Rethinking Compression within Deep Neural Networks

Sean Hooker

University of Warwick

September 9, 2020

Abstract

Compression techniques within Deep Neural Networks (DNN) are widely used to reduce computational costs whilst maintaining state-of-the-art performance. This research paper focuses on the impacts that compression techniques have on a class level as well as model performance. It is a natural follow on from the recent work done by Hooker et al. [2,5,24], where the authors found that certain classes within DNNs are more adversely affected than others once a compression technique has been applied. The compression techniques I look at are pruning and quantization which are used on Convolutional Neural Networks (CNN). Through my work, I reinforce Hooker et al. [2,5,24] work and show that compression techniques do in-fact hold a negative impact on certain classes and images. This is particularly the case with pruning which showed more significant damage to the CNNs. Analysis of the GPU usage of the pruned models during the experiments also showed a lack of efficiency. This cultivates the main body of my research which is to draw attention to the drawbacks associated with pruning in comparison to quantization.

I. Introduction

The demands to deploy DNNs onto resource-constrained environments (such as mobile phones) has grown dramatically in recent years. This falls in line with the increasing demand and use of these devices.

DNNs have proven to have state-of-the-art performance in various fields such as question answering, image recognition, and machine translation. These models have improved year in year out, driven by an increase in available data and computational power.

However, these models are also extremely demanding in terms of computational complexity and memory requirements. A majority of the DNNs deployed in recognizable areas contain millions of parameters requiring large amounts of storage, but the issues lie in deploying them to devices where storage is limited. Furthermore, battery demands required to run these networks on these devices is also extremely high due to inferences invoking $O(10^9)$ memory accesses and arithmetic operations [3].

Researchers are looking extensively to try and compress these neural networks to improve performance and efficiency whilst also preventing any significant loss in model quality. The compression of DNNs is a very promising field of research to address these issues. Popular methods of compression which have been proposed thus far include the approximation of weights, quantization, knowledge distillation, and network pruning [1].

DNNs which have been compressed via sparsity induction have yielded very promising results with one example showing it is possible to prune a ResNet-50 network to a 0.9 level (meaning that the number of weights within the network have been reduced by 90%) and still lose only 3% absolute top-1 test accuracy [4]. Out of these methods, network pruning has been seen as one of the most popular due

to its impressive performance and capabilities in reducing parameter size within a model with little or no loss in model quality.
However, a majority of literature [1,3,4] which analyzes these sparsity induction techniques mainly focus on limited metrics such as model accuracy.

i. Research Objectives

This research aims to analyze the impacts that compression techniques are having on CNNs at a more granular level. I take a look at the impact that both quantization and pruning have on VGG-19 models and the impact pruning has on ResNet-50 models. These models have been deployed on CIFAR-10 and CIFAR-100 datasets respectively. This body of research follows closely on from the work proposed by Hooker et al. [2,5,24]. They looked at the impact that model compression had on a class level rather than simply analysing how the overall models were affected. This led them to discover that certain classes (Compression Identified Exemplars - CIEs) are more adversely affected than others once a compression technique has been applied to a CNN.
My research furthers the work provided by Hooker et al. [2,5,24] by analysing the impact of compression using more metrics (such as f1-score and precision), visualising the inherent changes (via confusion matrices), and also analysing the apparent efficiency gains that pruning provides.

ii. Contributions

Applying widely used pruning and quantization techniques across classification tasks on CIFAR-10 and CIFAR-100 datasets provided consistent findings allowing me to show that:

1. Further classification tasks reinforce the results shown by Hooker et al. [2,5,24] showing that compression techniques harm certain classes.
2. Quantization provides far more encouraging signs of robustness regarding the impact on both model and class level accuracy compared to pruning.
3. The apparent efficiency benefits associated with pruning do not seem to be apparent during the experiments undertaken in this research paper.

iii. Significance of research

The main aim of this research is to further awareness regarding compression techniques and the impact they may have on certain tasks. Applying these compression techniques to sensitive domains such as healthcare, self-driving cars, and facial recognition could have serious consequences unless a full understanding of the implications is established. I hope that this research paper will allow users to gain a more holistic view of the compression technique they hope to utilize.
Addressing and solving the issue of CIEs could avoid serious issues concerning human welfare. Misclassification of CT (computed tomography) scans are a standout example of where CIEs can have significant implications.
Furthermore, DNNs have become ever-present in environmental quality analysis systems such as the "INTELLEnvQ" system [36]. These have proven to be hugely beneficial for monitoring environmental shifts and reporting on them within quick time. These tasks are huge and sparsity induction can benefit areas such as this greatly. However, these are extremely sensitive domains where miscalculations or misclassifications can have significant impacts on overall results. This again highlights the desire to gain a deeper understanding of CIEs.

iv. Structure

This paper begins by looking at two popular compression techniques used in deep learning: pruning and quantization. In section II. I provide background information on both techniques and how they work. I then introduce

CNNs in section III. to provide a background to the common architectures used in classification tasks.
In section IV. I discuss novel implementations of these techniques in wider literature which I have come across during my research. This then leads onto section V. which shows the specific techniques I have used during my research.
Section VI. specifically describes what constitutes as a CIE and the origins of this line of research. This is then followed by the current findings surrounding CIEs in other literature. I then discuss why these findings have inspired me to carry out this line of research in section VIII. and what I hoped to achieve before undertaking this project.
Sections IX., X. and XI. depict the experiment structure used for this body of research as well as the results and analysis. I then discuss areas in which this line of research can be taken further in section XII.
This paper is then concluded with an evaluation of how I felt my project has gone, a statement on the acknowledgement of ethics, and then final discussion/conclusion.

II. Compression techniques within DNNs

DNNs are often highly overparameterized and large that storage costs (in terms of memory) are extremely high. To overcome this issue, compression has been shown to be an extremely effective method. Below are two popular compression techniques and the basic idea behind them both.

i. Pruning

The main concept behind pruning is to induce sparsity within the network. Once sparsity has been induced the idea is that the output variable which is in focus can still be described using a reduced number of features within the input space [6]. This essentially means that a subset of the parameters within the model will not be used. Therefore the main goal associated with sparsity induction is to find the parameters which are most effective in describing the output.
Inducing sparsity within DNNs also allows them to be stored as sparse matrices which are more compact than their original size. This means that storage costs are also reduced.
One of the main computational costs which dominate DNNs is multiplication. To make these models more efficient regarding latency and inference, pruning causes them to essentially skip over a majority of the required multiplications. Inducing sparsity within the network causes a subset of parameters to have an assigned value of 0 and hence the costs associated with multiplication would be reduced.

ii. Quantization

Within a majority of DNNs, the standard numerical format used is 32-bit (FP32) floating points (with a dynamic range of the order of 1^{-38} to $1x10^{38}$) [7]. This numerical format causes efficiency issues regarding computation and storage. Quantization is a method of compression which essentially reduces the number of bits which represent a number (e.g. 8-bit integer between 0 and 255) [7].
Not only does reducing the bit-count representation of weights and activations cause little loss in model accuracy, but it is also much more area and energy-efficient (Figure 1) [8].

INT8 Operation	Energy Saving vs FP32	Area Saving vs FP32
Add	30x	116x
Multiply	18.5x	27x

Figure 1: *Improvements in energy and area efficiency caused by quantization [8].*

III. Convolutional Neural Networks

The tasks proposed in this research paper are based on image classification. The neural networks used to carry out these tasks are referred to as Convolutional Neural Networks (CNNs).

These have proven to be extremely successful in the field of visual imagery. A CNN consists of three main building blocks: convolutional layer, pooling layer, and fully-connected layer. These networks are inspired by biological systems. Their organization closely resembles that of an animal's visual cortex [19]. An illustration of a simple CNN can be seen in Figure 2.

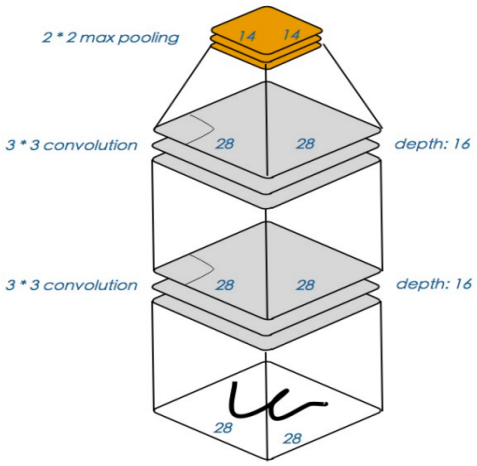

Figure 2: *Structure of a CNN [19]*

The convolutional layers act as a major influence as to why CNNs are so successful at tasks such as image classification. These layers are responsible for extracting features from an image via a set of filters. They also form part of what's known as the hidden layers of CNNs [19]. These layers combine via multiplication or dot product.

IV. Novel compression technique approaches for CNNs

i. Pruning

There are many different types of pruning. Some are based on simple rule-based approaches such as magnitude pruning, whilst others are more complicated in terms of both theory and application, such as variational dropout and L_0 regularization [4].
Magnitude pruning revolves around the weight associated with each parameter. The parameters with the lowest associated weights will be removed (pruned) according to the criterion set out during training. Successful implementation of this technique can be seen in the work done by Zhu et al. [3]. Their method essentially analyses the given weights in a layer within a DNN and sorts them in terms of their magnitude. The lower weights will then begin to have a "mask" placed over them until a desired level of sparsity is reached. This "mask" does not remove these lower weights but ensures that they are not included in the model. This is useful as these weights can then be "reactivated" at a later stage if the user desires. This method is also beneficial as it allows for a user-specified level of sparsity.
Variational dropout essentially performs variation inference on the parameters of a Gaussian posterior over the weights of the model which were under a log-uniform prior [4]. This method analyses which of the parameters hold the highest dropout rates through training. These parameters are then removed from the model. Molchanov et al. [20] provided a method of variational dropout where all possible values of dropout rates are evaluated and hence this leads to a sparse solution.
Finally, L_0 regularization is a method that encourages all weights during training to become exactly zero hence inducing sparsity [21]. However, the norm of weights (L_0) is non-differentiable and hence this regularization term can not be directly applied to an objective function (a function which is desired to optimize in this case). Louizos et al. [21] set out stochastic-gates sampled from hard concrete distributions which determine which of the parameters to set to zero. This is done through parameterising the weights within the DNN as the product of the weight and the stochastic-gate variable which has been sampled from the hard concrete distribution.

ii. 4-bit Quantization

Choukron et al. [22] proposed a 4-bit quantization framework that allowed for low-bit

precision inference without the need for full network retraining. This approach was very useful in terms of deploying these pre-trained quantized networks onto resource-constrained environments since the weights and activations were quantized to a very small number of bits (4).

A large issue with resource-constrained environments is that retraining DNNs on them is almost infeasible due to their memory demands. Hence, this paper [22] overcame this issue by presenting a framework that did not need a full network retraining scheme. They were able to accomplish this by formulating the linear quantization task as a Minimum Mean Square Error (MMSE) problem for both weights and activations [22]. This enabled the proposed framework to identify layers that were most sensitive to quantization error and account for these errors by implementing an adaptive quantization scheme that makes use of low precision tensors.

This paper [22] proved to be very successful in terms of top-1 accuracy results. They were able to achieve little loss in model accuracy whilst compressing the model size by up to 25x (in certain architectures). They also applied this framework to non-overparameterized models such as SqueezeNet and DenseNet that are typically a lot more sensitive to quantization. They produced highly competitive results in these domains with very little loss in model accuracy.

Banner et al. [23] proposed a very similar framework as Choukron et al. [22] in the sense that they suggested 4-bit quantized models which did not require full network retraining. Their approach did not need any fine-tuning requirements to regain model accuracy or indeed require the availability of the full data-set. This is particularly useful because in real-world scenarios most full data-sets are not available due to privacy issues [23]. This can be extremely beneficial when pre-trained models are not available with their accompanying dataset. Banner et al. [23] can deploy such a framework because they take advantage of the knowledge associated with the statistical characterization of neural network distributions. As a result, Banner et al. [23] were able to develop efficient quantization schemes that minimize the mean-squared error at the tensor level (again very similar to the work put forward by Choukron et al. [22]). Banner et al. [23] proposed three major contributions for post-training quantization:

- **Analytical Clipping for Integer Quantization (ACIQ)**: Limited the range of activation values within the tensor.
- 2: Allocated the optimal bit-width for each channel.
- **Bias-correction**: Correct for inherent bias in the mean and variance values of the quantized weights.

The results of these three methods can be seen in Figure 3 where these methods are applied to six different classification models.

PF 32 is the baseline 32-bit model and is used as a reference. A naive quantized model is also included for reference. It is clear to see that the most competitive performances across all models occur when the three post-training quantization techniques were applied in tangent to each other. There is also a clear peak in validation accuracy when 4-bits are used, with no apparent benefits in increasing bit count further than 4.

V. Compression techniques used in this body of research

The compression techniques I have decided to use for my research follow on closely from those used by Hooker et al. [2,5,24].

i. Magnitude pruning

Throughout their work, they utilized magnitude pruning which was the same form of magnitude pruning put forward by Zhu et al. [4]. For my research, I have also applied magnitude pruning but have used the Keras inbuilt pruning functionality instead. This form of pruning gradually zeros out insignificant model weights during the training process

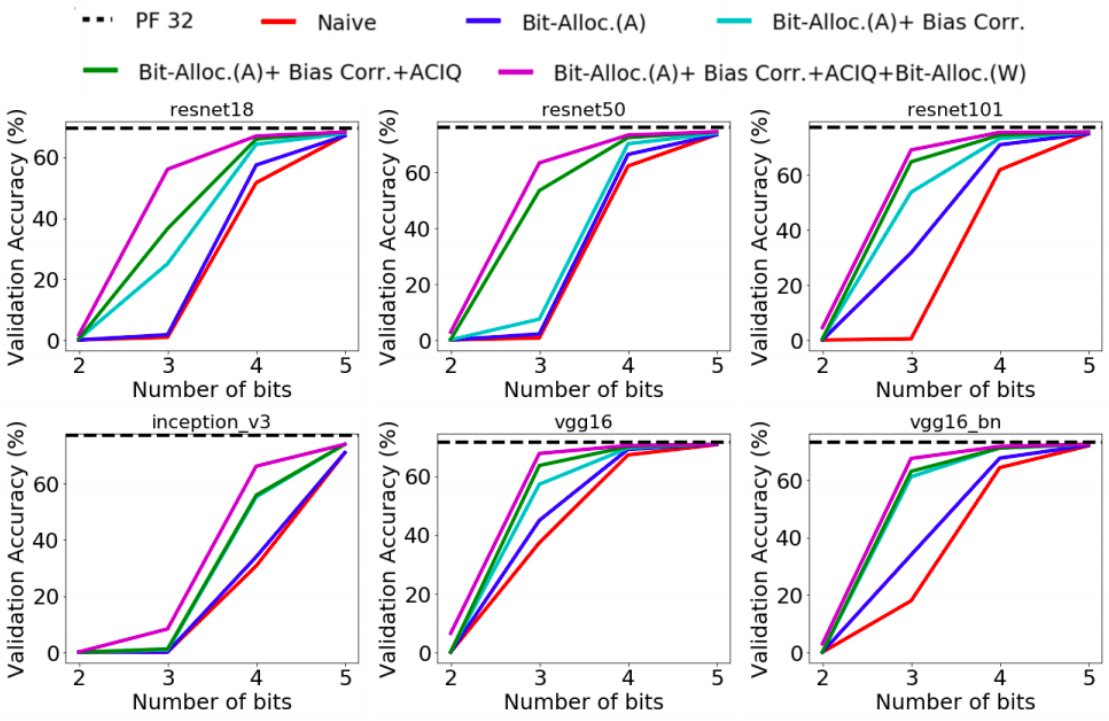

Figure 3: *Results of three post-training quantization methods [9].*

rather than placing a mask over them [25] (and hence there is no capability of reuse).

ii. Quantization

I had originally set out to apply the three quantization techniques used in the research put forward by Hooker et al. [5,24]. Unfortunately, I was only able to apply two of them to one classification task (on the CIFAR-10 dataset). This is due to the difficulties encountered in obtaining working algorithms to successfully quantize the models.
To discuss the work put forward by Hooker et al. [2,5,24] I have still depicted all three techniques below. These three quantization techniques are all applied post-training and hence reap the benefits of avoiding a full re-training (fine-tuning) regime which is required with pruning.

ii.1 Float16

This technique allows the user to reduce the numerical representation of the weights contained within the model from 32-bit floating-point numbers to 16-bit floating-point numbers [10]. The benefit of this is that the model will therefore be reduced by half whilst there is minimal loss of model accuracy [10]. However, this technique will not have a significant impact on the reduction of latency (in comparison with other quantization techniques).

ii.2 Dynamic range int8

This technique will reduce the numerical representation of the weights in the model from 32-bit floating points to 8-bit integers. However, at the inference level, these weights are then converted back to floating-point and then computed using floating-point kernels [10]. To avoid high latency costs associated with this conversion, the conversion is done only once and cached.

The reason this technique is described as dynamic relates to the fact that it can also quantize activation's down to 8-bits of precision based on their range. Not only this but the activations and weights are still converted back to floating-point and cached so that the speedup regarding the inference is still significantly improved.

ii.3 Fixed-point int8

This method ensures that all model math contained within the DNN is quantized to an 8-bit integer representation. This technique allows for further reductions in peak memory usage and further improvements in latency compared to *Dynamic range int8* [10].
Unfortunately, I was unable to perform this technique during my research.

VI. What are Compression Identified Exemplars?

The work put forward by Hooker. et al [2] looked at the effects of pruning at a lower level than typical model accuracy. The overall test-set accuracy after pruning may be at a high level but beneath that, this paper looked at how different classes are more adversely affected by pruning compared to others.
The author's initial paper on the topic [2] gauges the audience well by introducing an interesting comparative. It notes that during our childhood (from 2-10) we lose roughly 50% of our synapses in our brain and yet we are still able to function. They apply this logic to DNN pruning and ask the question *"What is lost when we prune a Deep Neural Network?"* [2] whilst also describing pruning on DNNs as *"selective brain damage"* [2].
Through their studies, they denote classes systematically more impacted by pruning as *pruning identified exemplars (PIE)* [2]. One of their most interesting discoveries was that both pruned and unpruned ResNet-50 models struggled when it came to identifying *PIEs*.

true: cloak
baseline model: gasmask
pruned model: breastplate

ladle
ladle
perfume

Figure 4: *Visualisation of PIE for the ImageNet dataset [2].*

From Figure 4 we can see that although the unpruned (baseline) model correctly identifies some classes/images, there are some classes that it still struggles with. The pruned model incorrectly classifies both, highlighting the original theory that different classes and images are more adversely affected by pruning compared to others. Reasons for both models struggling may be down to multiple objects being present in the image, image corruption, or an abstract representation (these all constitute as PIEs).
They furthered this body of research [5] by introducing quantization as a comparative compression technique used alongside pruning. This allowed them to analyze the apparent impact that these compression techniques were having whilst also enabling them to see if there were any differences or homogeneous traits between the two. Further to this, they proposed a framework which allowed users to mitigate the risks associated with these compression techniques. They proposed Compression Identified Exemplars (CIE) which was a human-in-the-loop auditing tool [24]. This is an evolution from PIE as it encompassed a wider scope of compression techniques. The CIE framework enabled users to become aware of certain subsets of data that may be vulnerable to these compression techniques and hence need further inspection or annotation by a domain expert [24].

VII. Relevant findings surrounding PIE

Hooker et al. [5] were able to find several classes that were disparately impacted as a result of compression techniques being applied to the CNNs. They undertook three classification tasks in their studies;

1. ResNet model trained and tested on a CIFAR-10 dataset.
2. ResNet-50 model trained and tested on ImageNet dataset.
3. ResNet-18 model trained and tested on a CelebA dataset.

When applying magnitude pruning they looked at 5 different levels of sparsity within the network [0.0, 0.3, 0.5, 0.7, 0.9]. At each level of sparsity, or quantization technique, they would assess whether or not certain classes were more adversely affected by compression than others (in terms of top-1 and top-5 accuracy).
Figure 5 [5] shows the ImageNet classes which were impacted (plum bars) in classification as a result of either pruning or quantization. The green scatter points highlight the normalized recall difference (which is normalized by the overall change in the model accuracy) whereas the bars are the absolute difference [5].

An interesting observation from Figure 5 is some classes improve as a result of the compression technique. One suggestion that was noted by Hooker et al. [5] was that the compressed networks seemed to cannibalize performance on a certain subset of classes to maintain overall top-line metrics. This is an interesting approach to explaining why we see such competitive model performance after these compression techniques have been implemented.
However, what is clear is that there is still a larger average class decrease compared to an increase. Further analysis of this impact can be seen in Figure 6 [5].

Figure 6 relates to the ImageNet classification task with various compression techniques applied to the ResNet-50 model. It shows the top-1 accuracy, top-5 accuracy, number of classes that have been impacted by compression, and the number of images that have been impacted within the classes. When looking at pruning, as the levels of sparsity increase it is clear to see that the number of impacted classes also increases. However, Hooker et al. [5] did not comment on the relationship of these increases in great detail.

i. Pruning results

Looking at the relationship between the level of sparsity and the number of impacted classes in Figure 7 we can see that up until the 0.5 sparsity level there seems to be a fairly linear relationship.
Past this point, the number of impacted classes appears to increase exponentially. This may highlight the dangers of increasing sparsity levels beyond the 0.5 level. Figure 8 has similar results. Here the number of impacted images (PIEs) are plotted against increasing sparsity levels.

We can see that the number of images that were deemed to be examples of PIE plateaus around the 0.5 sparsity level before increasing in an almost exponential fashion once larger levels of sparsity were introduced. This could be an interesting area to explore further with the benefits of pruning past 0.5 being weighed against the apparent cons associated with the increased presence of CIE.

ii. Quantization results

Applying **float16** quantization to the network had the most promising results. There was a negligible loss in top-1 accuracy compared to the full model (-0.03%) and the top-5 accuracy remained the same. On top of this, this technique also exhibited the fewest affected classes across all compression techniques.
A few interesting observations lie also in the number of CIE examples. If we compare **float16** to magnitude pruning at 30% we see that although it exhibits 23% fewer affected

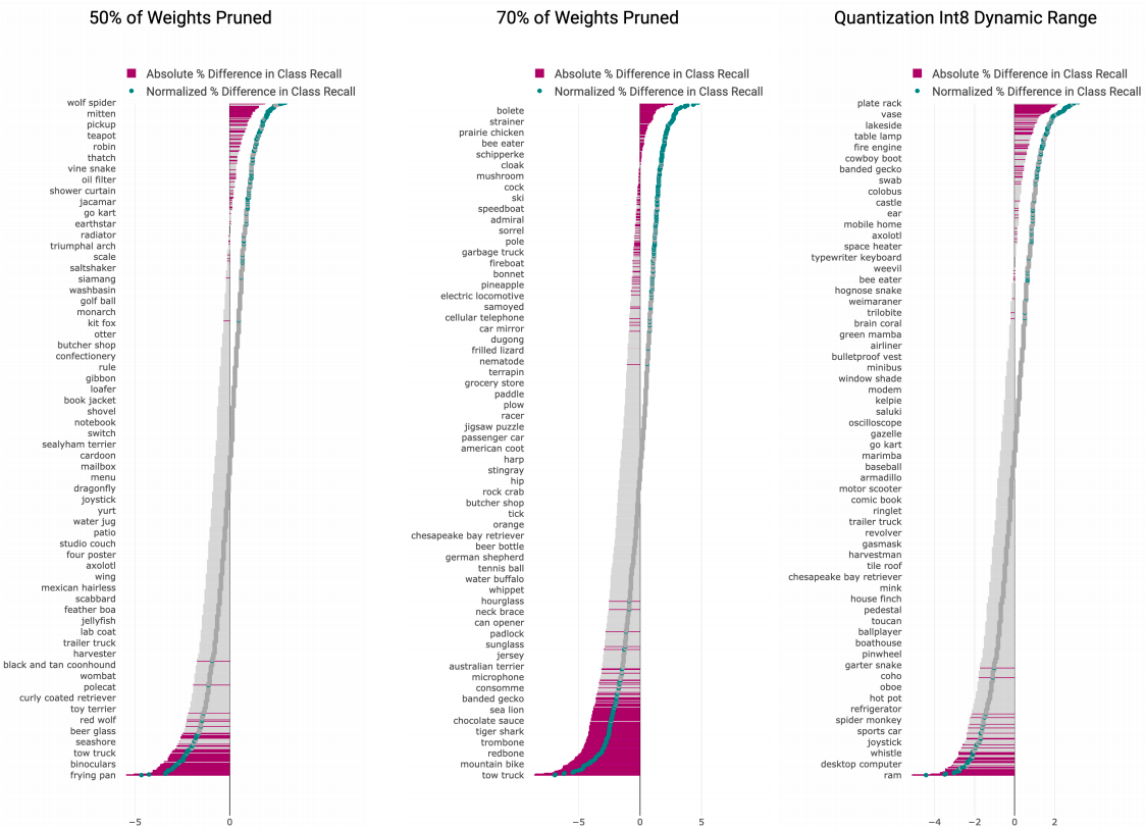

Figure 5: *ImageNet classes impacted as a result of compression techniques [5].*

Fraction Pruned	Top 1	Top 5	Count Signif Classes	Count PIEs
0	76.68	93.25	-	-
30	76.46	93.17	69	1,819
50	75.87	92.86	145	2,193
70	75.02	92.43	317	3,073
90	72.60	91.10	582	5,136
Quantization				
float16	76.65	93.25	53	2019
dynamic range int8	76.10	92.94	121	2193
fixed-point int8	76.46	93.16	113	2093

Figure 6: *ResNet-50 model performance metrics before and after compression techniques have been applied. This table also highlights the number of impacted classes and images (CIEs) as a result of the compression technique [5].*

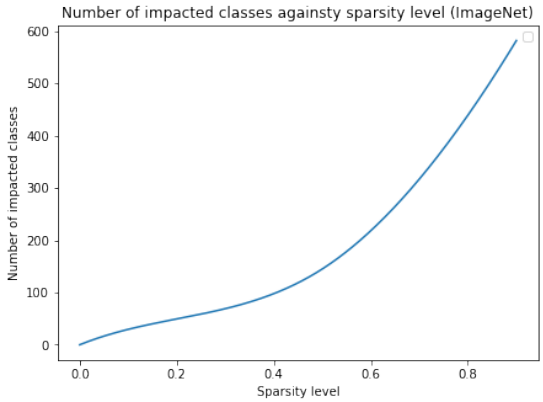

Figure 7: *The number of impacted classes as a result of pruning against increasing sparsity levels (ImageNet dataset).*

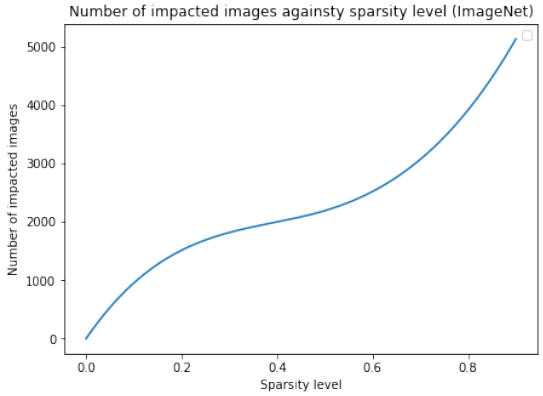

Figure 8: *The number of impacted images as a result of pruning (CIEs) against increasing sparsity levels (ImageNet dataset).*

classes, it does indeed have 11% **more** examples of CIE. This is maybe an indication that there may be fewer classes affected by this form of compression but they are more severely affected. It is however very difficult to directly compare both quantization to pruning as they both induce different properties within the networks. However, if we were to analyze at a higher level and essentially look at the model being reduced by a factor of 2x then a natural comparison would fall between **float16** and **50% magnitude pruning**. Here we see a similar trend again, there are 173% more affected classes as a result of 50% pruning compared to **float16**, however, there are only 9% more CIE examples. This is very insightful in showing that pruning may be a more damaging technique in terms of increased spread of affected classes however quantization has more of a concentrated impact on the classes it affects.

VIII. Motivations behind extending this area of research

Hooker et al. [2,5,24] furthered their earlier work [2] by looking at quantization in the context of PIE. This was very informative as it enabled high-level comparisons to be made between the techniques. It also reinforced their initial findings that some classes are more adversely affected than others. However, this area of research is still very much in its early stages and hence I have decided to continue it by analysing further areas.

i. Improving efficiency

One of my main goals associated with this project was to provide a pipeline that allowed users to gain a high-level understanding of how impacted their classification models were as a result of the compression technique they were planning on using. This motivation stems from the current framework surrounding CIE detection.

Hooker et al. [2,5,24] introduced a hypothesis test to detect whether or not a class had been affected by the introduction of a compression technique. Their null hypothesis stated that the difference between the class and model accuracy before and after compression should remain the same whilst their alternative hypothesis stated that compression would cause this difference to alter significantly.

To test this hypothesis they trained 30 independent models on each task and across each level of sparsity and quantization technique. For example, on 30% sparsity induction there were a total of 90 models trained (30 models for each task, 3 tasks were undertaken). All of these models were trained from scratch on each of their respective datasets. These models

were then randomly sampled from, to assess the hypothesis in question. To test the significance concerning a shift in class/model accuracy they applied a Welch's two-tailed t-test. All of this amounted to a CIE detection schema which was extremely demanding in terms of time and hardware.
The pipeline I have introduced requires training only one model. Once the compression technique is applied, an analysis of the number of impacted classes is then produced via my proposed pipeline. Although this pipeline is a far less demanding CIE detection schema (in terms of hardware), than the one proposed by Hooker et al. [2,5,24], it must be noted that the results are less robust to errors given that they are not obtained via populations. However, my results do have homogeneous traits to the ones displayed in [2,5,24] and hence could be seen as a cost-efficient method of obtaining CIE awareness regarding a compressed model.

ii. Further investigation on quantization

Hooker et al. [5] introduced the first comparisons between quantization and pruning regarding the impact they have on certain classes. Preliminary results from their findings have suggested that there may be an apparent benefit in using quantization over pruning as a compression technique. This is based on the apparent consistency in the number of affected images and the lower count of impacted classes.
In this body of research, I try and provide further clarity on this argument by extending the tasks to the CIFAR-10 dataset which includes comparatives between the two techniques. As a result, I am undertaking similar experiments but on the CIFAR-10 and CIFAR-100 datasets. However, the CIFAR-100 experiments do not have any quantization comparatives. These are relatively small datasets that contain only 10 and 100 classes respectively but allows for a more in-depth view of the classes particularly affected. Because of the size of CIFAR-10, I have also been able to utilize visualization techniques which allow for a clearer understanding of the impacted classes.

iii. Understanding the benefits associated with further compressing these models

The ultimate goal associated with compression is to reduce the model size to improve computational costs and storage capabilities. I believe highlighting the actual improvements in inference speeds and compression ratios alongside performance metrics (such as precision, recall, and f1-score) will allow users to get a good overview on the negatives and positives associated with the compression technique they are planning to use.
As we have seen in Figure 7 and Figure 8 there seems to be a linear increase in the number of impacted classes against increasing sparsity levels within the network. This linear relationship holds to a certain point (50%) before seemingly turning into an exponential relationship. There are still huge benefits associated with reducing the model by 50%. However, if a user was thinking of reducing the model size further than 50% then it would be beneficial to them to receive information regarding how much that will improve their model in terms of compute and size, and how much more of an impact that will have on their model in terms of performance metrics.
To provide users with an idea of the apparent benefits of utilising magnitude pruning, regarding efficiency, I have analyzed the GPU statistics during the training stage of my experiments. This will allow users to see the apparent efficiency gains associated with pruning their models at each level of sparsity. For quantization, I will provide an overview of the compression ratios and inference benefits. No GPU statistics will be provided for the quantization techniques as they are all applied post-training.

IX. Experiment setup

My research consisted of two classification tasks. These tasks can be summarized below.

1. VGG-19 model trained and tested on a CIFAR-10 dataset.

2. ResNet-50 model trained and tested on a CIFAR-100 dataset.

For both models, I utilized the transfer learning regimes available on Keras. This was mainly done to prevent large scale training regimes whilst also utilising models that have proven capabilities on both datasets.

i. What is transfer learning?

This is a machine learning technique in which a previously trained model is re-purposed on a new task [11]. These models could have been previously trained on large datasets. This means that reusing them is beneficial in terms of avoiding large scale training regimes which are hugely demanding in terms of storage and hardware. Not only this, but the pre-trained models also exhibit strong performance metrics regarding the tasks they were designed for. With regards to my project I am using pre-trained **VGG-19** and **ResNet-50** models. These models were loaded in using a Keras functionality. Once loaded, the feature extraction layers (convolutional layers) were frozen whilst new fully connected layers were added for training. What this means is that I will obtain all of the pre-trained convolutional layers with the parameters fixed (i.e. there will be no changes to the weights and activation's) but adding in my own fully connected layers makes the model applicable to the dataset I am working on. Once the fully connected layers were added, I trained the VGG-19 model on the CIFAR-10 dataset and the ResNet-50 model on the CIFAR-100 dataset to obtain the optimal parameters for their respective fully connected layers.

ii. VGG-19

The VGG-19 model is a CNN that consists of 19 layers within the network. These are 16 convolutional layers (which enable feature extraction capabilities) and 3 fully connected layers [12]. On top of this, there are also 5 MaxPool layers (to reduce dimensionality) and 1 SoftMax layer (to convert the output into a probability distribution). An outline of the architecture can be seen in Figure 9 [13].

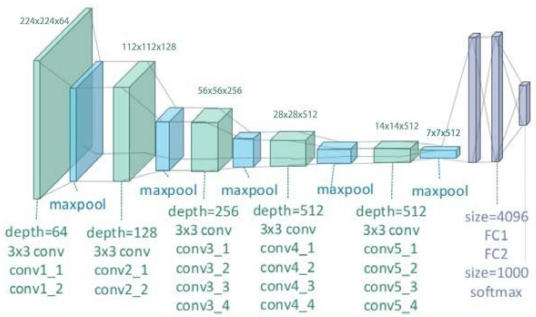

Figure 9: *VGG-19 Architecture [13].*

VGG stands for **Visual Geometry Group** which originates from **Oxford**. The group at Oxford came out with this model as a successor to **AlexNet** which was released in 2012 and is widely regarded as the network which significantly improved the performance and reputation of CNNs.
I have chosen this model to assess as it has a simple architecture of which can be easily analyzed but also produces strong performance results on classification tasks.

iii. ResNet-50

The ResNet-50 model derives from the ResNet model family, otherwise known as Residual Networks. The *50* denotes how many layers deep the model is.
Residual Networks gained large popularity after claiming success in the ImageNet challenge in 2015 [26]. The particular network which one this competition was the ResNet-152 model which is a deeper model compared to the ResNet-50.

The motivation to design residual networks lies in the issues associated with creating models with large depth. When models become too deep and more layers are stacked on top of each other, issues such as vanishing gradients can arise. These occur when the gradient is back-propagated in the earlier layers and repeated multiplication causes the gradient to become extremely small [26].
Residual networks overcome this issue by introducing skip connections in the network. Skip connections essentially allow an alternative path for the gradient. These paths will skip some layers of the network by feeding the output of one layer as the input to the next layers (avoiding the next immediate layer) [27]. Within ResNet models, depth is maintained my stacking these skip residual blocks on top of each other. What will then happen is that the gradient value will be maintained in the earlier layers which mitigates against the risk of vanishing gradients. Figure 10 [28] depicts a typical residual skip connection which is an essential building block of the ResNet models.

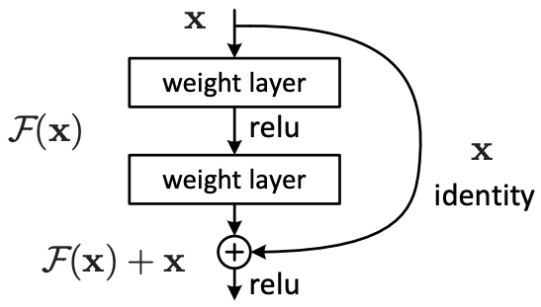

Figure 10: *Residual learning block present within ResNet architectures [28].*

I chose this architecture due to its cited competitive performance on classification tasks but also to analyze any comparisons between the VGG-19 model. The ResNet-50 model is noticeably larger with far more layers and hence could propose interesting similarities or differences with the VGG-19 model in the context of these experiments

iv. CIFAR-10 and CIFAR-100

The CIFAR-10 and CIFAR-100 datasets originate from the **Canadian Institute For Advanced Research**. The CIFAR-10 dataset contains 10 classes of images with 6000 images in each class. Each image is a 32x32 colour image [14]. There are a total of 50,000 training images and 10,000 test images. The CIFAR-100 dataset contains images with the same structure to CIFAR-10 and also the same quantity of images (60000). However, there are a total of 100 classes included within this dataset and each class contains 600 images (500 training and 100 testings).
The reason I have chosen these datasets to perform experiments on is that they are extremely well-explored datasets, and hence contain a wealth of documentation and benchmarks. On top of this, the small set of classes means that a more thorough analysis and visualization (regarding CIFAR-10) can be done regarding the impact of compression techniques. A typical image that can be found in both datasets can be seen in Figure 11.

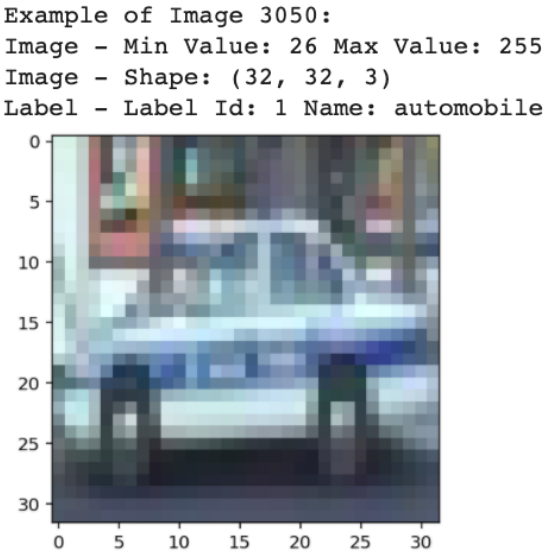

Figure 11: *An example of a typical image contained within the CIFAR-10 and CIFAR-100 datasets.*

v. Training regimes

Both the VGG-19 and ResNet-50 models were compiled using a cross entropy loss (CEL) function. This will assess the loss between the prediction the model makes and the true label of the unseen data based on the following equation.

$$Loss = -\sum_{i=1}^{outputsize} y_i \cdot log\hat{y}_i \quad (1)$$

Where y_i is the true label and $\hat{y}_i$ is the models prediction.
To minimize the CEL, the model parameters (specifically contained within the final 3 fully connected layers) will need to be updated throughout the training. These updates are intended to reduce the CEL. For this to occur, an optimization function needs to be specified. Stochastic gradient descent (SGD) was the optimization function used for the VGG-19 model whilst adaptive moment estimation (Adam) was used for the ResNet-50 model.
SGD works by updating the weights based on going against the gradient of the loss (cost) function. Each weight update is then applied and the CEL between the predicted and true values are recalculated to see if there are any improvements. This operation is iterated over several times before a global cost minimum is reached. The term "stochastic" refers to the path in which the optimizer takes to reach the global cost minimum. When visualising the cost on a 2D plane, SGD can be thought of like taking a "zig-zag" approach in finding the global minimum. This can be seen in Figure 12 [15]. This "random" approach is thought to be more beneficial in finding global minimums rather than potentially getting stuck at a local minimum using regular gradient descent.

Adam differs from SGD in the sense that it applies an adaptive learning rate. The learning rate is a hyperparameter that dictates the subsequent change (regarding magnitude) that the model takes in response to the estimated errors once the weights have been updated [29]. Small learning rates may result in the training regime becoming stuck whilst large learning rates may cause sub-optimal weights to be assigned [29]. Adam's adaptive learning rate allows for individual learning rates to be calculated for different parameters. This is done by using the estimations of the first and second moments of the gradient to adapt the learning rate for each weight within the network [30].

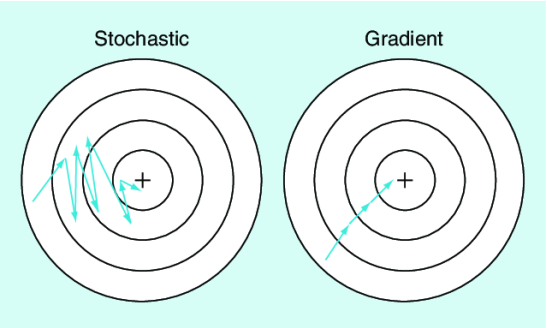

Figure 12: *SGD approach to finding the global cost minimum compared to regular gradient descent [15].*

Once the loss function and optimizer have been specified the models are ready for training. I decided to train the VGG-19 model over 50 epochs (life cycles) with validation/training accuracy and CEL being evaluated at each epoch. The reason a large number of epochs were chosen for this regime is regarding analysing the GPU statistics over an extended period. This is to analyze the efficiency associated with pruning the VGG-19 model at various levels of sparsity. The ResNet-50 model was trained for 15 epochs. It is a larger model and hence, given time constraints, 15 epochs allowed for adequate efficiency analysis whilst also training a strong performing model.
The CEL and model accuracy metrics were recorded during the training process of each respective model. The results of the uncompressed models training regimes can be seen in Figures 13, 14, 15, and 16.

Initial analysis of both training regimes shows that performance seems to plateau at the 10th (for VGG-19) and 7th (for ResNet-50)

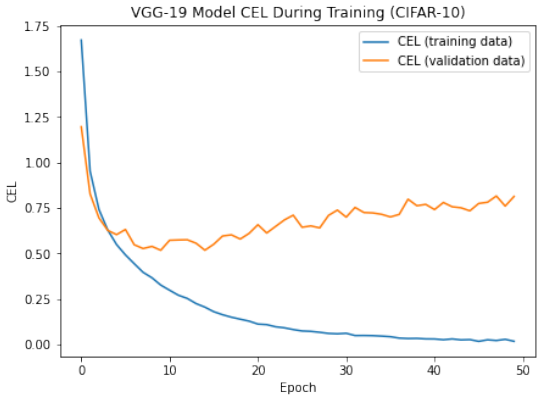

Figure 13: *The CEL between the predicted labels from the VGG-19 model and the true labels during the course of training.*

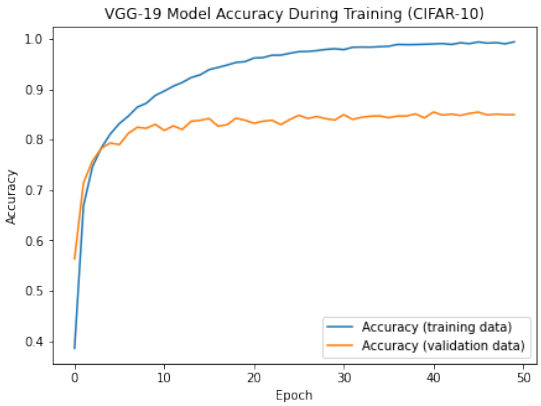

Figure 14: *The accuracy of the VGG-19 model during the course of training.*

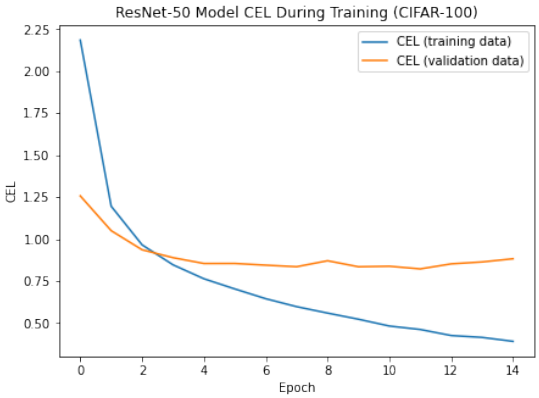

Figure 15: *The CEL between the predicted labels from the ResNet-50 model and the true labels during the course of training.*

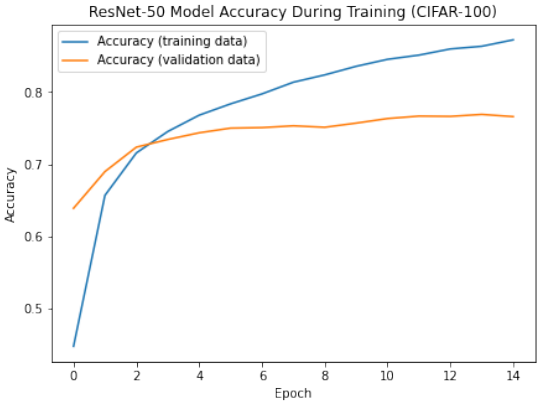

Figure 16: *The accuracy of the ResNet-50 model during the course of training.*

epoch. As previously stated, the reason the training regimes were allowed to carry on beyond these points was to analyze efficiency metrics over multiple life cycles. The overarching goal of this research is to see the potential damage that compression techniques may be causing. As a result, the emphasis to obtain state-of-the-art benchmarks was not a driver to the success of this research.

vi. Metrics being analyzed

To analyze the performance of the classification models I have looked at the following performance metrics:

- Top-1 Accuracy: This calculates the average number of correct predictions. Top-1 refers to only observing model answers with the highest probability that match exactly to the expected answer.
- Precision: This is the proportion of positive identifications of each class that were correct [16].
- Recall: This looks at what proportion of actual positive classifications (i.e. classifying the class in question) were identified correctly [16].
- F1 score: This is the weighted harmonic mean of precision and recall [17].

Each metric has its benefits. Top-1 accuracy

is a very popular metric and one that has been widely used to determine the success of compression techniques in DNNs. However, I believe I am extending compression research further by analysing the other three metrics in tangent with top-1 accuracy.
Precision is very useful for those users looking to deploy models in environments where the costs associated with false positives (i.e. incorrectly predicting a negative class) is high. An example of a false positive would be a non-spam email (actual negative) has been identified as spam (predicted spam) [18]. The recall metric is desirable for users who are deploying models in environments where the costs associated with false negatives [18]. A clear example of this would be classifying a positive CT scan as negative. The f1 score allows for a balance between both precision and recall.
Due to this being a multi-class problem, I will be looking at the micro average of each metric (apart from top-1 accuracy). A micro average will aggregate the contributions of all classes to compute the metric. This is useful when there is class imbalance present within the dataset as micro averaging adequately captures this. However, it must be noted that all classes within the CIFAR-10 and CIFAR-100 datasets are approximately all represented equally. These means that there will be little difference between micro, macro or weighted averaging. The reason micro is chosen now is for consistency reasons if a user/I decide to further this research agenda and move onto different datasets which may exhibit class imbalances.

vii. CIE detection schema

Unlike the CIE detection mechanism proposed by Hooker et al. [2,5,24], this body of work introduces a pipeline which requires only a single model for detecting impacted classes.
This is done by analysing the standard deviations before and after compression has been applied. A full breakdown of the pipeline is listed below.

1. The original uncompressed model accuracy is calculated along with every class accuracy.
2. The standard deviation for the uncompressed model is calculated and used a significant class boundary.
3. All classes which have a deviation from the model accuracy larger than the magnitude of the standard deviation (in terms of absolute value) are recorded and stored in a set (denoted **x** for the sake of explanation).
4. The idea is that these recorded classes should remain above the standard deviation boundary before and after compression.
5. After compression is applied to the model, the same process is applied again to determine which classes are currently above the standard deviation boundary. These classes are then added to a set **y**.
6. Any classes which were previously in set **x** and are no longer in set **y** are classed as significant and any classes which were not present in set **x** but are now present in set **y** are also classed as significant.
7. The number of significant/impacted images (CIEs) is calculated by the sum of the absolute differences in the deviations each class has from the model accuracy before and after compression. An example of this would be: class A deviates 3% before compression and then 1% after compression. Therefore the number of impacted images constitutes as 2% of the total images contained within that class.

X. VGG-19 Results

i. Uncompressed model results

The VGG-19 model results on an unseen test dataset of 10,000 images can be seen in Table 1.

	Top 1	Micro Precision	Micro Recall	Micro F-1
VGG 19 uncompressed	85.87%	0.86	0.86	0.86

Table 1: *VGG-19 uncompressed model performance metrics on CIFAR-10 unseen test data.*

The first thing to note is the similarity between all of the metrics, this is due to the class balance displayed within the CIFAR-10 dataset. This class balance can be seen in Figure 17.

```
# of Samples: 10000

Label Counts of [0](AIRPLANE) : 1014
Label Counts of [1](AUTOMOBILE) : 1014
Label Counts of [2](BIRD) : 952
Label Counts of [3](CAT) : 1016
Label Counts of [4](DEER) : 997
Label Counts of [5](DOG) : 1025
Label Counts of [6](FROG) : 980
Label Counts of [7](HORSE) : 977
Label Counts of [8](SHIP) : 1003
Label Counts of [9](TRUCK) : 1022
```

Figure 17: *Class balance seen within the 10,000 CIFAR-10 testing images.*

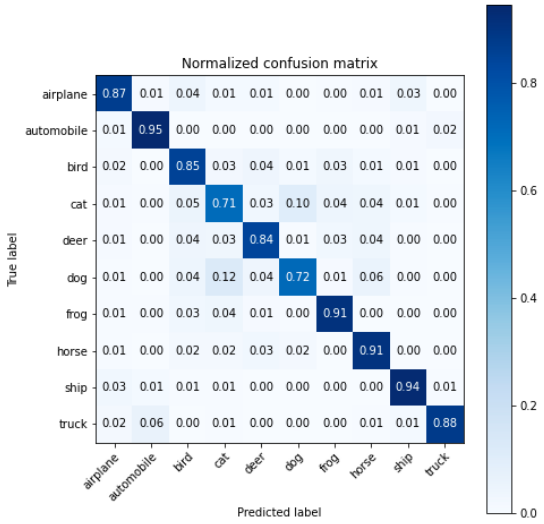

Figure 18: *Normalized confusion matrix of an uncompressed VGG-19 model on unseen CIFAR-10 test data.*

Overall the VGG-19 model performs well with strong metrics to show for it. It is however beneficial to look at how the model performs for each class. In Figure 18 we can see the normalized confusion matrix associated with each class. A confusion matrix allows for the visualization of the true labels vs the models predicted labels. This allows us to see the volume of correct/incorrect predictions associated with each class and where these predictions go.

Overall the VGG-19 model does well in classifying a majority of the classes (based on recall). We can however see a slight dip in performance when it attempts to classify images of cats and dogs. Here we only see a recall of 0.71 and 0.72 respectively, compared to the next lowest class (*deer*) at 0.84. As we can see, a majority of the incorrect classifications for both of these classes lie in classifying them as each other (i.e. a dog is misclassified as a cat and visa versa). This may be down to the similarities in the symmetry between these animals (regarding facial features). As a result the model, on occasion, struggles to distinguish between them.

Further analysis of the model's performance in each class can be seen in Figure 19. Like the confusion matrix highlighted, the model seems to be performing relatively poorly on classifying the *cat* and *dog* classes with f1 scores of 0.72 and 0.77 respectively. The model obtains a low recall values for the *cat* and *dog* class (0.71 and 0.72 respectively) indicating that when an image is in fact a cat/dog it will classify it as something else (i.e. false positive). This again reinforces the suspicions that the model is struggling to distinguish between images of cats and dogs.

ii. VGG-19 pruning results

Magnitude pruning was applied to independent VGG-19 models across four levels of spar-

Classification Report

	precision	recall	f1-score
airplane	0.89	0.87	0.88
automobile	0.91	0.95	0.93
bird	0.79	0.85	0.82
cat	0.74	0.71	0.72
deer	0.85	0.84	0.84
dog	0.83	0.72	0.77
frog	0.88	0.91	0.89
horse	0.85	0.91	0.88
ship	0.91	0.94	0.93
truck	0.96	0.88	0.92
accuracy			0.86
macro avg	0.86	0.86	0.86
weighted avg	0.86	0.86	0.86

Figure 19: *Classification analysis of VGG-19 model on unseen CIFAR-10 data.*

sity, [0.3, 0.5, 0.7, 0.9]. At each level of sparsity, the following results were recorded.

- Training performance regarding accuracy and loss on validation data.
- GPU utilization performance during training.
- Performance metrics on unseen test data (accuracy, precision, recall and f1-score).
- Classes and images impacted as a result of pruning.

ii.1 Training

Figure 20 and 21 shows the performance of five independent VGG-19 models during the training regime (with regards to validation accuracy and CEL). Each model has a different level of sparsity induced within the network ranging from 0% (the uncompressed model) to 90%.

From Figure 20 During the first 10 epochs, there appear to be varying performances from all of the models. This is most likely down to noise within the networks which causes variation in initial performances.
Figure 21 is interesting in the fact that the most highly pruned model ended up with the lowest CEL. This is because past the 10th epoch the generalisation performance of the other models starts to degrade due to overfitting. Below the 90% sparsity level the model remains over-parameterized and hence requires far fewer

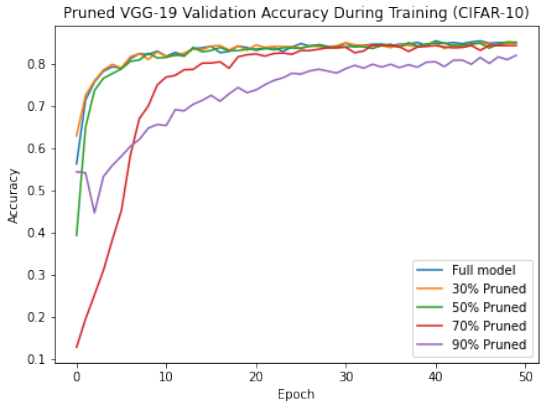

Figure 20: *The validation accuracy of five independent VGG-19 models during training. Each model has a different level of sparsity induced within the network.*

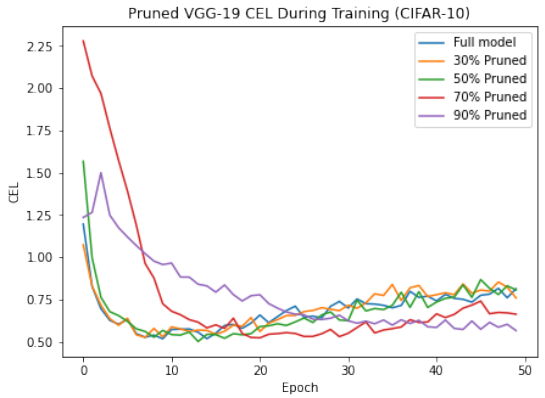

Figure 21: *The CEL of five independent VGG-19 models during training. Each model has a different level of sparsity induced within the network.*

epochs before the model begins to overfit the training data. At the 90% level, the model has far fewer parameters and hence the network is more regularized.
Looking at Figure 20 we can also see that past 10 epochs nearly all of the model's plateau into a steady validation accuracy. We can see that all models with 70% or less sparsity induced within them, produce almost identical validation accuracy's. Final validation accuracy's for all models can be seen in Table 2.

The models which had 30% and 50% sparsity induced within them outperformed the origi-

Sparsity Level	0.00	0.30	0.50	0.70	0.90
Top 1 Validation Accuracy	84.93%	84.98%	85.13%	84.35%	82.02%

Table 2: *VGG-19 final validation accuracy after training. Each independent VGG-19 model had a varying level of sparsity induced within the model. All models validation accuracy was assessed on CIFAR-10 validation data.*

nal uncompressed model. This mainly because a VGG-19 model would be considered highly overparameterized for a classification task on the CIFAR-10 dataset. As a result, pruning at these levels acts as a regularizer to the network and improves generalization performance on unseen data.
Past the 50% level of sparsity, we do see a drop in performance but only marginal. The worst performing model was the VGG-19 model with a sparsity level of 90% but it only saw a change -2.91%.

ii.2 GPU Utiliziation

GPU's, otherwise known as Graphics Processing Units, are hardware devices which dramatically improve the efficiency associated with training DNNs. They were originally designed to perform calculations relating to 3D computer graphics but also include 2D acceleration and framebuffer capabilities [31]. These devices are used to perform extensive mathematical and graphical computations which frees up the CPU (Core Processing Unit) cycles to perform other jobs [32]. This in turn dramatically increases the speed and efficiency of certain operations if a GPU is utilized.
Originally this device was not designed for the purpose to aid deep learning. It has however become a key tool in deep learning research. The reason being is that the training regimes which are applied to these neural networks are computationally heavy (particularly with multiplication), and hence utilising a GPU would improve the efficiency significantly. The more the GPU's are utilized during training, the better. Higher utilization implies that there is less of a workload being applied to the CPU regarding computation which frees it up to perform other tasks.

The GPU utilization statistics of each sparse and un-sparse VGG-19 models were recorded and can be seen in Figure 22 (note that the plot has been smoothed to account for noise).

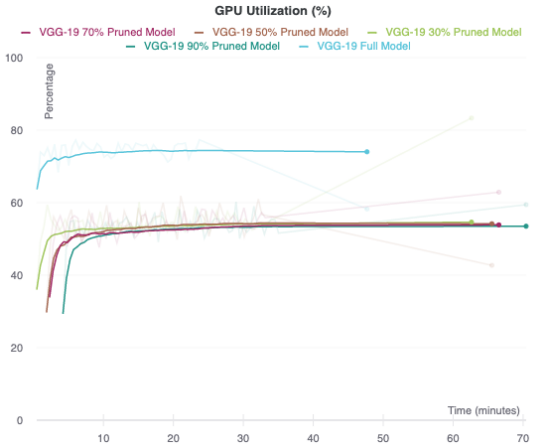

Figure 22: *GPU utilization percentage of each sparse and un-sparse VGG-19 model during the course of training.*

From Figure 22 we can see that the model which utilizes the available GPU the most is the original uncompressed model. All models which have any level of sparsity induced within them seem to utilize the GPU at the same rate. On top of this, all training regimes in which sparsity is involved have taken longer (in terms of absolute time) than the original full model.
It, therefore, seems that there are no apparent gains, relating to a more efficient training algorithm if a user decides to utilize magnitude pruning.
Improved efficiency is one of the standout reasons to apply pruning to such networks. In the context of this research, there seems to be no indication that this has occurred.

ii.3 VGG-19 pruned model results on unseen CIFAR-10 test data

After training each VGG-19 model was assessed against unseen CIFAR-10 test data. The results of which can be seen in Table 3.

As previously stated in this research paper, the difference between the metrics for each compression technique is limited due to the class balance within the dataset. The goal is to create a pipeline for users to utilize these performance metrics on any dataset (which may, in turn, be unbalanced).
From Table 3 we can see that as the level of sparsity within the network increases, the top-1 accuracy of the model decreases. However, all degradation may be considered negligible until the network has a sparsity level of 0.9 induced within it. At this point, the largest drop in top-1 accuracy can be seen (-4.81%) compared to the next largest drop seen at the 0.7 level (-0.73%). It seems as if the model can handle the loss of weights well and re-calibrate the remaining parameters during training to maintain a strong top-1 performance.
Further analysis of the most affected model (0.9 sparsity level) can be seen in Figures 23 and 24. These are the normalized confusion matrix of the model as well as the classification report depicting the performance on all classes.

Figure 23 shows that the *airplane* and *truck* classes in fact improve on the original uncompressed model in terms of recall (+0.02). This is in line with what we were seeing in the work presented by Hooker et al. [2,5,24], that there are some classes which benefit from compression being applied to the model.
On the other hand, we can also see a significant degradation in a few classes. Most notably is the *bird* class which sees a loss of -0.16 in recall as a result of magnitude pruning (at 90% level). The previously underperforming classes of *cat* and *dog* have also been affected negatively. The models relatively poor performance on these classes has been emphasised even further as a result of pruning.
Figure 24 also allows us to see the precision and f1-score associated with each class. The

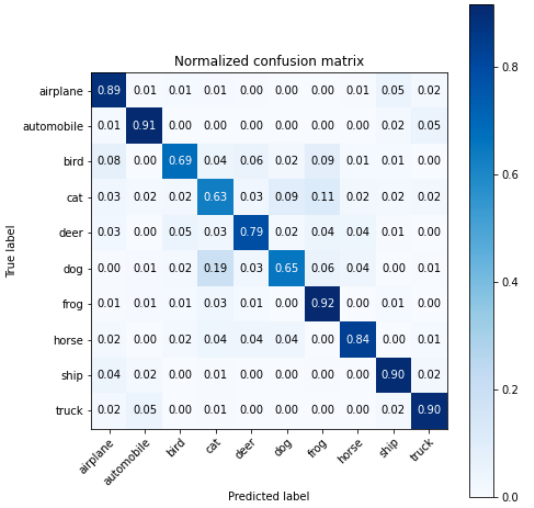

Figure 23: *Normalized confusion matrix of an VGG-19 model, with 0.9 level of sparsity induced within it, tested on unseen CIFAR-10 data.*

Classification Report

	precision	recall	f1-score
airplane	0.79	0.89	0.84
automobile	0.89	0.91	0.90
bird	0.84	0.69	0.76
cat	0.64	0.63	0.63
deer	0.82	0.79	0.80
dog	0.79	0.65	0.72
frog	0.75	0.92	0.82
horse	0.87	0.84	0.85
ship	0.87	0.90	0.88
truck	0.87	0.90	0.88
accuracy			0.81
macro avg	0.81	0.81	0.81
weighted avg	0.81	0.81	0.81

Figure 24: *Classification analysis of VGG-19 model, with 0.9 level of sparsity induced within it, tested on unseen CIFAR-10 data.*

airplane and *truck* classes which benefited from pruning (in terms of recall) incur significant losses in their associated precision metrics (losses of -0.10 and -0.09 respectively). On top of this, the most affected class in terms of recall (*bird*) improved the most in terms of precision as a result of pruning (+0.05). These offsets in precision and recall cause the overall f1-score decrease to be reduced.

Sparsity Level	Top 1	Micro Precision	Micro Recall	Micro F-1
0.00	85.87%	0.86	0.86	0.86
0.30	85.46%	0.85	0.85	0.85
0.50	85.20%	0.85	0.85	0.85
0.70	85.14%	0.85	0.85	0.85
0.90	81.06%	0.81	0.81	0.81
Quantization Technique				
float16	85.88%	0.86	0.86	0.86
dynamic range int8	85.71%	0.86	0.86	0.86

Table 3: *Compressed VGG-19 model performance metrics on CIFAR-10 unseen test data.*

ii.4 Affected classes and images

The number of affected classes and images as a result of magnitude pruning can be seen in Table 4.

Table 4 shows a general trend that increasing the sparsity level results in the number of impacted images increasing. This is not the case however for pruning the model at a 70%sparsity level. At this level, we see the lowest number of impacted images. This due to the standard deviation in the class accuracy's remaining fairly consistent with the original model (7.79 before pruning and 7.49 after inducing a sparsity level of 70%). A reason for this performance could be that inducing sparsity at this level removes a majority of the unnecessary weights which has a uniform effect across all classes rather than removing a certain subset of weights which affects certain classes. The relationship between increasing sparsity levels and the number of impacted images can be seen in Figure 25.

Looking at both figures 8 and 25 there are certain similarities between the relationships I have found in my research compared to that found in Hooker et al. [2,5,24]. In figure 8 we see that the number of impacted images seems to level off around the 50% mark before seemingly increasing again. In figure 25, past the 50% mark, the number of impacted images decreases slightly before increasing once again. The overall trend of both figures tends to agree that increasing the level of sparsity causes an increase in the overall affected images.

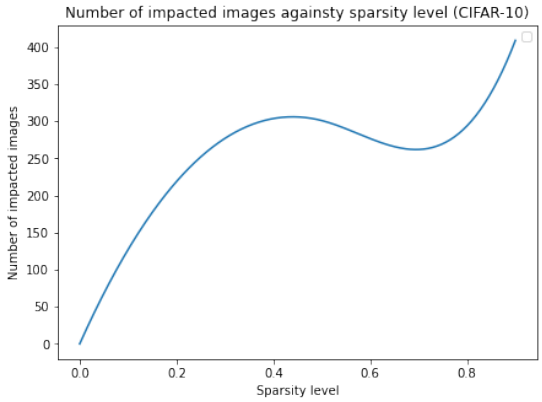

Figure 25: *The number of impacted images against increasing sparsity levels for the VGG-19 model on unseen CIFAR-10 data.*

iii. VGG-19 quantization results

For this body of research I have applied two post-training quantization techniques previously stated in this report; Dynamic range quantization and float16 quantization. Both these techniques have been applied post-training and hence the analysis regarding GPU statistics during training are not applicable (nor accuracy on training/validation data). Due to difficulties pertaining to working algorithms, I was unable to successfully apply fixed-point int8 quantization.

Sparsity Level	Number of impacted classes	Number of impacted images
0.00	-	-
0.30	2	277
0.50	2	301
0.70	3	262
0.90	3	409
Quantization Technique		
float16	0	2
dynamic range int8	0	31

Table 4: *Number of impacted classes and images found as a result of compression within the VGG-19 model (CIFAR-10).*

iii.1 VGG-19 quantized model results on unseen CIFAR-10 test data

After applying both quantization techniques two independent VGG-19 models, the resulting performance metrics can be seen in Table 3.

The float16 quantization technique does improve the models top-1 accuracy performance. It must be noted that this is quite an unusual result as we would expect a slight loss in top-1 accuracy. The improvement is extremely marginal however and hence could be taken as slight rounding issues regarding the classification algorithm. Although it is a strange result, it has been shown previously that other compression techniques such as pruning have had the same effect in some instances if the compression technique is applied during training. In the case of pruning, this is because it reduces the size of the classifier hence reducing the chance of overfitting and improving predictive accuracy. A similar inference could also be made about quantization aware training. When quantization is applied during training it reduces the precision of operations, these low-precisions of operations can then be considered as noise which does not damage the performance of the neural network but instead act as regularizers for the network [9]. Regularising the network means that the model may indeed have better generalization capabilities on unseen data.
There are significant advantages associated with utilising these post-training quantization techniques. To assess whether some of these benefits were present during my research, I analysed the compression ratios associated with each quantized model and the original full VGG-19 model. The compression ratios of both techniques are also shown in Table 5. This highlights the factors to which the models have reduced by as a result of each quantization technique. These were calculated by analysing the size of the original models against the quantized ones. They fall in line to what we expect, as the TensorFlow documentation for each technique states the benefits shown in Table 5 [10].

Further analysis of these quantized models can be seen in their respective classification reports in Figures 26 and 27.

Classification Report

	precision	recall	f1-score
airplane	0.88	0.87	0.88
automobile	0.90	0.94	0.92
bird	0.79	0.85	0.82
cat	0.73	0.71	0.72
deer	0.84	0.85	0.84
dog	0.84	0.71	0.77
frog	0.88	0.91	0.89
horse	0.86	0.90	0.88
ship	0.90	0.94	0.92
truck	0.95	0.88	0.92
accuracy			0.86
macro avg	0.86	0.86	0.86
weighted avg	0.86	0.86	0.86

Figure 26: *Classification analysis of a dynamic range int8 quantized VGG-19 model on unseen CIFAR-10 data.*

Technique	Actual Compression Ratio	TensorFlow Documented Benefits
float16	1.99	2x smaller, GPU acceleration
dynamic range int8	3.99	4x smaller, 2x-3x speedup

Table 5: *Benefits of applying dynamic range int8 quantization and float16 quantization [10].*

Classification Report

	precision	recall	f1-score
airplane	0.89	0.87	0.88
automobile	0.91	0.95	0.93
bird	0.79	0.85	0.82
cat	0.74	0.71	0.72
deer	0.85	0.84	0.84
dog	0.83	0.72	0.77
frog	0.88	0.91	0.90
horse	0.85	0.91	0.88
ship	0.91	0.94	0.93
truck	0.96	0.88	0.92
accuracy			0.86
macro avg	0.86	0.86	0.86
weighted avg	0.86	0.86	0.86

Figure 27: *Classification analysis of a float16 quantized VGG-19 model on unseen CIFAR-10 data.*

As we can see in both figures there are marginal differences in these results compared to the classification report produced by the original full model (Figure 19). The float16 quantized model does produce the same precision (0.88) and recall (0.91) values as the original full model for the *frog* class. However, their respective f1-scores differ slightly (the quantized model holds an f1-score of 0.01 larger than the original full model). This might be a slight indication that there are marginal rounding differences between the two models and hence why we were seeing the float16 model outperform the original full model in Table 3.

iii.2 Affected classes and images

From Table 4 we can see very encouraging results regarding the use of quantization as a compression technique. Both techniques exhibit no impacted classes as well as fewer impacted images compared to magnitude pruning at any level. Float16 holds the most promising results with only 2 affected images. This is indicative that simply reducing the bit representations of the weights within the network by half will cause little to no damage to the quality of the model.

XI. ResNet-50 Results

i. Uncompressed model results

The ResNet-50 model results on an unseen dataset of 10,000 images can be seen in Table 6.

Similar to the VGG-19 results, all the metrics appear to produce the same score. This is again because the CIFAR-100 dataset exhibits a class balance throughout the dataset.

The results indicate a lower performance compared to the VGG-19 model tested on the CIFAR-10 dataset. However, the CIFAR-100 dataset has far fewer training examples per class compared to the CIFAR-10 dataset. This means that the ResNet-50 model will have fewer images to distinguish the key features of certain classes. This, therefore, makes it harder for the model once it is deployed on an unseen dataset. Overall though, the ResNet-50 model still produces strong classification results.
Given the number of classes present, the visualization techniques (confusion matrices and classification reports) used in analysing the CIFAR-10 dataset were not possible in this case.

ii. ResNet-50 pruning results

Similar to the VGG-19 experiment setup, magnitude pruning was applied to the ResNet-50 model across four levels of sparsity [0.3, 0.5,

	Top 1	Micro Precision	Micro Recall	Micro F-1
ResNet-50 uncompressed	77.31%	0.77	0.77	0.77

Table 6: *ResNet-50 uncompressed model performance metrics on CIFAR-100 unseen test data.*

0.7, 0.9]. These were applied on independent models and the same metrics were recorded.

ii.1 Training

The performance of the five independent ResNet-50 models during training can be seen in Figure 28 and 29.

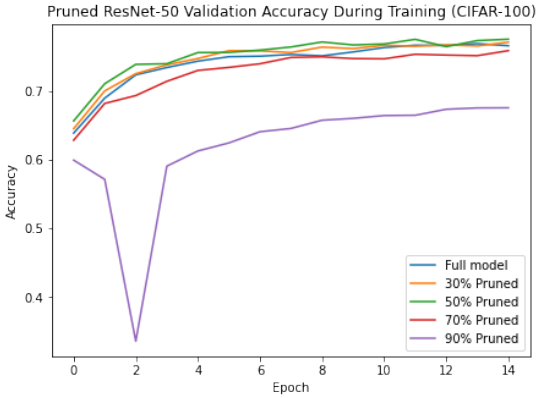

Figure 28: *The validation accuracy of five independent ResNet-50 models during the course of training. Each model has a different level of sparsity induced within the network.*

From figures 28 and 29 we can see a clear dip in performance for the 90% pruned model on the second epoch. This is most likely due to noise within the network during training. It is also clear to see that the 90% pruned model is also the most under-performing. This is slightly different to what we were seeing in Figure 21 where the model pruned to the 0.9 level ended on the lowest CEL. This is since the training regime consisted of far fewer epochs and hence the effects of overfitting were less.
It can also bee seen that both the 30% and 50% pruned models outperform the original full model. This is again because the ResNet-50 model is highly overparameterized and hence pruning at these levels acts as a regularizer

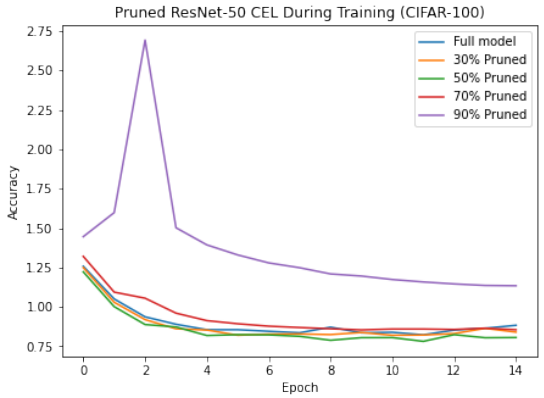

Figure 29: *The CEL of five independent ResNet-50 models during the course of training. Each model has a different level of sparsity induced within the network.*

for the model which improves generalisation performance.
The corresponding final validation accuracy's after training for each of the ResNet-50 models can be seen in Table 7.

Table 7 confirms what we were seeing in Figures 28 and 29. Pruning up to the 50% level seems to improve the model due to reducing the chance of overfitting during training. Beyond this point, however, we can see a reduction in model quality.

ii.2 GPU Utilization

Similar to the VGG-19 experiments, the GPU utilization of each independent ResNet-50 model was recorded during training. This was done to see if the same traits of inefficiency would be seen in the ResNet-50 model.

From Figure 30 we can see similar results to what we were seeing in Figure 22. The model which seems to be the most efficient during training is the original full model whilst the pruned models all seem to hold very similar

Sparsity Level	0%	30%	50%	70%	90%
Top 1 Validation Accuracy	76.58%	77.10%	77.53%	75.87%	67.55%

Table 7: *ResNet-50 final validation accuracy after training. Each independent ResNet-50 model had a varying level of sparsity induced within the model. All models validation accuracy was assessed on CIFAR-100 validation data.*

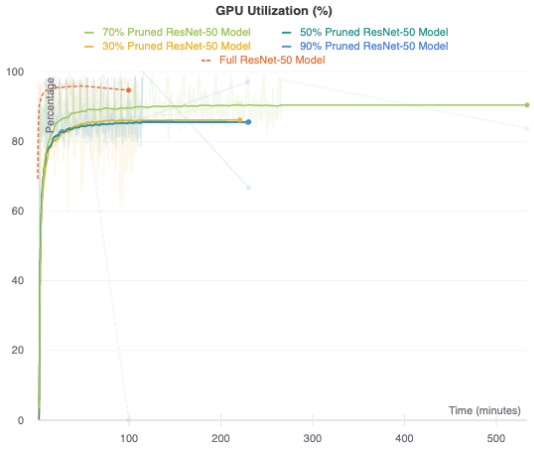

Figure 30: *GPU utilization percentage of each sparse and un-sparse ResNet-50 model during the course of training.*

utilization metrics. This is again a discouraging sign for users hoping to use magnitude pruning to take advantage of the documented efficiency benefits. It can also be seen in Figure 30 that the actual time take for the training process to complete for all of the pruned models is longer than the original full model. This is particularly true for the 0.7 pruned model which took significantly longer than all of the other models. It is still not clear as to why this was the case.

ii.3 ResNet-50 pruned model results on unseen CIFAR-100 test data

All independent ResNet-50 models were tested against unseen CIFAR-100 test data. Their corresponding performance metrics can be seen in Table 8. Again the metrics all produce very similar results at all sparsity levels due to the class balance exhibited within the dataset.

Table 8 shows that the ResNet-50 model pruned at the 30% sparsity level does in fact outperform the original full model which highlights the previous findings regarding improved generalisation performance. All other pruned ResNet-50 models performed worse than the original but the damage incurred in their performance is minimal (excluding the 0.9 sparsity model). At the 90% sparsity level, we see the largest loss to model performance with a change of -9.38% top 1 model accuracy. However, given how severe this sparsity level is, it is not as bad as one might expect. These results fall in line with the experiment findings regarding the CIFAR-10 dataset.

ii.4 Affected classes and images

The number of impacted classes and images as a result of magnitude pruning on the ResNet-50 model can be seen in Table 9.

Table 9 holds fairly similar trends to what we were seeing in Table 4. The number of impacted images again increases as the level of sparsity increases but seems to drop at the 70% level. Interestingly the model which had a sparsity level of 50% seemed to have the fewest impacted classes and yet the second most amount of impacted images. This is suggestive that the impacted classes were more severely affected compared to the other pruned models.
The ResNet-50 model with a sparsity level of 30% also encountered a high number of impacted images (917) and classes (26) even though it improved on the original full model (in terms of top-1 accuracy). This is likely due to a certain subset of images and classes which were impacted positively. This again falls in line with the work that Hooker et al. [2,5,24] had put forward in which they showed a certain subset of classes/images improving as a

Sparsity Level	Top 1	Micro Precision	Micro Recall	Micro F-1
0.00	77.31%	0.77	0.77	0.77
0.30	77.82%	0.78	0.78	0.78
0.50	76.40%	0.76	0.76	0.76
0.70	76.09%	0.76	0.76	0.76
0.90	67.93%	0.68	0.68	0.68

Table 8: *Pruned ResNet-50 model performance metrics on CIFAR-100 unseen test data.*

Sparsity Level	Number of impacted classes	Number of impacted images
0.00	-	-
0.30	26	917
0.50	17	1051
0.70	25	1016
0.90	26	1286

Table 9: *Number of impacted classes and images found as a result of magnitude pruning within the ResNet-50 model (CIFAR-100).*

result of the compression technique.
The trend of increasing sparsity level against impacted images can be seen in Figure 31.

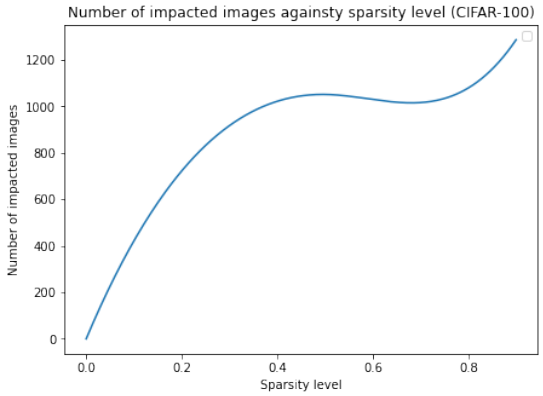

Figure 31: *The number of impacted images against increasing sparsity levels for the ResNet-50 model on unseen CIFAR-100 data.*

Figure 30 again shows a fairly similar trend to what we were seeing in Figure 25. This is indicative that increasing the level of pruning can cause the number of impacted images to also increase.

XII. Areas for further research

This body of research has been focused on highlighting the apparent impact that compression techniques have on CNNs. It has been insightful to see the comparisons between quantization and pruning as well as analysing the efficiency performance of pruning. However, more work needs to be done to reinforce these findings. This includes extending this area of research to new architectures such as recurrent neural networks (RNNs).
RNNs are widely used networks that hold strong performances on tasks such as question answering, predictive text and sentiment analysis. These networks are again hugely demanding in terms of memory and efficiency making compression highly applicable to them. Due to their popularity, it is important to ensure the same type of experiments are applied to them to mitigate against adverse outcomes when these networks are deployed into the wild.
Further research also needs to be done to understand what specific underlying reasons are associated with compression harming

these classes and images. There also needs to be an effective framework put in place to manage these risks and uncertainties should a user decide to utilise compression. An interesting area of mathematics that could be used to possibly address these points is PAC Bayesian Theory.

The concept of Bayesian Machine learning allows for the management and evaluation of randomness and uncertainty within machine learning tasks. This is the goal associated with understanding CIEs, to understand the uncertainty they cause with classification/NLP and to manage this uncertainty.

PAC stands for Probably Approximately Correct. PAC learning refers to the idea that a learner will select a generalization function within a certain class of possible functions. The idea behind this choice is that there is a high probability (the "probably" part) that the chosen function will have a low generalization error (the "approximately correct" part) [33]. When this idea is applied to Bayesian learning algorithms it is referred to as PAC-Bayesian [34]. It has been shown that PAC-Bayesian applications are a *"relevant toolbox to derive theoretical guarantees"* [34] on areas such as DNNs. It is guarantees surrounding classification/NLP tasks applied on CIE that compression research should aim to achieve.

Letarte et al. [35] looked at developing an end-to-end training regime for a binary DNN but also produced insightful PAC-Bayesian generalization bounds for binary activated DNNs. Letarte et al. [35] were able to design an algorithm which enabled them to learn the parameters surrounding the predictor in question. This allowed them to gain a deeper understanding of the generalization abilities of the binary activated multi-layer networks. It would be very interesting to see if similar results could be achieved with compressed neural networks.

XIII. Evaluation, appraisal and reflection

I was extremely ambitious with my original outline of how I envisaged the progress of my dissertation. I was hoping to have applied quantization and pruning techniques to RNNs, looked at PAC Bayesian theory as a way of controlling uncertainty and also quantified fairness within the context of PIEs. Looking back on these goals I feel I was overambitious and did not fully assess the technical demands of the task at hand.

A major stumbling block for me has been establishing working algorithms which can quantize CNNs and record their performance on both a model and class level. Originally my goal was to recreate the results seen in the work by Hooker et al. [2,5,24] but with quantization as the method of compression. What became apparent soon after undertaking the task, was the scale of their experiment set up and the difficulty in recreating it with the hardware and storage access I had. As a result, I needed to rethink how to approach the problem with limited resources and still yield meaningful results.

Even though I still produced some comparisons regarding pruning and quantization I was disappointed not to have further working algorithms which could have proved to hold more insight into the area.

I decided to make use of transfer learning techniques to overcome the issues associated with computation and hardware costs. I also decided to focus my quantization research on tasks Hooker et al. [2,5,24] did not look into (such as GPU analysis during training). On top of this, I also felt that I conducted extensive research on not only the theories behind these compression techniques but also methods on implementing them. I believe my understanding regarding the field of compression is very strong and my interest to continue research within this field is ever-present.

The lessons I have learned throughout this process relate to being realistic with my expectations. I believe this is a very exciting field and

one with a lot of open questions, but I must be honest with myself and not expect to answer them all.

XIV. Ethics

All research undertaken within this research paper has conformed to the University's research code of practise and I have also familiarised myself with the University's policies and procedures regarding research governance and ethics.
I have not had to submit an ethical review for my work as all my experiments have been based on a publicly available datasets (CIFAR-10 and CIFAR-100). All material, not mine has been referenced and conforms to the original author's copyright requirements.

XV. Conclusion

The importance of reducing the size and efficiency costs of DNNs is evident throughout this corpus. There have multiple research papers depicting state-of-the-art techniques for applying quantization and pruning across a range of neural networks. All of these papers analysed the performance of their respective techniques by looking at how the overall model performed. Hooker et al. [2,5,24] highlighted the need for more granular analysis of these compression induction techniques. Through their findings, they were able to show the negative implications that pruning and quantization had on certain classes.
Through undertaking my research I have found similar findings to what was put forward by Hooker et al. [2,5,24] - Compression techniques do harm certain classes and images within classification tasks. Furthermore, more extreme compression regimes can have a more severe effect in this regard.
I also believe my research has highlighted the benefits of using quantization techniques over magnitude pruning. This can be seen with the results seen in section X. where both quantization regimes showed far fewer signs of inflicting underlying damage than magnitude pruning.
My results also suggested that magnitude pruning does not exhibit the efficiency benefits associated with inducing sparsity within neural networks. This is due to the inefficiency's seen with pruned models during my experiments provided by the GPU analyses seen in sections Xii.2. and XIii.2. These results are very informative to show users that certain popularized pruning regimes may not be beneficial for their models.
More research does need to be done however to reinforce and extend this area of deep learning. This is particularly the case with analysing these compression techniques on NLP tasks, mitigating against the risks associated with these compression techniques and gaining a deeper understanding as to why compression can have such an effect on classes and images. Compression is an area of deep learning that is extremely beneficial for the realistic deployment of extremely large architectures on resource-constrained environments. However, care needs to be taken when utilising these techniques. This is especially the case in areas such as medicine where the margin for error is very small and the consequences for such errors are large. It is important to follow on from this research to ensure users are reaping the benefits associated with these techniques whilst practising them in an informed manner.

References

[1] Liu, Zhuang, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. "Rethinking the Value of Network Pruning". *CoRR*, ArXiv:1810.05270 [Cs, Stat], 5 March 2019. *Available at:* `http://arxiv.org/abs/1810.05270`.

[2] Hooker, Sara, Aaron Courville, Yann Dauphin, and Andrea Frome. "Selective Brain Damage: Measuring the Disparate Impact of Model Pruning". *CoRR*, ArXiv:1911.05248 [Cs, Stat], 12 November 2019. *Available at:* `http://arxiv.org/abs/1911.05248`.

[3] Zhu, Michael, and Suyog Gupta. "To prune, or not to prune: Exploring the efficacy of pruning for model compression". *CoRR*, ArXiv:1710.01878. 5 October 2017. *Available at:* https://arxiv.org/abs/1710.01878v2.

[4] Gale, Trevor, Erich Elsen, and Sara Hooker. "The state of sparsity in deep neural networks". *CoRR*, ArXiv:1902.09574 [Cs, Stat], 25 February 2019. *Available at:* http://arxiv.org/abs/1902.09574.

[5] Hooker, Sara, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. "What Do Compressed Deep Neural Networks Forget?" *CoRR*, ArXiv:1911.05248 [Cs, Stat], 13 July 2020. *Available at:* http://arxiv.org/abs/1911.05248.

[6] "Structured sparsity regularization" 16 March 2020. *Online Article*. *Wikipedia The Free Encyclopedia*. *Available at:* https://en.wikipedia.org/wiki/Structured_sparsity_regularization

[7] Yau, Soon. (2018). *Online Article*. "Speeding up Deep Learning with Quantization" *Towards Data Science*. *Available at:* https://towardsdatascience.com/speeding-up-deep-learning-with-quantization-3fe3538cbb9

[8] "Quantization". *Online Article*. *Neural Network Distiller*. *Available at:* https://nervanasystems.github.io/distiller/quantization.html

[9] Hubara, Itay, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. "Quantized neural networks: Training neural networks with low precision weights and activations". *CoRR*, ArXiv:1609.07061 [Cs], 22 September 2016. *Available at:* http://arxiv.org/abs/1609.07061.

[10] "Post-training quantization" 2020. *Online Article*. *TensorFlow*. *Available at:* https://www.tensorflow.org/lite/performance/post_training_quantization

[11] Brownlee, Jason. December 20, 2017. "A Gentle Introduction to Transfer Learning for Deep Learning". *Online article*. *Machine Learning Mastery*. *Available at:* https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[12] Kaushik, Aakash. "Understanding the VGG19 Architecture". *Online article*. *OpenGenus IQ: Learn Computer Science*. *Available at:* https://iq.opengenus.org/vgg19-architecture/

[13] Yang, K., Zheng, Y. (2018). "Illustration of the network architecture of VGG-19 model: conv means convolution, FC means fully connected". *Online image*. *ResearchGate*. *Available at:* https://www.researchgate.net/figure/llustration-of-the-network-architecture-of-VGG-19-model-conv-means-convolution-FC-means_fig2_325137356

[14] Krizhevsky, Alex. 2009. "The CIFAR-10 dataset". *Online article*. *Computer Science Department: Toronto University*. *Available at:* https://www.cs.toronto.edu/~kriz/cifar.html

[15] Huang, Xudong. 2018. "Stochastic gradient descent compared with gradient descent". *Online image*. *ResearchGate*. *Available at:* https://www.researchgate.net/figure/Stochastic-gradient-descent-compared-with-gradient-descent_fig3_328106221

[16] "Classification: Precision and Recall" 2020. *Online course*. *Machine Learning Crash Course: Google*. *Available at:* https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

[17] "F1 score" 8 July 2020. *Online article*. *Wikipedia*. *Available at:* https://en.wikipedia.org/w/

index.php?title=F1_score&oldid=966641660

[18] Shung, Koo Ping. 10 April 2020. "Accuracy, Precision, Recall or F1?". *Online article. Towards Data Science. Available at:* https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[19] "Convolutional neural network" 24 August 2020. *Online article. Wikipedia The Free Encyclopedia. Available at:* https://en.wikipedia.org/wiki/Convolutional_neural_network.

[20] Molchanov, Dmitry, Arsenii Ashukha, and Dmitry Vetrov. "Variational Dropout Sparsifies Deep Neural Networks". In Proceedings of the 34th International Conference on Machine Learning - Volume 70, 2498–2507. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017. *Available at:* https://dl.acm.org/doi/10.5555/3305890.3305939.

[21] Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through L_0 Regularization". Published as a conference paper at the International Conference on Learning Representations (ICLR) 2018. *CoRR*, ArXiv:1712.01312 [Cs, Stat], 22 June 2018. *Available at:* http://arxiv.org/abs/1712.01312.

[22] Choukroun, Yoni, Eli Kravchik, Fan Yang, and Pavel Kisilev. "Low-bit Quantization of Neural Networks for Efficient Inference". *CoRR*, ArXiv:1902.06822 [Cs, Stat], 25 March 2019. *Available at:* http://arxiv.org/abs/1902.06822.

[23] Banner, Ron, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In Advances in Neural Information Processing Systems 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, 7950–7958. Curran Associates, Inc., 2019. 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada. *Available at:* https://papers.nips.cc/paper/9008-post-training-4-bit-quantization-of-convolutional-networks-for-rapid-deployment.pdf.

[24] Hooker, Sara, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. "Characterizing and Mitigating Bias in Compact Models". Proceedings of the 37 th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. *Available at:* http://whi2020.online/poster_73.html?fbclid=IwAR29N1cUi8AncdgbBP-wfKGgmox5_-qYwnOSoGosiaQ8c7SKpCp-7ehkC8w.

[25] "Trim Insignificant Weights". *Online article. TensorFlow. Available at:* https://www.tensorflow.org/model_optimization/guide/pruning.

[26] Dwivedi, Priya. 27 March 2019. "Understanding and Coding a ResNet in Keras". *Online article. Medium. Available at:* https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33.

[27] Adaloglou, Nikolas. 23 March 2020. "Intuitive Explanation of Skip Connections in Deep Learning". *Online article. AI Summer. Available at:* https://theaisummer.com/skip-connections/.

[28] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". *CoRR*, ArXiv:1512.03385 [Cs], 10 December 2015. *Available at:* http://arxiv.org/abs/1512.03385.

[29] Brownlee, Jason. 24 January 2019. "Understand the Impact of Learning Rate on Neural Network Performance". *Online article. Machine Learning Mastery (blog). Available at:* https://machinelearningmastery.com/

understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

[30] Bushaev, Vitaly. 24 October 2018. "Understand the Impact of Learning Rate on Neural Network Performance". *Online article*. *Medium*. *Available at:* https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.

[31] "Graphics Processing Unit". *Online article*. *Wikipedia*. *Available at:* https://en.wikipedia.org/wiki/Graphics_processing_unit.

[32] BDsouza, Jason. 25 April 2020. "What Is a GPU and Do You Need One in Deep Learning?". *Online article*. *Medium*. *Available at:* https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d.

[33] "Probably Approximately Correct Learning". 21 August 2020. *Online article*. *Wikipedia*. *Available at:* https://en.wikipedia.org/w/index.php?title=Probably_approximately_correct_learning&oldid=974229195.

[34] Guedj, Benjamin. 7 May 2019.. 'A Primer on PAC-Bayesian Learning' (2019). *CoRR*, ArXiv:1901.05353 [Cs, Stat]. *Available at:* http://arxiv.org/abs/1901.05353.

[35] Letarte, Gaël, Pascal Germain, Benjamin Guedj, and François Laviolette. 4 February 2020. "Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks". NeurIPS 2019, Dec 2019, Vancouver, Canada. ffhal-02139432v2f *CoRR*, ArXiv:1905.10259 [Cs, Stat]. *Available at:* http://arxiv.org/abs/1905.10259.

[36] Oprea, Mihaela, and Lazaros Iliadis. 4 February 2020. "An Artificial Intelligence-Based Environment Quality Analysis System". In Engineering Applications of Neural Networks, edited by Lazaros Iliadis and Chrisina Jayne, 499–508. IFIP Advances in Information and Communication Technology. Berlin, Heidelberg: Springer, 2011. *CoRR*, ArXiv:1905.10259 [Cs, Stat]. *Available at:* https://doi.org/10.1007/978-3-642-23957-1_55.