On Developing Efficient and Robust Neural Networks for Healthcare using Condensa Model Compression System

Keaton Rowley
University of Utah

UUCS-21-015

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

18 May 2021

Abstract

This thesis describes a study of how compressing neural networks used to identify malaria cells and respiratory diseases affects network accuracy, and system performance metrics. Its focus is on a state-of-the-art framework for neural network (NN) compression called Condensa to compress network size and improve network performance according to different compression schemes. It details the impact malaria and lung disease have on a worldwide level each year. It then describes previous research in automating medical image classification. It also gives a background on what research has been applied towards network compression. The study also describes work in developing a CNN for the Malarial and Chest-X-ray datasets. It details the results of compressing the CNN using Condensa's Filter, StructPrune, Prune, and Quantization schemes. This thesis provides a complete software implementation to help reproduce our results and facilitate tool adoption. It also indicates a plan for future research in applying Condensa towards the problem of developing an efficient system of disease identification for different medical dataset problems.

On Developing Efficient and Robust Neural Networks for Healthcare using Condensa Model Compression System

Approved:

May 17, 2021

/DATE
Ganesh Gopalakrishnan
Thesis Faculty Supervisor

May 19, 2021

/DATE
Mary Hall
Director School of Computing

May 17, 2021

/DATE
Director of Undergraduate Studies
H. James de St. Germain

By:
Keaton Rowley

A Senior Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Computer Science

School of Computing
University of Utah
November 2020

May 18, 2021

Contents

1 Abstract **3**

2 Introduction **3**
2.1 Disease Context 3
2.2 Identifying Malaria 4
2.3 Identifying Lung Disease 4
2.4 Review of Previous Research 5
2.4.1 Previous Network Compression Research 5
2.4.2 Network Compression Techniques 6
2.5 Condensa 7

3 Methods **8**
3.1 Thesis 8
3.2 Condensa Testing Script 8
3.3 Datasets 8
3.3.1 Malaria Dataset 9
3.3.2 Chest-X-ray Dataset 9
3.4 Condensa Testing for the Malarial Cell Dataset 11
3.5 Condensa Testing for the Chest-X-ray Dataset 11

4 Results **12**
4.1 Cinchona Network Density Results 13
4.2 Cinchona Compression Method Results 16
4.3 Chest-X-ray Network Density Results 18
4.4 Chest-X-ray Compression Method Results 21

5 Discussion **22**
5.1 Density Network Trends 22
5.2 Compression Method Trends 23

6 Conclusion **23**

1 Abstract

This thesis describes a study of how compressing neural networks used to identify malaria cells and respiratory diseases affects network accuracy, and system performance metrics. Its focus is on a state-of-the-art framework for neural network (NN) compression called Condensa to compress network size and improve network performance according to different compression schemes. It details the impact malaria and lung disease have on a worldwide level each year. It then describes previous research in automating medical image classification. It also gives a background on what research has been applied towards network compression. The study also describes the author's own work in developing a CNN for the Malarial and Chest-X-ray datasets. It details the results of compressing the CNN using Condensa's Filter, StructPrune, Prune, and Quantization schemes [1]. This thesis provides a complete software implementation to help reproduce our results and facilitate tool adoption. It also indicates a plan for future research in applying Condensa towards the problem of developing an efficient system of disease identification for different medical dataset problems.

2 Introduction

2.1 Disease Context

Machine learning is a technology which has revolutionized a variety of industries from online advertising to facial recognition. However, its widespread adoption for medical applications has been somewhat slower. Machine learning has potential applications in medical imaging, automated disease detection, and automating medical records. Its slow uptake is primarily due to several factors. Health data availability is often limited as it is usually spread through multiple insurance agencies and different providers. Because of this, split data is also often represented in different formats. Additionally, most medical data has high privacy requirements due to laws such as HIPAA. This makes the transfer and storage of medical data a complicated process. Limited data sets can also create problems with bias within the system. Machine learning software may perform worse when provided data from an underrepresented group or group different from the provided training data.

Also, of high importance is the performance and accuracy of such systems

as inaccurate diagnosis could potentially lead to wrong treatment and harm for the patient.

Despite these setbacks, many methods are being developed to address concerns about data transfer and to successfully combine data sets. With the growing size of medical data sets over time, it is increasingly important that efficient methods can be found to process data and generate accurate predictions. One growing use case for machine learning is for medical imaging. Machine learning can be used in applications such as identifying diseased malarial cells and detecting lung diseases from chest X-ray scans.

2.2 Identifying Malaria

Malaria is a disease which negatively impacts millions of people worldwide. It is caused by the plasmodium parasite which is contracted through the bite of certain species of female mosquito. It infects multiple systems of the human body and often causes permanent health issues and death. The species of parasite which cause malaria are plasmodium falciparum, plasmodium vivax, plasmodium ovale, plasmodium knowlesi, and plasmodium malariae. Of these, plasmodium falciparum contributes to the most deaths. Between the five of them, malaria kills over 400 hundred thousand people every year [2]. The effective treatment of malaria hinges on being able to rapidly diagnose patients with malaria to stall the disease in as early a stage as possible. Since malaria is endemic in many less developed areas, finding a cost-effective way to diagnose the 220 million infections that occur each year is critical. A large majority of malaria cases are identified using light microscopy. This is done by a trained technician using a droplet of the patients' blood on a specially prepared slide to analyze red blood cells for the presence of parasites. As this is a highly time-consuming endeavor, automating the process has been a desirable research area. In general, research in automating malarial cell identification has followed two stages. The first is to extract images of red blood cells from slide images and normalize those images for lighting and color variations. The second focus is creating a classifying algorithm which can be trained to generate a model to a sufficient accuracy to classify future malarial cell infections.

2.3 Identifying Lung Disease

Lung disease is another problem which impacts millions of people. In 2016,

approximately 3.4 million people died due to chronic obstructive pulmonary disease (COPD). Another 400 thousand passed away from asthma. Traditionally, radiology has been the standard method for detecting lung diseases. There is currently a shortage of radiologists worldwide. For example, a report by the Royal College of Radiologists (U.K.) stated that only 2 percent of radiology departments can fulfill their imaging reporting requirements within contracted hours. Machine learning could serve to alleviate this problem as it is able to identify lung diseases with similar accuracy to a trained radiologist [3]. Chest X-rays are one of the most widely used tools for diagnosing lung diseases. As such, it represents a possible opportunity for automation via computer vision technologies.

2.4 Review of Previous Research

Deep neural networks (DNNS) are becoming increasingly larger to obtain greater accuracy and include more information. Consequently, there have been many efforts to try and reduce the size of these networks to improve inference time. In general, the metrics used for network compression are the accuracy and the number of floating-point operations (FLOPS). Also, the run-time memory (ratio of space for storing hidden layer features vs. the original network) footprint can be reported to show improvements in storage capacity.

2.4.1 Previous Network Compression Research

Previous research indicates that training a small, shallower network has poorer generalization compared to an equally sized pruned network generated from a larger pretrained network [4]. It has also been found that pruned CNNs and LSTMs outperform smaller dense models. In a study carried out by Gupta, the models achieved a compression ratio of 10 in the number of non-zero parameters with tiny losses to accuracy [5].

Other experiments have shown that a DNN with more degrees of freedom allows for a larger solution set to be chosen from the parameter space [6]. Additionally, overparameterization has been shown to have a regularization effect on the loss space when trained with stochastic gradient descent [7].

Another phenomenon that was recently discovered is the double descent phenomenon. Basically, on certain classes of deep learning models, the test accuracy goes through a descent phenomenon when not using early stopping

or regularization. As the model size grows bigger, the test / train error initially decreases to an initial minimum. It then increases because of the bias-variance tradeoff. Unexpectedly, with further increases of the network size, the test error drops lower than the initial minimum.

Having larger networks after the second descent results in weights close to zero. Or in other words, the norm of the weights gets dramatically smaller with increasing network size during the second descent. As network sizes increases, there is a smaller return on accuracy with a large complexity increase. [8].

Frankle and Carbin also showed that training a network to convergence with many parameters made it easier to find a sub network that maintains performance when trained from scratch. This implies that compressing large pretrained and overparametrized DNNS trained to convergence has advantages from a performance and storage perspective. In their study, they found that retraining a compressed model can maintain performance.

2.4.2 Network Compression Techniques

An important tool in network compression is retraining. Basically, a pruned network may undergo a drop in accuracy as nodes are zeroed out or removed. To make up for this, a network may be retrained to recover that accuracy. This can either be done using unsupervised or supervised training methods.

There have been several methods developed to reduce network size. One of the simplest forms is weight sharing. This is accomplished by reducing 32-float weight numbers to 2-bit unsigned integers (uints). Each of these is indexed to a centroid which is averaged from the original float weights. In this case, a cluster index of the original weights is stored with a list of centroids. This massively reduces the amount of data that must be stored. The averaging to produce the centroids also has a regularizing effect on the network [9]. Various variations to generate the centroids have been proposed.

Network pruning is the process by which weights less than a specified threshold are zeroed out of the network by assigning their weight to be zero and then restructuring the network to remove zeroed out nodes [10]. More recent models focus on pruning a certain density of the weights. Basically, the weights closest to zero are pruned to match the desired density. Once pruning takes place, a network must often be retrained to regain accuracy.

Quantization is another common approach to reducing network size. Typical GPUs use either 32-bit floating point or half precision floating point.

Large floating point networks are converted into networks with lower precision (INT–8, INT–4, INT–2 or 1 bit representations) to reduce the memory footprint of the network.

2.5 Condensa

Much of recent medical computer vision research has focused on the use of deep neural networks (especially convolutional neural networks which specialize in computer vision). DNNs offer an advantage over other methods in that they do not require a set of identification features to be selected by the network creator. Instead, the connections in the network can be trained sufficiently to achieve similar if not greater accuracy than other methods. However, the drawback of DNNs is that the scale of computations required to train the network and generate predictions is much higher than other methods. Since modern DNNs often contain millions of parameters or hundreds of layers, it is highly desirable to be able to reduce the size of the network (or compress the network). Doing so can be an often-experimental endeavor and require a great deal of manual tuning. Consequently, tools are being built to systematically automate the process of compressing DNN networks. Condensa is a state-of-the-art tool which was developed as part of Josephs's work in collaboration with Nvidia Research [1].

Condensa is a framework for programmable model compression that offers many advantages. In compressing a DNN, the requirements of different compression contexts (such as DNN structure, target hardware platform, and user's optimization objective) may vary widely. For example, a user may focus on reducing inference latency or instead focus on reducing the total memory footprint. The former strategy may focus on pruning convolutional filters while the latter may prune individual non-zero weights. However, different network types may necessitate different types of compression for the same optimization objective. A language modeling network such as a transformer might require pruning 2-D blocks (since it does not have convolutional layers) while a filter pruning strategy could be employed on a CNN. Condensa provides built in support for compressing according to these various schemes. Condensa works by taking a reference model and a compression scheme, and then uses a black box sample-efficient Bayesian optimization to generate a sparsity value. The model, scheme, and sparsity value are fed into the L-C optimizer [1]. The optimizer works by globally distributing the sparsity value across the model and then uses the L-C algorithm to recover

accuracy in multiple training cycles [1]. Because of its advantages, Condensa is a promising tool to use in creating an efficient DNN for classifying medical image datasets. The focus of our study is on how Condensa can be utilized to optimize DNN network performance for medical datasets.

Before Condensa was developed there has been a large body of research dedicated to compressing neural networks. Some of these include reinforced learning, AutoSlim, and ADMM [1]. The L-C algorithm was recently developed and is utilized by the Condensa framework.

3 Methods

3.1 Thesis

The goal of this thesis is to present a study on creating an efficient DNN to classify medical image datasets using the Condensa software. The two test data sets were the Malarial Cell Images Dataset and the Chest-X-ray Datasets, both of which are publicly available through NIH [11][12]. Different network condensing methods and density levels were compared. Comparison was done via GPU memory and percent of GPU utilization, latency, and test accuracy parameters.

3.2 Condensa Testing Script

A major component of this project was streamlining the Condensa testing process for each dataset. This was done by creating a Bash script that could run in a Linux environment. This script enabled the user to create models with settings adjusted in a global file. Latency, throughput, GPU memory, and GPU utilization tests could also be run from the same script. The script also allowed the user to combine the results from multiple tests into a single graph. One of the notable features was that the script also integrated with Comet ML. Data from the Condensa network generation process is piped to the users Comet account. Many of the graphs in this paper are sourced from Comet.

3.3 Datasets

The two datasets used in this study were the NIH Malaria Dataset and the

NIH Chest X-ray dataset.

3.3.1 Malaria Dataset

The former consists of 27,558 images split evenly between parasitized and uninfected cell images. These were generated from thin blood smear slide images from the Malaria Screener research activity. Giemsa-stained thin blood smear slides were taken from 150 P. falciparum-infected patients and 50 healthy patients. The images were labeled by an expert slide reader from the Mahidol-Oxford Tropical Medicine Research Unit in Bangkok, Thailand. They were then segmented into individual cells and split into individual images [11]. Images were transformed to a pixel dimension of 32X32 pixels. Additionally, random horizontal flips, and rotation up to 10 degrees, and color jitter were introduced to account for variation in lighting among samples. Figure 1 and Figure 2 demonstrate typical images found within the Malaria dataset.

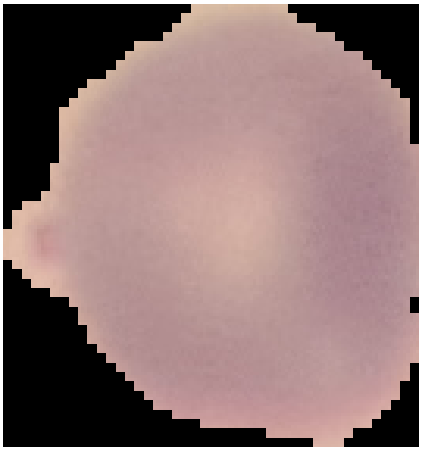


Figure 1: Healthy Cell

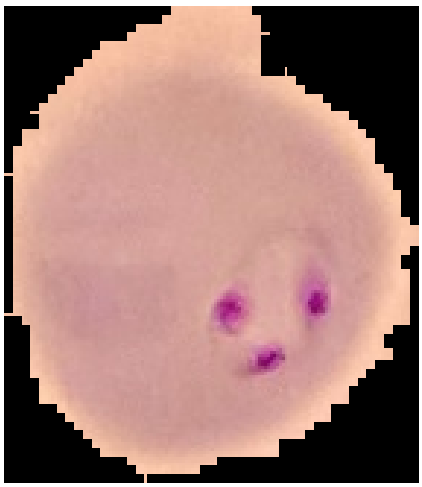


Figure 2: Malarial Cell

3.3.2 Chest-X-ray Dataset

The Chest-X-ray dataset contains 108,948 frontal chest-X-ray images from 32,717 patients stored in a pdf format [12]. Each image is labelled with a subset of 15 labels (Normal, Atelectasis, Cardiomegaly, Consolidation, Edema, Effusion, Emphysema, Fibrosis, Hernia, Infiltration, Mass, Nodule, Pleural

Thickening, Pneumonia, Pneumothorax) The labels were extracted originally via NLP methods from radiology reports included with each of the X-rays to detect phrases indicating the presence of disease. The reports were sourced from the Picture Archiving and Communication Systems of the NIH Clinical Center [13]. The original method for labelling this data set was validated by comparing results from a small set of 3,851 expertly labelled radiology to labels generated by the NLP report labelling algorithm.

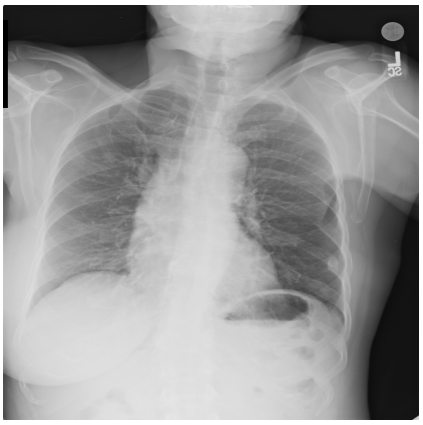


Figure 3: No Label Lung

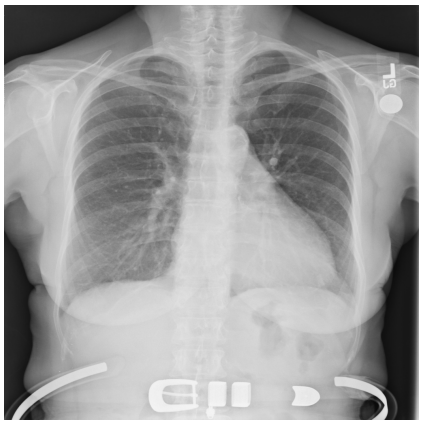


Figure 4: Cardiomegaly Lung

Dataset Preparation Prior to doing any data processing, the images were split into three subsets of (training, validation, and test) data such that no duplicate patient images (as patients had multiple scans) could exist in the training and validation split categories. Additionally, some of the classes had much higher numbers of training examples. This could induce a bias towards certain classes without correction. This was accounted for with an additional weight vector which weighted each class by frequency in all calculations. The weights were calculated by summing the frequency of all types of classes and dividing the occurrence of each class by this total.

The original dimensions of each X-ray image was 3000x3000 pixels. During data processing, images were resized to a pixel dimension of 64x64 pixels as well as having random rotation within 10 degrees. No image was flipped as this could potentially generate a false disease result if the heart were represented on the patient's right side. Additional transformations were random additions of color jitter to normalize lighting differences.

3.4 Condensa Testing for the Malarial Cell Dataset

Once pre-processing was complete, training was carried out via the Condensa system. All models used the Pytorch Resnet-18 DNN network. The structure used for the malarial cell dataset is shown in Table 1. Condensa was run on this model and trained with the malarial cell dataset with different density targets and different methods to reduce the density. The operations included pruning, structured pruning, filter pruning, and quantization network reduction operations. The prune method works by pruning an overall network to a given density. This is in general done by zeroing out weights that are very close to zero. The structured pruning scheme works by pruning blocks of non-zeroes. Filter pruning prunes filters from convolutional layers. Finally, quantization converts from 32-bit float point to 16-bit [14].

These different models were then compared based on various performance criteria.

Table 1: Malaria Cell Dataset Resnet-18 Structure

Layer Name	Output Size	ResNet-18
conv1	16x16	7x7, 64, stride 2 3x3 max pool, stride 2
layer1	8x8	[3x3, 64 3x3,64] x2
layer2	4x4	[3x3, 128 3x3, 128] x2
layer3	2x2	[3x3, 256 3x3, 256] x2
layer4	1x1	[3x3, 512 3x3, 512] x2
average pool	1x2	1x1, 2 fc, softmax

3.5 Condensa Testing for the Chest-X-ray Dataset

To maintain comparability, the Chest X-Ray dataset was also modeled using the ResNet-18 architecture. The main difference is that Chest-X-ray images may have multiple labels. This also meant that the chest-X-ray network used a weighted logit threshold value to determine the label for each

class. As with the malaria cell dataset, the Chest-X-ray dataset was tested using pruning, structured pruning, filter pruning, and quantization network reduction operations. Its structure is also listed in Table 2.

Besides different network pruning operation schemes, networks were generated with different densities to compare how accuracy recovery progressed with Condensa.

Table 2: Chest X-ray Dataset Resnet-18 Structure

Layer Name	Output Size	ResNet-18
conv1	32x32	7x7, 64, stride 2 3x3 max pool, stride 2
layer1	16x16	[3x3, 64 3x3,64] x2
layer2	8x8	[3x3, 128 3x3, 128] x2
layer3	4x4	[3x3, 256 3x3, 256] x2
layer4	2x2	[3x3, 512 3x3, 512] x2
average pool	1x1	1x1, 15 fc, logit threshold

4 Results

One of the primary focuses of testing was on the difference of performance between networks with different condensing strategies. These included differences in overall network density reduction, focused filter size reduction, or only a reduction in bit size. Each network in the Malaria dataset was tested using the same dataset and allowed to train for 45 cycles of the Condensa algorithm.

In each of these cycles, a sparsity value is chosen which will lose only up to a certain accuracy threshold. This sparsity value is used to calculate a final sparsity that optimizes a user-defined objective function in the constrained sparsity domain. The model is accordingly modified / compressed. Afterward, Condensa performs L-C accuracy recovery which retrains the network to recover the lost accuracy. This process is repeated a user specified number

of times [1].

4.1 Cinchona Network Density Results

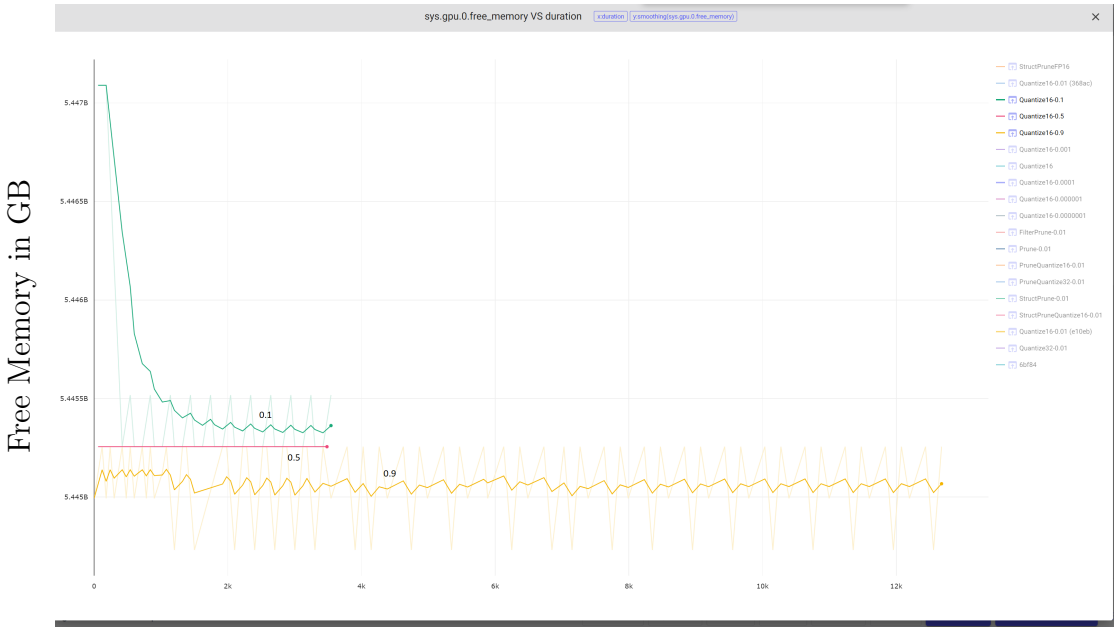


Figure 5: Malaria data set Free Memory (GB) vs. Condensa Training Duration with Quantize16 0.1, 0.5, 0.9 densities.

In Figure 5, the overall free GPU memory available to each network is graphed. The Quantize16 networks shown in this graph are trained to a density of 0.1, 0.5, and 0.9, respectively, represented as a decimal percentage density. Quantize16 uses 16-bit float values for its network. Of note in this graph is that the amount of free memory for 0.1 is greater than 0.5, and the 0.9 density network has the least free memory.

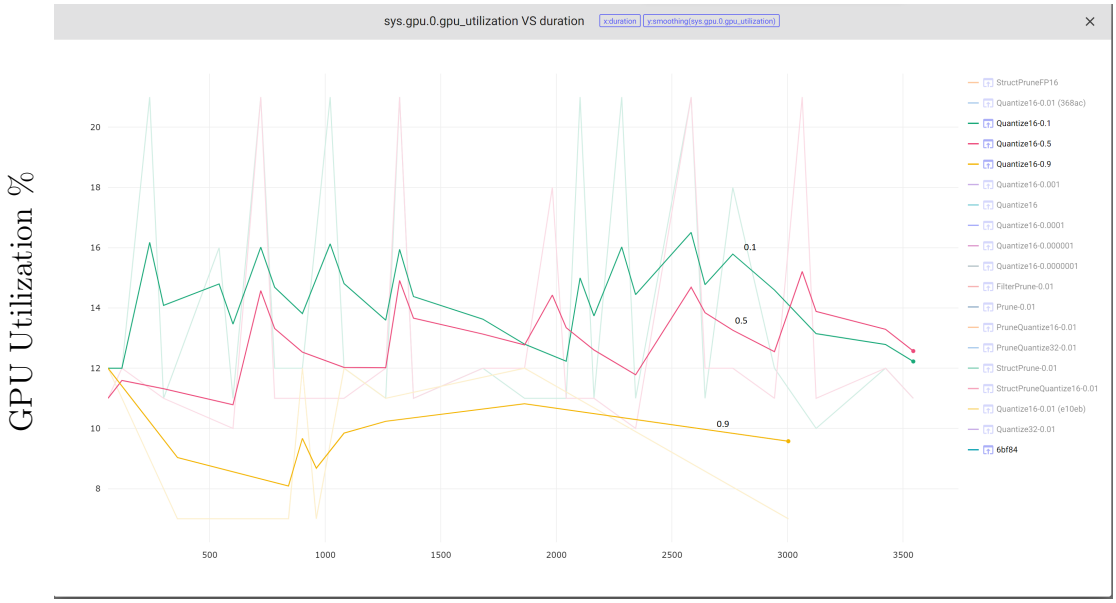


Figure 6: Malaria data set GPU Percent Utilization vs. Condensa Training Duration with Quantize16 0.1, 0.5, 0.9 Densities.

The same networks are represented in Figure 6. In this graph the average GPU utilization of each graph throughout the Condensa training duration is depicted. The GPU utilization has been smoothed for ease of seeing overall trends in the data. Overall, the percent GPU usage was highest for 0.1, second highest for 0.5, and smallest for 0.9.

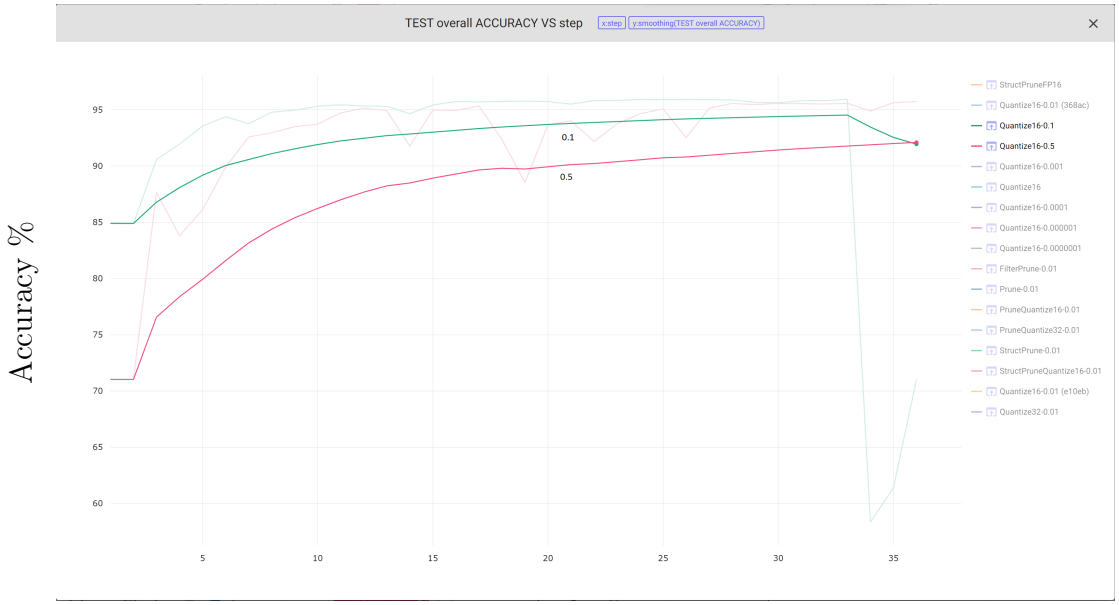


Figure 7: Malaria data set Test Accuracy vs. Condensa Training Step with Quantize16 0.1, 0.5.

The accuracy for the 0.1 and 0.5 density network is shown in Figure 7. In this case, the 0.1 density network consistently maintains a higher accuracy level. It should be noted that with sufficient training cycles, the networks will converge to a common accuracy. This is usually the maximum accuracy of the network. An aspect of testing the network was to see how sparse a network could become before accuracy losses could be seen. In Figure 8, different densities were tried for the network. At densities below 0.01, the overall accuracy seemed to drop. However, this trend switched over for networks with a density smaller than 0.00001. This could be attributed to the fact that the dataset is a binary classification problem and ResNet-18 can tend to be over-parameterized.

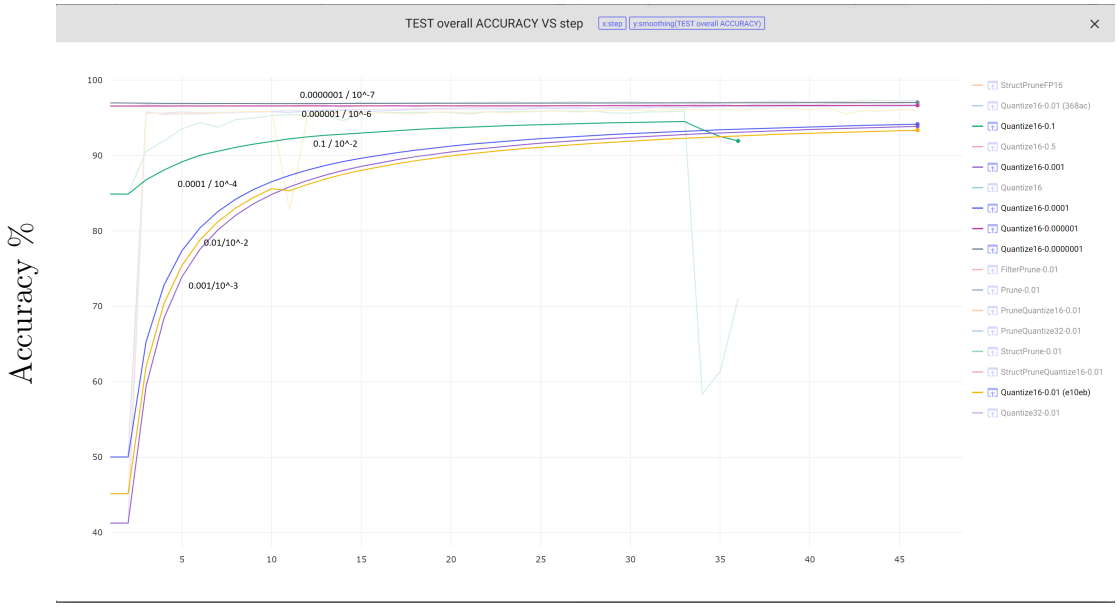


Figure 8: Malaria data set Test Accuracy vs. Condensa Training Step with Quantize16 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.0000001

4.2 Cinchona Compression Method Results

Other than different densities, the network was also tested with different compression schemes. In each of these tests, the network was pruned to a 0.1 density. These included Quanize-16, FilterPrune, Prune, PruneQuantize16, StructPrune, and StructPruneQuantize. Quantize-16 was basically using 16-bit floats to store data instead of the standard 32-bit numbers. FilterPrune pruned (removed) some of the filters between convolutional layers. Prune pruned the overall network to a given density. StructPrune removed neurons from fully connected layers and removed filters from convolutional layers. In Figure 9, FilterPrune, StructPrune, and StructPruneQuantize all achieved around 50 percent accuracy which is basically random for a binary classified dataset.

Quantize32 and Quantize16 achieved the highest accuracy. Quantize32 is a network which quantizes to the standard 32-bit operators. As expected, it achieved the highest accuracy as it underwent minimum modification. Be-

neath it in accuracy, Quantize16 achieved an average 72.44 percent accuracy.

For the percentage of GPU utilization, Quantize16, FilterPrune, PruneQuantize16, and StructPruneQuantize had the lowest GPU utilization. Prune32 had the highest GPU utilization and Quantize32 and StructPrune had similar performance.

GPU memory usage was very similar for all networks except StructPrune and Quantize16.

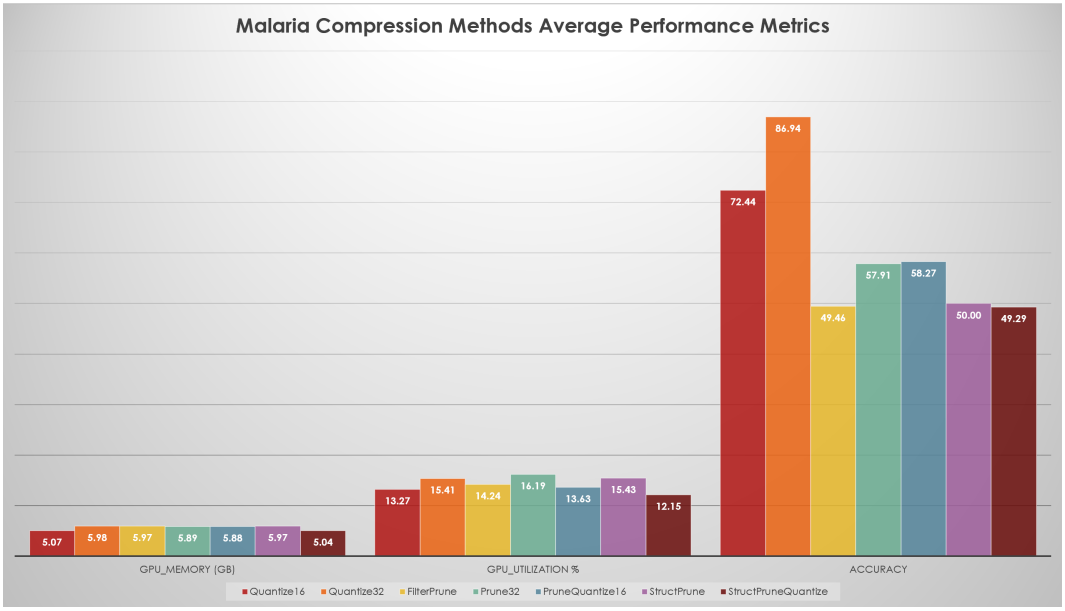
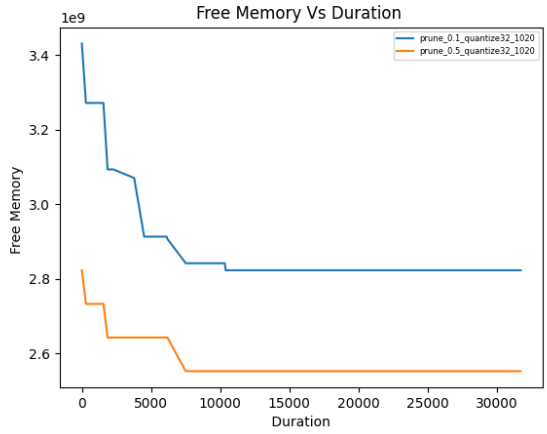
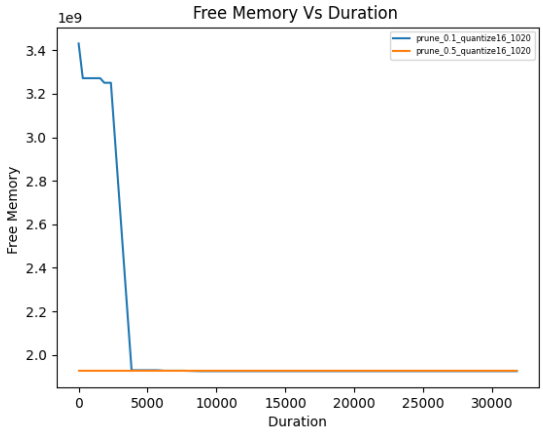


Figure 9: Malaria Compression Methods Average Performance Metrics

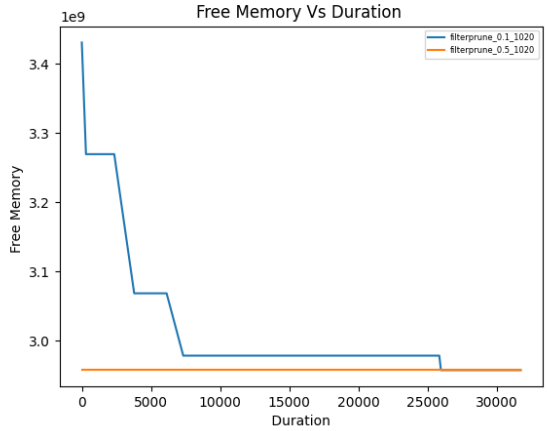
4.3 Chest-X-ray Network Density Results



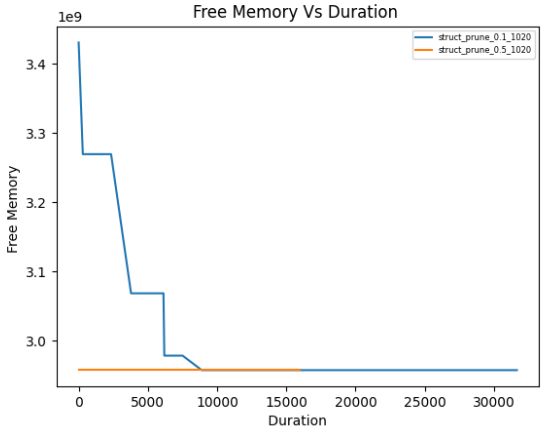
(a) Prune 0.1 vs 0.5



(b) PruneQuantize16 0.1 vs 0.5



(c) FilterPrune 0.1 vs 0.5



(d) StructPrune 0.1 vs. 0.5

Figure 10: Chest-X-ray: Free Memory (GB) vs Duration Densities: (0.1, 0.5), Methods: (Prune16, Prune32, FilterPrune, StructPrune)

The other dataset analyzed was the Chest-X-ray dataset. The dataset required a multi-classification approach as a patient could have more than one disease. Like the malaria dataset, different network densities were tested according to different performance metrics. In Figure 10, the average free

memory for each of the 0.1 density networks was higher or equivalent to the higher density network.

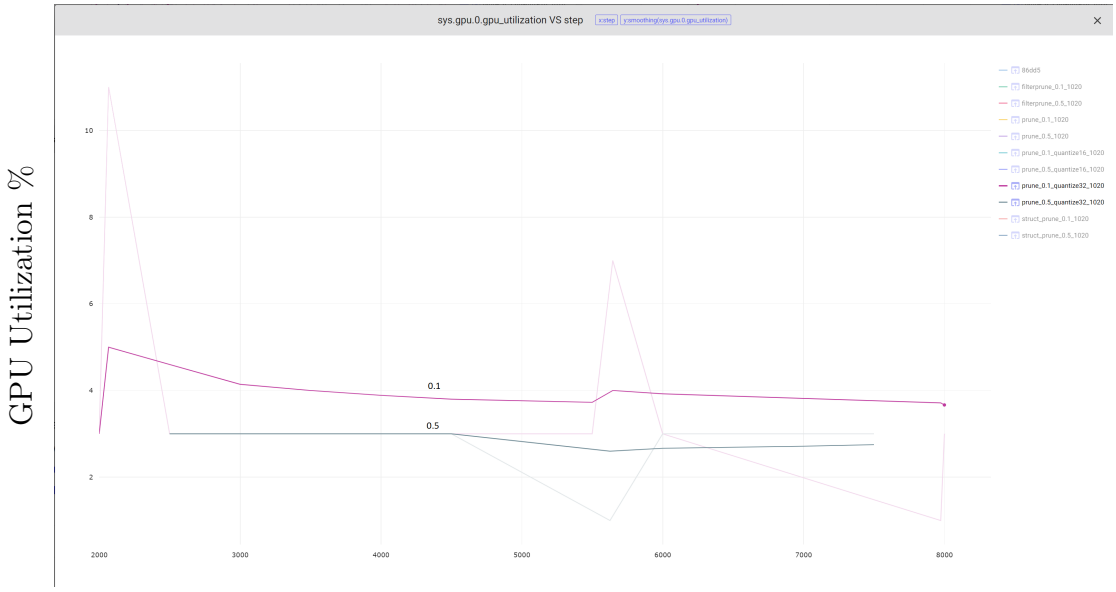


Figure 11: Chest-X-ray data set GPU Utilization vs. Condensa Training Duration with Prune32 0.1, 0.5 densities.

In contrast, the GPU utilization for the lower density network seemed to be higher. An example can be seen in Figure 11 for the PruneQuantize32 network comparing densities 0.1 and 0.5.

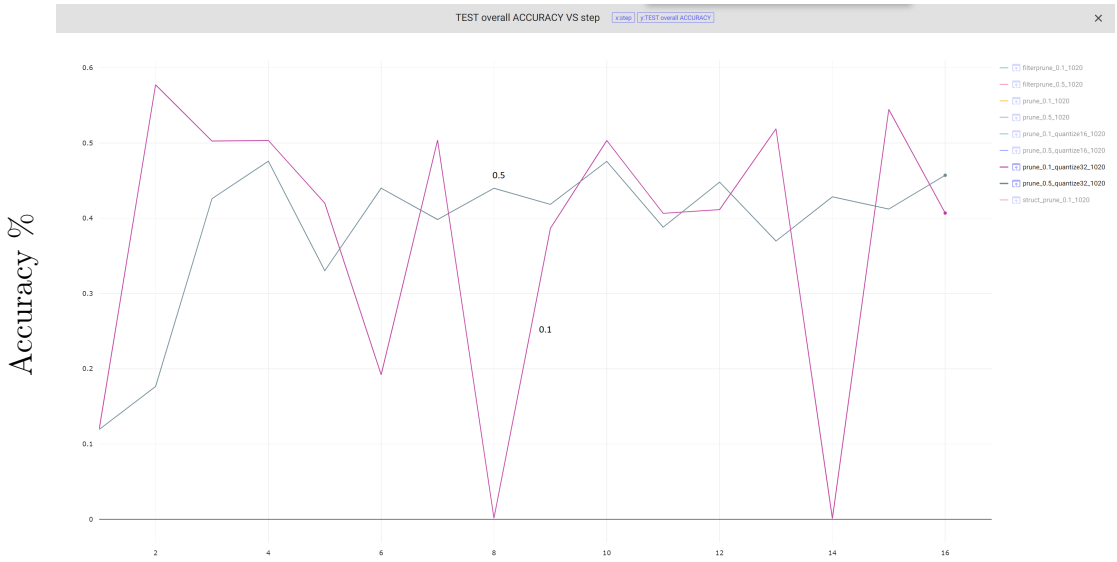


Figure 12: Chest-X-ray data set Accuracy vs. Condensa Training Steps with Prune32 0.1, 0.5 densities.

As expected, the accuracy of the higher density network remained more stable during training as can be seen in Figure 12.

Table 3: Chest-X-ray 0.1/0.5 Network Density Accuracies

Compression Method	Density	Accuracy
Prune	0.1	0.27
Prune	0.5	0.26
Prune_Quantize16	0.1	0.39
Prune_Quantize16	0.5	0.48
Prune_Quantize32	0.1	0.36
Prune_Quantize32	0.5	0.32

4.4 Chest-X-ray Compression Method Results

Table 4: Chest-X-ray Top-5 accuracy 0.1 density network

Lung Disease Class	Accuracy
Atelectasis	0.84
Cardiomegaly	0.96
Consolidation	0.92
Edema	0.97
Effusion	0.85
Emphysema	0.97
Fibrosis	0.98
Hernia	0.90
Infiltration	0.83
Mass	0.95
Nodule	0.90
PleuralThickening	0.97
Pneumonia	0.97
Pneumothorax	0.96

It should be noted that the Chest-X-ray dataset allowed for multi-class labelling. As a result, while the top-1 accuracy as depicted in Figure 13 was quite low for the chest-X-ray dataset, the individual class accuracies were much higher. Table 4 depicts the class accuracies in a typical 0.1 density network. Classes with lower accuracies had fewer samples. For example, Infiltration had the lowest accuracy because it had a much smaller number of data samples compared to the rest of the classes.

As with the malaria dataset, the Chest-X-ray dataset was tested using different compression schemes. As running times took substantially longer with a larger dataset, a more limited set of methods was tested. These include PruneQuantize16, PruneQuantize32, FilterPrune, and StructPrune.

In Figure 13, the highest overall accuracy was achieved by PruneQuantize16, and FilterPrune. StructPrune achieved very low accuracy as compared to the other methods. What it came to GPU utilization, PruneQuantize16 had the highest while StructPrune had the lowest GPU utilization.

PruneQuantize16 had the lowest memory usage but had a slightly higher GPU usage than all other networks.

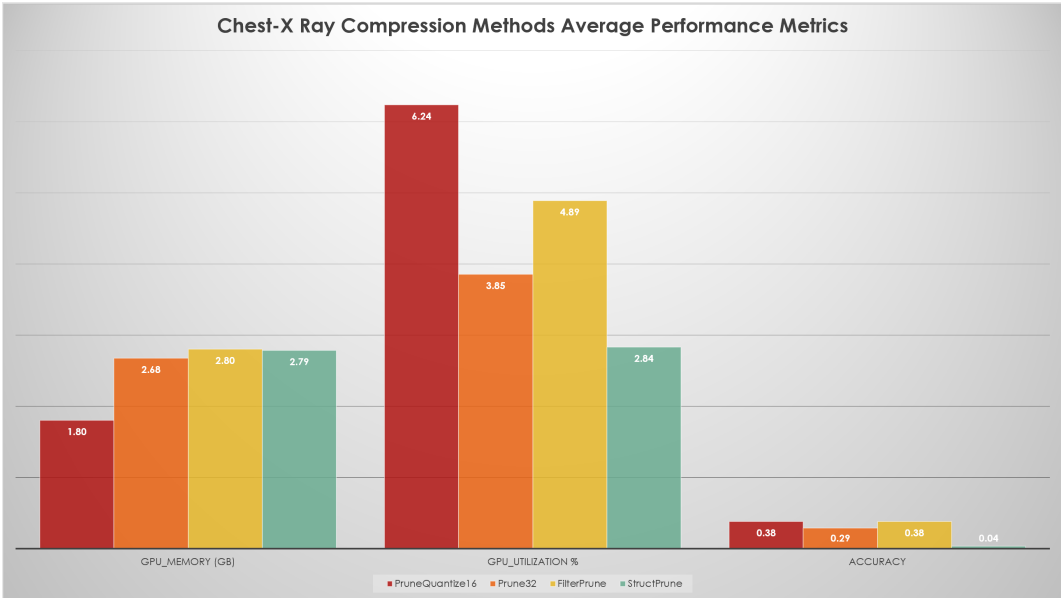


Figure 13: Chest-X-ray Compression Methods Average Performance Metrics

5 Discussion

5.1 Density Network Trends

In the malaria dataset, the smaller density networks appeared to have a lower GPU memory usage. However, they also had a slightly higher GPU percent usage. Similar accuracy could be attained for different densities. As Resnet-18 is a large network, it could be extensively compressed and maintain similar accuracy. Some smaller networks appeared to obtain greater accuracy than larger density networks. For example, the networks with a density of 1e-6 and 1e-7 appeared to achieve greater accuracy than a network with density of 0.1 (1e-1). However, networks with 1e-2, 1e-3, and 1e-4 appeared to achieve similar or lower accuracies compared to the baseline density of 0.1.

Similarly, the Chest-X-ray dataset networks had lower GPU memory usage for networks with a density of 0.1 as compared to networks with a density of 0.5. This trend was similar for networks compressed using Prune, PruneQuantize16, FilterPrune, and StructPrune compression schemes.

In a similar trend to the malaria dataset, the lower density networks also had a slightly higher GPU percent utilization. The lower density networks

had lower GPU memory usage. Also similar was the fact that networks with lower density were more erratic in accuracy recovery but could recover to a similar accuracy to their higher density counterparts. In contrast to the malaria dataset, the chest-X-ray dataset had varying accuracies between its different density networks with different compression schemes.

5.2 Compression Method Trends

Between both datasets, the StructPrune compression method had the lowest accuracy at low densities. However, StructPrune also achieved the lowest GPU utilization between both datasets. It had the best GPU memory utilization for the malaria dataset and had comparable memory performance to Prune and Filter prune in the case of the Chest-X-ray dataset. In both datasets, PruneQuantize16 used less memory than its counterpart.

6 Conclusion

In our work we have developed a script which can be adapted to process a variety of image datasets using the Condensa framework. It also provides throughput, latency, and GPU memory / utilization tests for generated networks. Experiments with the Malaria and Chest-X-ray dataset suggest relationships in how the density of a network affects GPU utilization and GPU memory usage. The experiments also suggested the optimum networks compression scheme for the respective datasets. In the future, our work will be to further improve the portability of our tool to other datasets with different classification schemes. Our goal is to gain understanding in how different compression schemes and network types perform in combination with the unique limitations of medical datasets.

References

[1] V. Joseph, S. Muralidharan, A. Garg, M. Garland, and G. Gopalakrishnan, "A Programmable Approach to Model Compression," *arXiv:1911.02497 [cs, stat]*, Nov. 2019, arXiv: 1911.02497. [Online]. Available: http://arxiv.org/abs/1911.02497

[2] D. A. Ghate, C. Jadhav, and N. Rani, "Automatic detection of malaria parasite from blood images," *Int J Comput Sci Appl*, vol. 1, 2012.

[3] S. Summers and S. Wang, "Machine learning and radiology," *Medical Image Analysis*, vol. 16, pp. 933–951, 07 2012.

[4] G. Castellano, A. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 8, pp. 519–530, 1997.

[5] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," *arXiv:1502.02551 [cs.LG]*, 2015, arXiv:1502.02551 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1502.02551

[6] J. O'Neill, "An overview of neural network compression," *arXiv:2006.03669 [cs.LG]*, 2020, arXiv:2006.03669 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2006.03669

[7] Y. Li, T. Ma, and H. Zhang, "Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations," *arXiv:1712.09203v5 [cs.LG]*, 2018, arXiv:1712.09203v5 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1712.09203

[8] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *arXiv:1912.02292*, 2020, arXiv:1912.02292 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1912.02292

[9] H. Song, M. Huizi, and J. William, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv:1510.00149 [cs.CV]*, 2016, arXiv:1510.00149 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1510.00149

[10] M. Hagiwara, "A simple and effective method for removal of hidden units and weights," *Neurocomputing*, vol. 6, pp. 207–218, 1994.

[11] S. Jaeger. Malaria datasets. 2018. [Online]. Available: https://lhncbc.nlm.nih.gov/LHC-downloads/downloads.htmlmalaria-datasets

[12] N. I. of Health. Nih clinical center. 2017. [Online]. Available: https://nihcc.app.box.com/v/ChestXray-NIHCC

[13] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," *arXiv:1705.02315 [cs.CV]*, 2017, arXiv:1705.02315 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1705.02315

[14] NVIDIA. Compression schemes. 2020. [Online]. Available: https://nvlabs.github.io/condensa/modules/schemes.html