected was AdamW after testing it with other optimizers like SGD, Adam, RMSprop, and Adagrad. AdamW was consistently used throughout all tests with a learning rate of 0.001 due to time constraints. A smaller batch size of 12 was utilized in the data loader, which was needed by computational constraints on the device. Finally, the testings were conducted for 15 epochs of fine-tuning training. [14]

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 2: This figure shows a visual representation of the original ResNet18 Architecture, though this does not accurately represent our base model, since our base model will have 9 output neurons, instead of 1000 which was for the ImageNet dataset.

2.4 Procedure

The procedure for conducting these tests involves defining objects, parameters, and hyperparameters utilized throughout the process. This includes specifying the loss function, optimizers, importance criterion, pre-trained model, layers to ignore, and example inputs. The initial step is to configure the ResNet18 model [11] to suit our specific problem; in this context, adjusting the fully connected layer from 1000 classes (ImageNet) to 9 classes (Skin Disease Classification). Subsequently, creating the MetaPruner object from the authors to define our

pruning strategy, which incorporates values such as the importance criterion, model, example inputs, ignored layers, and the pruning ratio.

Following the definition of the pruning object, the model undergoes pruning based on the selected importance criterion, which in our case is the Taylor Group importance criterion. The model is updated during this process, removing the need to save the model separately using libraries like joblib. After pruning, the fine-tuning phase begins, involving data transformation for both the training and test sets, and the creation of data loaders based on these transformed datasets. The pruned model is then trained for 15 epochs, and its performance is evaluated using the test data loaders. [14]

3 Results

In this section, I present the outcomes of our experiments involving fine-tuning pruned models using Torch Pruning [0] within a ResNet18 architecture [11] for skin disease classification. I aimed to explore the trade-offs between compression and accuracy for the fine-tuned model. The authors of 'DepGraph: Towards Any Structural Pruning' [0] primarily focused on conducting tests on popular pre-trained neural network models however, their code did not include results from fine-tuning any models. Below, I outlined the key additions made to the original code base, which initially only encompassed the pruning algorithm and steps.

```
with open("results_prune_first_10.txt", "a") as f:
    for epoch in range(num_epochs): #(loop for every epoch)
        print("Epoch {} running".format(epoch)) #(printing
            message)
        """ Training Phase """
        model.train()    #(training model)
        running_loss = 0.   #(set loss 0)
        running_corrects = 0
        # load a batch data of images
        for i, (inputs, labels) in enumerate(
            train_dataloader):
            inputs = inputs.to(device)
            labels = labels.to(device)
            # forward inputs and get output
            optimizer.zero_grad()
            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)
            loss = loss_function(outputs, labels)
            # get loss value and update the network weights
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
            running_corrects += torch.sum(preds == labels.
                data).item()
```

```
        epoch_loss = running_loss / len(train_dataset)
        epoch_acc = running_corrects / len(train_dataset) *
            100.
        # Append result
        train_loss.append(epoch_loss)
        train_accuary.append(epoch_acc)
```

Listing 1: Training Process for Fine Tuning

```
    with open("results_prune_first_10.txt", "a") as f:
    model.eval()
    with torch.no_grad():
        running_loss = 0.
        running_corrects = 0
        for inputs, labels in val_dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)
            loss = loss_function(outputs, labels)
            running_loss += loss.item()
            running_corrects += torch.sum(preds == labels.
                data).item()
        epoch_loss = running_loss / len(val_dataset)
        epoch_acc = running_corrects / len(val_dataset) *
            100.
        # Append result
        val_loss.append(epoch_loss)
        val_accuary.append(epoch_acc)
        # Print progress
```

Listing 2: Evaluation Process for Fine Tuning

```
transforms_train = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomCrop((224, 224)),
    transforms.RandomRotation(10),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(brightness=0.1),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229,
        0.224, 0.225])
])

transforms_val = transforms.Compose([
    transforms.Resize((224, 224)),
     transforms.CenterCrop((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229,
        0.224, 0.225])
```

```
])
```

Listing 3: Data Pre-Processing for Training and Test Data

Figure 3 illustrates the pruning ratio and accuracy of the fine-tuned model on the Skin Disease Classification [1] test data. Compared to the base model's accuracy of 82.3204, the pruned models have good inference performance. When performing tests for different pruning ratios I observed an increase in accuracy with a 20 percent pruning ratio, reaching 82.8729 accuracy. But, as the pruning ratio increased there was a noticeable decline in accuracy loss, with the lowest accuracy recorded at 72.3757 when applying a 60 percent pruning ratio. This decline is expected since a 60 percent pruning ratio removes a substantial number of parameters, including some that would be deemed to be crucial for inference.

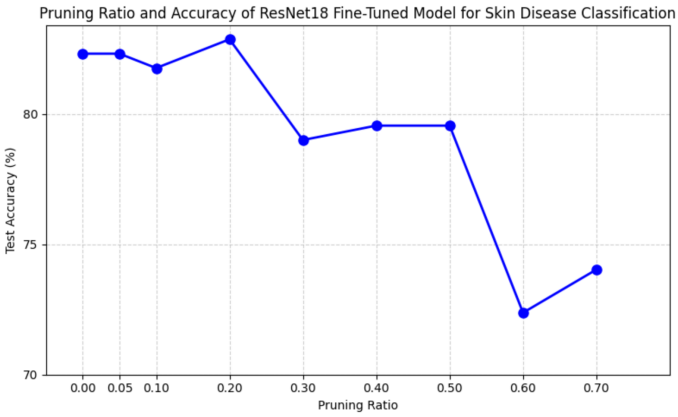

Figure 3: Pruning Ratio and Accuracy

Figure 4 presents a comparison of parameter counts in the pruned models along with their respective inference accuracies. Parameter count serves as a reliable metric for assessing neural network compression. Unlike unstructured pruning methods that just zero out weights without physically removing parameters, this approach reduces the physical size of the model. This eliminates the need for external tools (hardware or software) to detect compression performance. During the tests, the base architecture was initially constructed with approximately 11.181120 million parameters. Notably, the model exhibiting the best inference performance was achieved with a 20 percent pruning ratio, reducing the parameter count to 7.12807 million parameters. This reduction represents a nearly 1.5x decrease in parameter count compared to the base model.

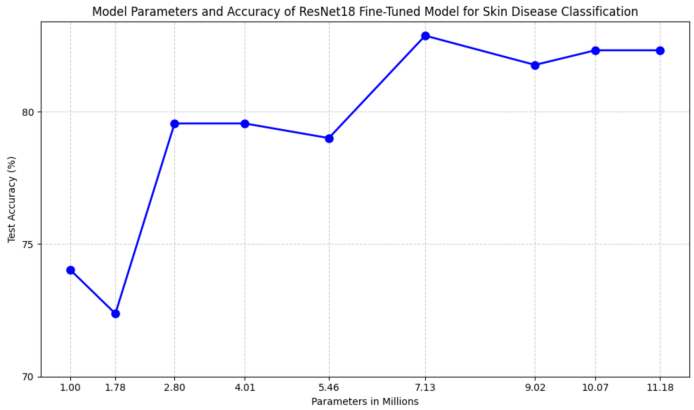

Figure 4: Parameter Count and Accuracy

The table below provides a comprehensive and easily interpretable summary of the results obtained from various tests conducted. Additionally, it displays the changes in the number of MACs (Multiply-Accumulate), which is a performance metric indicating the computational complexity of a model. MACs consider numerous operations performed during inference, including matrix multiplications. A smaller MAC count is desirable as it means a less computationally intensive model. The results of the tests demonstrate the effectiveness of the pruning method introduced in 'DepGraph: Towards Any Structural Pruning' [0]. This method accurately identifies parameters for pruning while maintaining a high level of accuracy comparable to the base model.

	Accuracy	Change in Accuracy	Parameters (Millions)	Change in Parameter Size	MACs (Millions)	Change in MAC's
ResNet18-Base	82.3204		11.181129		1821.669385	
ResNet18- 5% Pruning Ratio	82.3204	0	10.066435	-1.114694	1632.000245	-189.66914
ResNet18-10% Pruning Ratio	81.768	-0.5524	9.024737	-2.156392	1474.061545	-347.60784
ResNet18-20% Pruning Ratio	82.8729	0.5525	7.12807	-4.053059	1178.374212	-643.295173
ResNet18-30% Pruning Ratio	79.0055	-3.3149	5.464787	-5.716342	905.419253	-916.250132
ResNet18-40% Pruning Ratio	79.558	-2.7624	4.01367	-7.167459	677.19855	-1144.470835
ResNet18-50% Pruning Ratio	79.558	-2.7624	2.801193	-8.379936	486.947849	-1334.721536
ResNet18-60% Pruning Ratio	72.3757	-9.9447	1.779457	-9.401672	315.407593	-1506.261792
ResNet18-70% Pruning Ratio	74.0331	-8.2873	0.999398	-10.181731	187.320644	-1634.348741

Figure 5: Comparison table displaying the performance metrics of pruned fine-tuned ResNet18 models relative to each other.

4 Discussion

In this study, I utilized the 'DepGraph: Towards Any Structural Pruning' [0] algorithm to prune a ResNet18 model [11], subsequently fine-tuning it for skin disease classification. The comparative analysis presented in the tables and graphs offers insights into the impact of model compression on accuracy, serving

as indicators of how neural network compression affects inference performance, which will then be applied to use cases within industry for neural network compresion use on edge devices.

Our results clearly demonstrate that neural network compression significantly reduces the size of the base ResNet18 model [11] without substantially compromising inference performance. In fact, in one instance, observed an improvement in inference performance compared to the base model, particularly with the 20 percent pruning ratio model. This finding is intriguing and suggests that the 20 percent pruning ratio may represent an optimal balance for generalizability in this specific use-case.

This is not an entirely new idea; it resonates with concepts such as dropout in neural networks. Dropout acts similarly by randomly dropping neurons during training, enhancing generalizability and, in some cases, improving inference performance.

However, it's crucial to highlight that beyond the 20 percent pruning ratio, higher pruning ratios led to a significant decline in inference performance, with accuracy dropping as low as 10 percent from the base model. This observation points to a potential threshold effect where a 20 percent pruning ratio optimizes model generalizability. Higher ratios resulted in the removal of parameter groups which seem to be important for inference, causing a greater loss in inference performance.

5 Limitations and Future Work

5.1 Limitations

The trade-off between compression and accuracy in neural network models is context-dependent and lacks a universal rule or standard. Each scenario requires careful consideration based on specific use-cases. For instance, in applications such as health diagnostics deployed on edge devices, where accuracy is very important for critical predictions like high blood pressure or risk of heart disease, prioritizing the highest compression performance at the expense of prediction accuracy may not be advisable.

It is essential to recognize that the optimal pruning ratio or compression level varies based on real-world applications. What works well for one use-case may not be suitable for another. Therefore, decision-making regarding compression and accuracy trade-offs should be informed by the specific requirements and priorities of the application in question.

5.2 Future Work

Future work will involve exploring alternative pre-trained models specifically tailored for medical image classification tasks. While ResNet18, pretrained on

ImageNet [12], demonstrated a base accuracy of approximately 82 percent, this accuracy may fall short of industry and applicational standards, particularly in critical tasks such as skin disease classification. This relates to the ongoing challenge of pruning while maintaining good inference performance being a case-by-case problem, a consideration highlighted in the limitations section. Also, this work was not able to reach the stage of implementation on edge devices, which is something I look forward to pursuing in future work.

6 References

[0] G. Fang, X. Ma, M. Song, M. Bi Mi and X. Wang, "DepGraph: Towards Any Structural Pruning," in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 2023 pp. 16091-16101.

[1] Riya Eliza Shaju. Skin Disease Classification [Image Dataset]. Kaggle. https://www.kaggle.com/datasets/riyaelizashaju/skin-disease-classification-image-dataset

[2] Rural Health Information Pub. Healthcare Access in Rural Communities. ruralhealthinfo. https://www.ruralhealthinfo.org/topics/healthcare-access

[3] Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4$_2$8

[4] Calisto, F.M.; Nunes, N.; Nascimento, J.C. BreastScreening: On the use of multi-modality in medical imaging diagnosis. InProceedings of the International Conference on Advanced Visual Interfaces, Ischia, Italy, 28 September–2 October 2020; pp. 1–5

[5] Bernstein, Liane Sludds, Alexander Hamerly, Ryan Sze, Vivienne Emer, Joel Englund, Dirk. (2021). Freely scalable and reconfigurable optical hardware for deep learning. Scientific Reports. 11. 10.1038/s41598-021-82543-3.

[6] W. Chen, J. Wilson, S. Tyree, K. Weinberger and Y. Chen, Compressing Neural Networks with the Hashing Trick, Proc. International Conference on Machine Learning (ICML-15), Lille, France.

[7] Hinton, G.E., Vinyals, O., Dean, J. (2015). Distilling the Knowledge in a Neural Network. ArXiv, abs/1503.02531.

[8] Emile Fiesler, Amar Choudry, H. John Caulfield, "Weight discretization paradigm for optical neural networks," Proc. SPIE 1281, Optical Interconnections and Networks, (1 August 1990); https://doi.org/10.1117/12.20700

[9] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," in IEEE Transactions on Neural Networks, vol. 1, no. 2, pp. 239-242, June 1990, doi: 10.1109/72.80236.

[10] Lee, N., Ajanthan, T., Torr, P.H. (2018). SNIP: Single-shot Network Pruning based on Connection Sensitivity. ArXiv, abs/1810.02340.

[11] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[12] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer

Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," Aug. 2016, [Online]. Available: http://arxiv.org/abs/1608.08710

[14] Kirudang. "Deep Learning Computer Vision using Transfer Learning ResNet-18 in PyTorch (Skin Cancer)." Medium, ttps://medium.com/@kirudang/deep-learning-computer-vision-using-transfer-learning-resnet-18-in-pytorch-skin-cancer-8d5b158893c5.