Analysis of pruning methods for compact neural networks on embedded devices

SEBASTIAAN HOFSTEE, University of Twente, The Netherlands

Pruning of neural networks is a technique often used to reduce the size of a machine learning model, as well as to reduce the computation cost for model inference. This research provides an analysis on four current pruning techniques that theoretically efficiently reduce the machine learning model size, where efficiency is defined by the relation between the compression of the model and the accuracy of the model. Furthermore, this research will assess in what way these four neural network pruning techniques affect the total energy consumption during model inference on a Raspberry Pi 4B board, applied to MobileNetV2, a machine learning model architecture optimized for image classification on embedded devices. Lastly, the research will analyze the trade-offs between energy consumption, model size and model accuracy for each of the assessed pruning algorithms applied to one of the most commonly used neural network architectures, MobileNetV2, on a Raspberry Pi 4B prototyping board. The research is expected to provide engineers a reference providing guidance upon deciding what pruning technique to use for a machine learning model to be deployed on an embedded device.

Additional Key Words and Phrases: Pruning, Neural Networks, Machine Learning, Embedded Devices, Energy Consumption, Efficiency

1 INTRODUCTION

1.1 Context and relevance

Machine learning models have grown increasingly large over the past years [14], while at the same time increasingly often machine learning models are used on embedded devices [22]. As embedded devices often have very limited resources, pruning algorithms can be used to effectively reduce the size of a model, as well as lower the model inference time [2].

There are various novel pruning techniques proposed [30][8][24] which allow for significant reduction of model size while keeping the accuracy of a model acceptable.

Blalock et al. [2] show that existing work analyzing individual pruning techniques, often has shortcomings, among other things regarding the identification of experiment setups and metrics, the usage of too few combinations of datasets and architectures as well as failure to control confounding variables. As the scope of this research is somewhat limited due to time constraints, it is not possible to analyze a vast set of (dataset, architecture) combinations. Regardless of this, the work of Blalock et al. still provides valuable insights regarding the analysis of pruning methods.

While research has been done evaluating the energy consumption of classification algorithms [26], as well as on minimizing the energy consumption of embedded neural networks by introducing quantized neural networks (QNNs) [21], there appears to be only little research done on the impact of algorithms on the energy consumption of embedded devices. The work that has been done however, is laid out in section 3 of this paper.

Furthermore, García-Martín et al. [9] have shown in their analysis on the estimation of energy consumption in machine learning that merely the number of weights in a machine learning model is too simplistic and cannot be seen as a good estimator for the energy consumption of a machine learning model. Something which was also subscribed in research by Yang et al. [29], in which is stated that not only computation, but also memory access affects the total energy consumption of a neural network, where it is important to note that fetching data from memory takes multiple order of magnitudes more energy than the energy required for the computation itself, as shown by Horowitz in 2014 [12].

Based on this, it appears reasonable to assume that the energy consumption of a machine learning model does not only depend on the model size size, and that further research into the real-world energy consumption of different pruning algorithms on embedded devices, among which this work covers, is relevant.

1.2 Objective

The objective of this research is to provide engineers a reference providing insight and guidance upon deciding what pruning technique could for a machine learning model to be deployed on an embedded device, depending on the accuracy, energy consumption and model size requirements. The analysis of the results should contain information about the trade-offs regarding energy consumption, model size and accuracy that the four different pruning techniques entail and try to generalize, where appropriate, the results to be able to make a possible prediction about other, not touched upon pruning techniques based on for example whether the pruning technique is structured or unstructured.

1.3 Research question

The aforementioned objective can be reached by answering the following research question, which is the main question to be answered for this research:

- **RQ**: What are the trade offs regarding accuracy, energy consumption and model size between unstructured global magnitude pruning, network slimming, L1 norm structured pruning and random pruning on MobileNetV2?

To be able to give a thorough answer to the main research question however, it is necessary to first answer the following subquestions:

- **Sub-RQ1**: What neural network pruning techniques currently exist that efficiently reduce the model size? (efficiency is in this case defined by the relation between the compression of the model and the accuracy of the model)
- **Sub-RQ2**: How do current neural network pruning techniques affect the total energy consumption during inference when applied to MobileNetV2 on a Raspberry Pi 4B, independent of model size?

TScIT 37, July 8, 2022, Enschede, The Netherlands

2 BACKGROUND

2.1 Preliminary knowledge

The idea of pruning neural networks, in the sense that one removes parameters that are deemed unimportant, has been around for quite some time since it was first proposed by Lecun et al in 1989 [16]. To obtain a better idea on how pruning exactly affect the neural network, an example is given in Figure 1, which can originally be found in [10].

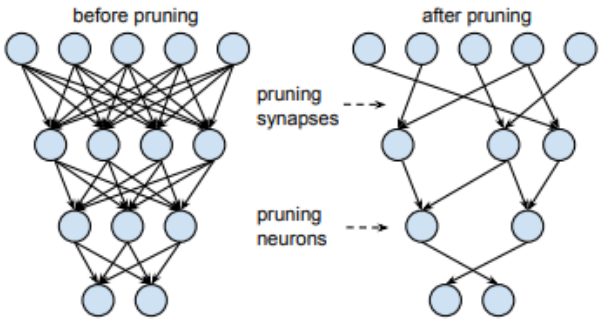

Fig. 1. Neural network before and after pruning [10]

As the pruning of parameters in a machine learning model generally results in the accuracy of a machine learning model decreasing, pipelines have been devised to minimize this accuracy loss. In particular, model fine tuning is often applied as to regain a certain degree of accuracy after pruning a model. This fine tuning entails retraining the model after pruning the network, as proposed by Han et al. in 2015 [10]. The retraining phase might be implemented in several ways, among which the approach used in this work, which was originally proposed by Frankle et al. in their 2019 work [7]. This approach entails rewinding the weight values to those before the pruning took place for those parameters left unpruned, and retrain the model starting from those values rather than resetting them to the values they had at the first iteration. A concise overview of a typical pruning pipeline, which has also been used for this work, can be seen in Figure 2, which was shown in a paper by Chen et al. in 2022 [4].

Fig. 2. Typical neural network pruning pipeline [4]

Furthermore, there is two main categories of pruning techniques: unstructured pruning, which relies on the removal of individual connections between neurons, and structured pruning, which relies on the pruning of entire convolution channels, filters or layers. In Figure 3, originally presented in [5], this difference between these two fundamentally different pruning techniques is made clear.

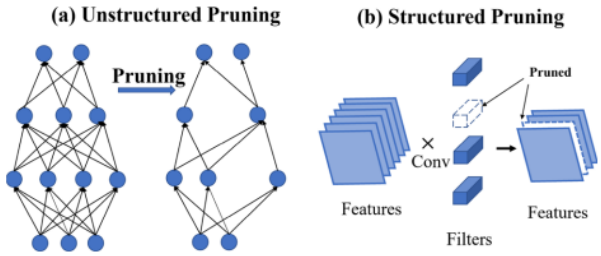

Fig. 3. **a** For unstructured pruning, individual connections between neurons are removed. **b** The structured pruning example shows an entire convolution filter being pruned [4]

As to obtain somewhat of an insight on the difference between these two categories of pruning methods, this work analyses two structured pruning techniques as well as two unstructured pruning techniques.

Regarding the unstructured pruning techniques, the first pruning technique to be analyzed is unstructured global magnitude pruning, introduced by Han et al. in 2015 [10]. This pruning method relies on pruning weights in a neural network based on their magnitude, where the lowest magnitude weights are set to zero. The second unstructured pruning technique that is analyzed in this work is unstructured random pruning. This pruning method serves as a baseline to compare other pruning techniques to and provides an hence excellent sanity check when analyzing pruning techniques (does a technique perform better than random pruning).

Regarding the structured pruning techniques, the first pruning technique to be analyzed is structured L1-norm filter pruning, introduced by Li et al. in 2016 [17], which relies on pruning CNN filters that are identified as having only a very limited effect on the accuracy of the model. In their work, Li et al. mention that magnitude based pruning of weights might not reduce the inference cost, and hence the energy consumption, to a large enough extent despite the significant pruning of weights due to the irregular nature of its sparsity. The second structured pruning technique to be analyzed is network slimming, intoduced by Liu et al. in 2017 [18]. This pruning method is based on automatically indentifying insignificant channels and immediately pruning these during training. The significance of channels is determined by analyzing the scaling factors in batch normalization, and removing those channels with scaling factors near zero. An example of this can be seen in Figure 4, which is a graphic from the original paper by Liu et al. [18].

The choice of pruning techniques is not arbitrary, as disregarding the random unstructured pruning baseline, the selected pruning methods are among those referenced the most. With Han et al. their global unstructured magnitude pruning [10] being referenced almost five thousand times, L1-norm filter pruning introduced by Li et al. [17] being referenced more than two and a half thousand times and network slimming introduced by Liu et al. [18] being referenced over one and a half thousand times.

As these pruning methods appear to be of such popularity, it appears likely that these methods are among the most relevant methods to analyze in this work.

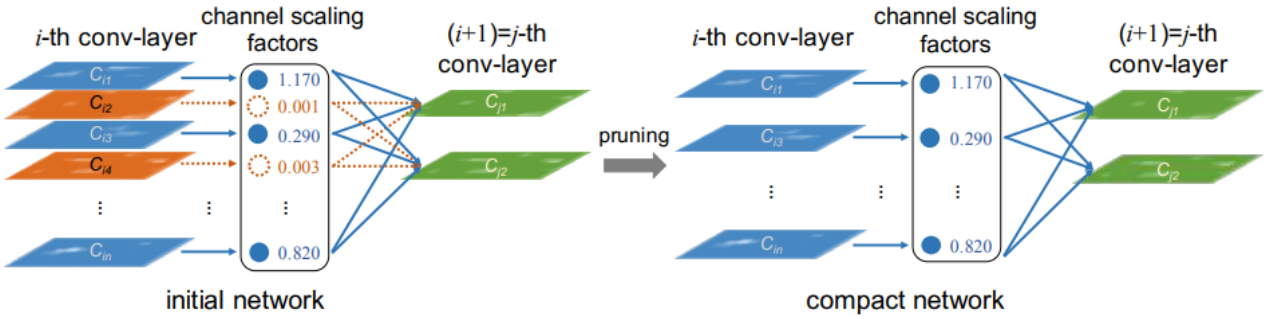

Fig. 4. The concept of network slimming. Channels with near-zero scaling factors are pruned. [18]

A number of neural network architectures optimized for edge devices have been identified in recent work by Chen et al. in 2020 [5]. These compact architectures make use of specific design strategies, such as but not limited to reducing the size of convolutional filters, introducing shortcut connections to building blocks (as is the case for MobileNetV2 [23]) and channel shuffling.

An overview of a selection of compact neural network architectures as presented in [4] can be seen in Figure 5

The reason why for this work, MobileNetV2 was selected to apply the analyzed pruning methods on is because the goal of this work is to provide an analysis of the pruning methods that is as relevant as possible. As MobileNetV2 has by far, the most citations out of all architectures presented in Figure 5, with over ten thousand citations to the original paper [23], and hence appears to be the most relevant architecture to apply the pruning methods to.

3 RELATED WORK

Neural network pruning has been around since the late 1980s [13][15] and much research has since been done in the field.

Blalock et al. [2] provides a very useful view into the state of neural network pruning and its references provide an excellent collection of novel pruning techniques [11][30][8][24]. The paper furthermore looks into how one can systematically compare different pruning methods by laying out a list of best practices, and shows that many novel pruning techniques have not been compared according to these best practices.

A novel paradigm in neural network pruning is pruning at initialization (PaI), which is well-covered by Wang et al.[28] in a recent paper. This new paradigm does however currently underperform traditional pruning methods in practical performance [27], and hence is not a suitable paradigm to consider as part of this research.

Research on the energy consumption of pruned neural networks has mostly been done in a theoretical way, and seems to mainly focus on estimating the overall energy consumption of a model as a whole, as has been done in work by Cai et al. in 2017 [3], which makes use of a regression based approach. An excellent overview of the state of the art of such theoretical energy consumption models can be found in the work of Garciá-Mártin et al. in 2019 [9].

One of the few researches that focus on the energy consumption of pruning algorithms however, is a very interesting paper by Yang et al. [29], which proposes a novel pruning algorithm for Convolutional Neural Networks, Energy-aware Pruning, and approximates the energy consumption by means of a model which takes into account the computation and memory accesses, and uses values for energy which are extrapolated from hardware measurements in the real-world, making the approximation for the energy usage more accurate. In their research, their novel pruning algorithm is compared to magnitude-based pruning [10] and using no pruning algorithm at all. While it is still mostly theoretical and only compares the novel pruning algorithm to one other, more common pruning algorithm, this research is of great value for the research to be conducted and provides a useful reference regarding the analysis of obtained data for the energy consumption of different pruning algorithms.

Moreover, work from Mirmahaleh and Rahmani in 2019 [20] proposes a novel pruning method for Deep Neural Networks which relies on pruning weights, layers and neurons based on the minimum distance error, and shows to speed up inference by approximately 22.56% – 77% and a reduction in energy consumption by 65.94% - 88.54% as compared to not utilizing the novel pruning algorithm based on simulations. Even though Mirmahaleh and Rahmani their work does not provide a comparison with other pruning algorithms, it does show interest in the topic, which is of relevance to this work.

As the differences in the energy consumption and inference time of different pruning algorithms on embedded devices seem to not have been explored much, and as no thorough real-world analysis has been composed that can be used by engineers as a reference providing guidance upon deciding what pruning technique to use for a machine learning model to be deployed on an embedded device, there appears to be ample scientific value to presented work.

4 METHODOLOGY

In this section, the exact approach to answer the research question posed will be laid out.

4.1 Implementing pruning algorithms

The pruning algorithms that have been analyzed in this work, global unstructured random weight pruning, global unstructured magnitude pruning [10], L1-norm filter pruning [17] and network slimming [18], as well as the MobileNetV2 network have all been implemented in PyTorch. This is due to the fact that for two structured

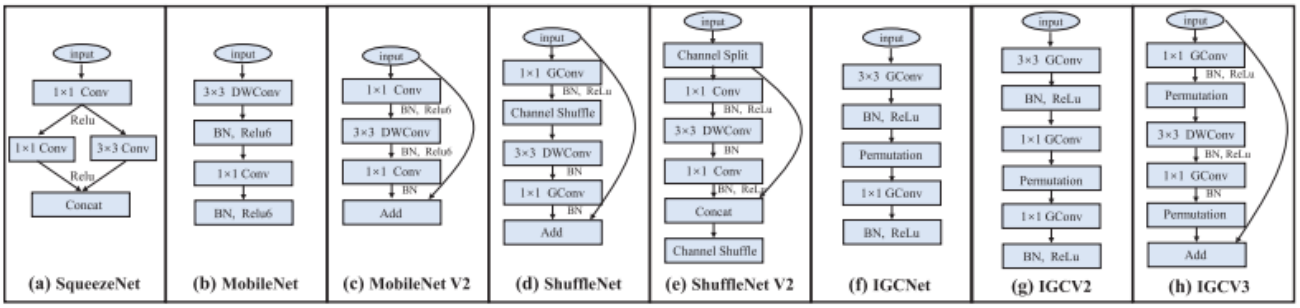

Fig. 5. An overview of compact neural network architectures as presented in [4]. **a** SqueezeNet, **b** MobileNet, **c** MobileNetV2, **d** ShuffleNet, **e** ShuffleNetV2, **f** IGCNet, **g** IGCV2, **h** IGCV3

pruning methods, existing PyTorch implementations exist which are based directly on their respective original papers. These implementations have been adapted for the purpose of this paper, and can be accessed through a special GitHub repository.

4.2 Training and pruning MobileNetV2

MobileNetV2 has been trained on the CIFAR-10 dataset, in PyTorch, through the Jupyter notebook hosted by the University of Twente. It was chosen to use this Jupyter notebook as it gives access to high-performance GPU equipment that is able to quickly and efficiently train, convert and prune models CIFAR-10 was chosen as a dataset to train MobileNetV2 on for this work, as it appears to be widely used in literature, as shown in [2], as a dataset used to train neural network architectures on for the purpose of assessing the (theoretical) performance of pruning networks. Furthermore, CIFAR-10 is a convenient as it is of a relatively manageable size (163MB) and hence does not require extremely high-performance equipment to utilize it for training neural networks, which is among other things beneficial for the reproducability of this work. The unpruned MobileNetV2 network trained on CIFAR10 acts as the baseline of the experiment, to which all pruning algorithms have been applied. After applying a pruning technique, fine-tuning is used to improve the overall accuracy of the pruned model. This fine-tuning entails retraining unpruned parameters from their final trained values.

4.3 Determining model accuracy

To be able to determine the accuracy of the (pruned) models, the model is evaluated by using PyTorch model inference on the CIFAR-10 test set, and then use the standard formula for accuracy: $Accuracy = \frac{TruePositive+TrueNegative}{TruePositive+TrueNegative+FalsePositive+FalseNegative}$

4.4 Exporting the model

The model is exported by first converting the model to an Open Neural Network Exchange (ONNX) [6] format. The ONNX platform allows one to interchange machine learning models between various frameworks. In the case of this work, the ONNX platform is used to export the original PyTorch model into a TensorFlow Lite [1] model. TensorFlow Lite is a frame work that is often used to deploy machine learning models on edge devices, and is the format to which the PyTorch models are converted before being loaded on the testboard in this work.

4.5 Energy consumption measurements

For measuring the energy consumed, the following setup will be used: An arduino Uno, which is connected to an INA219 current sensor [25], as well as the prototpying board, a Raspberry Pi 4B, of which the VCC line from the power supply will be routed through the INA219 chip by means of connecting it to the V_{in} and V_{out} of the INA219 chip, hence connecting the INA219 chip in series with the prototpying board. The Arduino Uno can then report the measured values for the current and the voltage, together with a timestamp to the computer that is connected by means of a Serial connection. A schematic of the circuit can be seen in Figure 6.

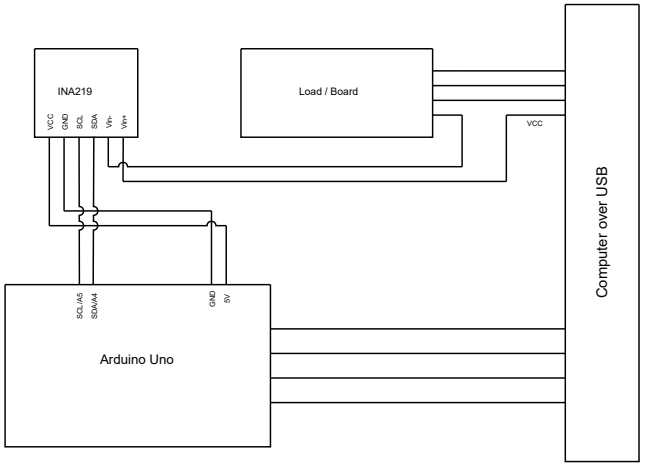

Fig. 6. Circuit of current measurement setup

The values for the voltage (which stays approximately constant) and current, will provide one with a power consumption in mW for each timestamp at which a measurement has been taken.

The INA219 chip has, according to its specification, a current and bus voltage accuracy measurement error of typically ±0.2%

with a maximum of ±0.5%. Furthermore, the chip has a 12-bit ADC resolution.

5 EXPERIMENTS

Each pruning algorithm has been applied to the MobileNetV2 model trained on CIFAR-10 with exactly five different sparsity levels, namely, 0.4, 0.5, 0.6 and 0.8 sparsity. It is important to define the sparsity level, as the term has been ambiguously used throughout literature, as pointed out by Blalock et al. [2]. In this work, the sparsity level is referred to as the fraction of the network parameters that has been pruned: $Sparsitylevel = \frac{Parameterspruned}{Totalparameters}$

For each sparsity level of each pruning method, the pruned model accuracy (after finetuning) is determined as described in subsection 4.3. Furthermore, the total energy consumed for exactly 2000 inferences is determined by taking a sample of the current energy consumption, as described in subsection 4.5, every 0.5 second. The start and end time of the inference (in miliseconds) is recorded, and each energy consumption data point is tagged with a timestamp. By means of this, it is possible to synchronize this data (required as the energy consumption is recorded on a separate board) and determine which energy consumption data points have been recorded during inference. After this, trapezoidal numerical integration is used to approximate the total amount of energy consumed during inference. The trapezoidal integration for N points can be described by the following formula: $\int_a^b f(x)\,dx = \frac{1}{2}\sum_{n=1}^{N}(x_{n+1}-x_n)[f(x_n)+f(x_{n+1})]$ To give one a clearer insight into this method, Figure 7, a figure originally published in the MATLAB software documentation [19], shows an example of numerical integration by means of the trapezoidal method for a sine function.

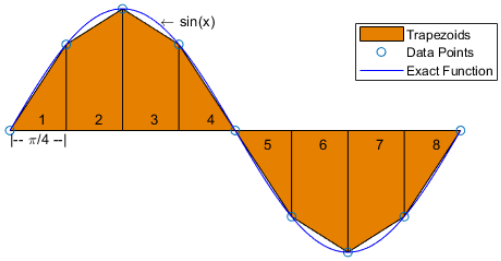

Fig. 7. Example of numerical integration by means of the trapezoidal method for a sine function [19]

6 RESULTS

In this subsection, the experimental results shall be presented in various graphs and tables. These results shall be analyzed in this section, whereas further discussion based on results presented in this section shall be made in section 7.

6.1 Analysis of results

Firstly, when analyzing Figure 8, it becomes apparent that when increasing sparsity, global unstructured random pruning as well as L1-norm pruning lose accuracy rather rapidly as compared to the Network Slimming and global unstructured magnitude pruning approaches. Now, it is expected that the global unstructured random pruning method loses accuracy rapidly, as to be pruned weights are, as its name suggests, selected randomly. Hence for global unstructured random pruning, there is a possibility that weights closer to one rather than those closer to zero are pruned, as is the case with the unstructured magnitude pruning. Furthermore, from this figure it becomes apparent that when requiring a sparsity larger than 60% on MobileNetV2 and still requiring acceptable accuracy, which we shall define as 50%, one might choose Network Slimming or global unstructured magnitude pruning over the other two pruning methods.

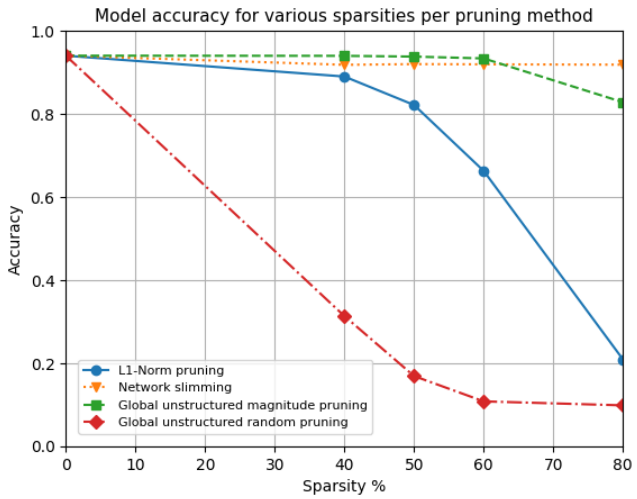

Fig. 8

Next, when analyzing Figure 9, it becomes apparent that the energy consumption of the unstructured pruning methods stays rather constant, whereas the structured pruning methods significantly decrease the energy consumption when increasing the sparsity of the model. A possible explanation for this could be, as previously referred to in this paper, the theory by Li et al., who mentions that magnitude based pruning of weights might not reduce the inference cost, and hence the energy consumption, to a large enough extent despite the significant pruning of weights due to the irregular nature of its sparsity. This result obtained from real-world inference data seems to support this theory. Another interesting pattern that can be seen in Figure 9 is that the Network Slimming pruning methods appears to converge around 100 mWh / 2000 inferences. It could be a variation on the theory by Li et al. mentioned above for the structure pruning methods, and that there might be a certain degree of irregular sparsity of channels at which the phenomenon occurs, which from this data appears to be before or around 40% sparsity. Future research might give insight in this matter by pruning a model at lower sparsity levels to see from what point this behaviour occurs, and possibly find the cause of why the behaviour occurs.

Now, let us examine Figure 10, which shows the relation between the model sparsity and the size of the model. It appears that all

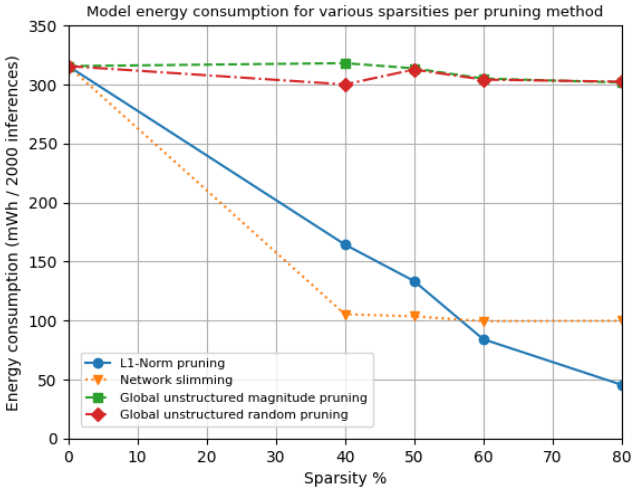

Fig. 9

pruning methods except for the Network Slimming, have some sort of linear relation between the sparsity of the model and the real-world model size. This is of course expected as one is removing data completely, or in the case of structured data, replacing near-zero weights by zero, allowing for efficient compression of the model. Like the apparent plateau that could be seen in Figure 9, again a plateau can be seen for the Network Slimming pruning method. As previously mentioned, a possible explanation for the observed phenomenon could be the fact that there might be a certain degree of irregular sparsity of channels at which the phenomenon occurs. Again, this phenomenon would need to be investigated further to be able to produce a theory that bears more certainty than the speculation presented. Despite the cause for the plateau behaviour not being known for certain, linking the graphs together, which has been done in Figure 11, which compares the real-world model size with the energy consumed per 2000 inferences. At first sight, for the structured pruning methods, there appears to be evidence for a linear relation existing between the size of the model and the energy consumed by the model, however the data for the L1-Norm pruning method in Figure 11 could also be considered to be somewhat concave upward, which is a possibility somewhat reinforced by the L1-Norm method data in Figure 12, which will be discussed in a later part of this section. Furthermore, from this figure it becomes clear that the energy consumption of models pruned with the unstructured pruning methods, does not significantly change when the model size changes as compared to the changes in model energy consumption of models pruned with the structured pruning methods when the model size changes. It seems to make intuitive sense that a model uses less energy when its models size is smaller. Especially for structured pruning methods, it would make sense that when removing complete parts of the network, there is simply less calculations to be done. Now, a possible explanation on why the unstructured pruning algorithms do not use significantly less energy when their model size decreases, could firstly be due to the theory by Li et al., proposing that magnitude based pruning of weights might not reduce the inference cost, and hence the energy consumption, to a large enough extent despite the significant pruning of weights due to the irregular nature of its sparsity. Secondly, a possible reason why the energy consumption does not decrease significantly when the model size decreases, is that because the pruned weights are set to be zero, they might be easily compressed, reducing the model size, however the calculations that need to be done, might still cost the same amount of energy i.e. the calculation $0 \cdot 256$ might cost the same amount of energy as the calculation $42 \cdot 256$.

Next, figure Figure 12 shows an interesting relation between the sparsity of the model against the energy consumption per Megabyte of the model. This relation is interesting as it gives insight if pruning the model to a higher sparsity, is more or less expensive in terms of energy consumption per MB of the model size. To explain this relation further, imagine a horizontal line in Figure 12. Such horizontal line would mean that whenever the sparsity changes, the amount of energy consumed for each MB of the model size stays the same, or in other words, when the model size gets multiplied by x, the energy consumption also gets multiplied by x Now, imagine a line trending downwards. This would mean that when the sparsity increases, the model uses less energy per MB of model size. In other words, when the model size increases by x, the energy consumption increases by $a \cdot x \mid a > 1$.

When looking at the data for the pruning methods in this figure, the first observation is that it appears that the amount of energy consumed per MB by the structured pruning methods increases exponentially when the sparsity of the model increases. This is an indication that, if one would like to reduce the energy consumption of a model by means of decreasing its model size, this is not the best approach for these two pruning methods when using MobileNetV2.

Furthermore, it appears that for the Network Slimming method, the energy consumption per MB of model size is roughly constant. As in the original data, the model size as well as the energy consumption stay roughly constant when changing the sparsity of the network, the ratio between these two variables also stays constant when sparsity changes, hence the curve (or lack thereof) in the figure produced by the data on the Network Slimming pruning method is trivial based on the original data. Lastly, and most interestingly, the data in Figure 12 shows, after pruning (from 40% sparsity onwards), a possible, alebeit slight, negative linear relation. This would mean that when the sparsity increases by say x, the the energy consumption decreases by $a \cdot x \mid a < 1$. This is additional evidence to the previously mentioned idea that the relation between the energy consumption and actual model size in MB for the L1-Norm pruning method is not strictly linear, but rather slightly upward concave. This could be useful if one would want to decrease the energy consumption of a model by means of decreasing its model size, as a model size reduction would result in a reduction in energy consumption greater than the magnitude by which the model size was reduced. To be able to confirm this proposed theory however, further research that focuses on this exact relation is required.

Now, when analyzing Figure 13, showing the energy consumption per percentage point of accuracy for each of the sparsity levels used for the experiments, it is apparent that the lowest amount of energy consumed per percentage point gained is the Network Slimming

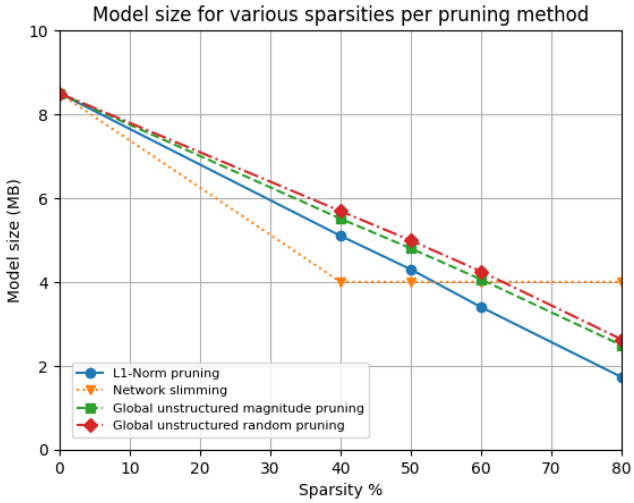

Fig. 10

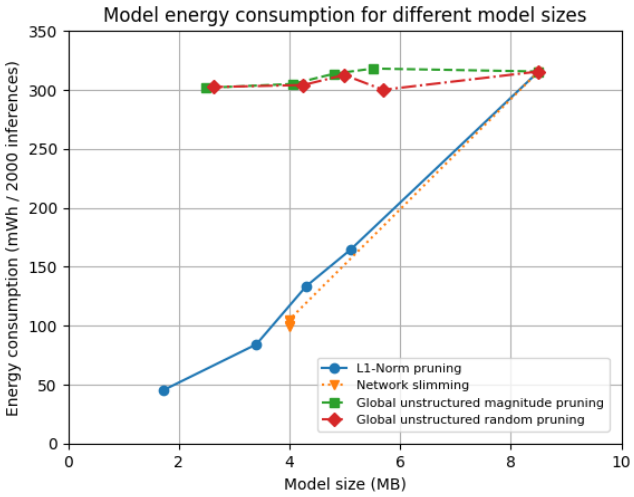

Fig. 11

method. The L1-Norm pruning method appears to consume somewhat more energy per accuracy percentage point gained as compared to the Network Slimming method, however the global weight pruning method consumes even more energy for each percentage point of accuracy gained. Lastly, the global random unstructured pruning method appears to firstly, consume significantly more energy per percentage point of model accuracy, and secondly appears to consume more energy per percentage point of accuracy when the sparsity of the model increases. A possible explanation for this behaviour could be derived from Figure 8 and Figure 9. Namely, it appears from Figure 9 that when the sparsity of the model pruned with the global unstructured random pruning increases, the model energy consumption does not decrease significantly, which could be explained by the previously mentioned theory by Li et al. [17].

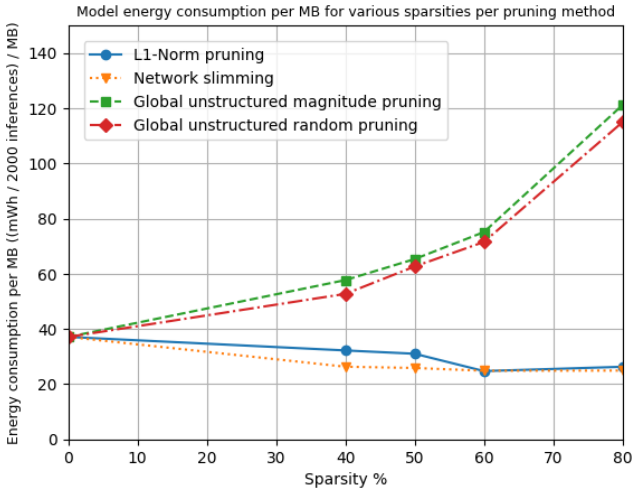

Fig. 12

At the same time, when increasing the sparsity of the model, the accuracy of the model appears to decrease drastically for the global random unstructured pruning method, as can be seen in Figure 8. A combination of both a drastically decreasing accuracy and a model energy consumption that stays approximately the same with increasing sparsity, results in relatively more energy being consumed for each accuracy percentage point for higher sparsities as compared to the energy consumption per percentage point of accuracy for lower sparsities.

Next, when analyzing Figure 14, which gives insight in the model accuracy percentage points for each MB of the model size for each of the sparsities experimented with. This is an interesting insight when one wants to maximize accuracy and minimize the model size in MB. One can see that, until about 50% model sparsity, network slimming provides the most accuracy percentage points per MB of the model size. At the same time, around this level of sparsity, L1-norm pruning and global unstructured magnitude pruning seem to provide approximately the same amount of accuracy percentage points per MB of their respective model sizes. All but the global unstructured magnitude pruning method appear to be somewhat constant (L1-Norm pruning and global unstructured random pruning do appear to have more variance however, but they do not appear to display a certain pattern). The global unstructured magnitude pruning method does however appear to show an upward concave, which would mean that for every scaling factor f the network size gets multiplied by, the accuracy decreases with a factor $p \mid p < f$. This would mean that if one wants to decrease the model size as much as possible and keep the accuracy as high as possible at the same time (keeping the energy consumption of the model out of the equation), it would be beneficial to select the global unstructured magnitude pruning as a pruning method from a set of the four methods analyzed in this work on MobileNetV2. Furthermore, the global random unstructured pruning method consistently has a fraction of the accuracy percentage points per MB of the model size compared to the other pruning techniques analyzed.

Lastly, in Table 1, average values over all tested sparsities have been laid out regarding the energy consumption per percentage point of accuracy of a pruned model, as well as the average amount of percentage points of accuracy against the model size in MB of the pruned model. Analyzing Table 1, provides one with similar insights to those already mentioned above when analyzing Figure 13 and Figure 14. It must be noted however that Table 1 does not provide one insights into possible patterns in data, such as the apparent increase of energy consumption per percentage point of accuracy when the sparsity increases for the global unstructured random pruning method and the increase in the amount of accuracy percentage points for each MB of the model size for the global unstructured magnitude pruning method.

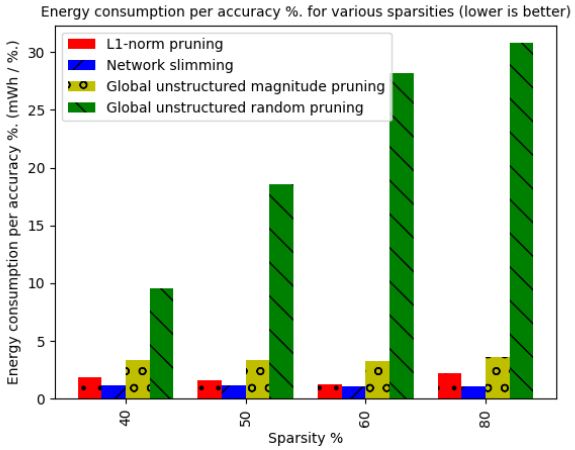

Fig. 13

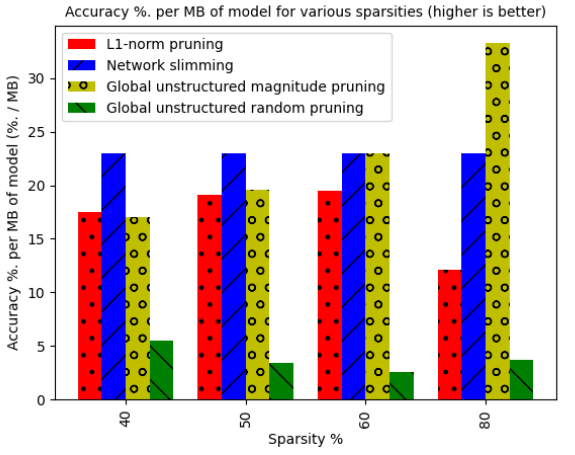

Fig. 14

7 DISCUSSION

After thorough analysis of the figures presented in section 6, one can come to a number of findings and hypotheses that shall be stated below.

7.1 Key takeaways based on result analysis

From the obtained data it has become clear that each pruning method that has been analyzed has its own trade offs regarding accuracy, energy consumption and model size. Network Slimming appears to be decreasing the model energy consumption by almost three times, and has a consistently high model accuracy of about 91%, regardless of the sparsity level. Although this method appears to reach some plateau for firstly the model size, and to a certain extent the model energy consumption from at least 40% sparsity onward. Due to these plateaus, despite the method's relatively constant high accuracy, excellent average energy consumption per 2000 inferences as well as a very acceptable average amount of accuracy percentage points per MB of the model size and on the lowest energy consumption per MB across almost all sparsity levels, it appears to not be possible to prune the model to achieve a model size lower than 4MB or a model energy consumption of less than approximately 100mWh per 2000 inferences. For this reason, if one's main goal is to decrease the model size as much as possible as well as if one's main goal is to achieve a minimal energy consumption, one has to take into account that if one's required model size or energy consumption is lower than the plateau values, one should most likely consider an alternative pruning approach. The relatively consistent high accuracy seen in the experiments conducted in this work, is similar to the accuracy behaviour observed in experiments done on VGGNet, DenseNet-40 and ResNet-164 trained on the CIFAR-10 dataset in the paper in which the method was originally proposed [18]. The model size and energy consumption plateauing behaviour is however not observed the the original paper, which does not report a real-world energy consumption and model size, but rather reports the theoretical number of float point operations and sparsity. From this work, it appears that the theoretical sparsity and theoretical amount of float-point-operations required for model inference does not necessarily reflect the real-world model size and energy consumption respectively. This could possibly be due to the framework used for the real-world experiments in this work, Tensorflow Lite, being unable to compress the network architecture further due to for example a certain degree of irregular sparsity of channels, as theorized above as a possible adaptation of the theory proposed by Li et al. [17].

Next, L1-Norm pruning appears to be very effective in decreasing the model energy consumption when model sparsity is increased, even below the earlier mentioned Network Slimming plateau value of around 100mWh per 2000 inferences. The energy consumption of the L1-Norm pruning per MB, as can be seen in Figure 12, is furthermore comparable to that of the Network Slimming pruning method. The largest trade off of using the L1-Norm pruning method however is the model accuracy, which as can be seen in Figure 8 decreases significantly more than all other pruning methods, with the exception of the global unstructured random pruning method.

Table 1. Average energy per accuracy percent point and average percent point per MB of model size

Pruning method	Average mWh/2000 inferences per %. (lower is better)	Average %. per MB of model size (higher is better)
L1-Norm pruning	1.73	17.03
Network slimming	1.11	22.98
Global unstructured weight pruning	3.41	20.78
Global unstructured random weight pruning	21.94	3.79

Furthermore, the energy consumption appears to decrease approximately linearly with the model size and the energy consumption per MB across all sparsity levels is not much higher than that of the Network Slimming pruning method. It appears that if one would like to effectively decrease model size and/or the energy consumption beyond the plateau level to which the Network Slimming approach seems to be constrained, and if a relatively low model accuracy compared to Network Slimming and global unstructured magnitude pruning (< 60%) is not an issue for the implementation, then using L1-Norm pruning to prune a MobileNetV2 model could be a beneficial choice. The results obtained from the experiments in this work regarding L1-Norm pruning appears to largely correspond to the paper originally introducing the pruning method [17]. In said paper, the curve describing the model accuracy compared to the sparsity for VGG-16 trained on CIFAR-10 is largely similar in its shape as to the curve seen in this work. The main difference between this work and the original paper being the real-world accuracy being lower, despite the trade-off curve between accuracy and sparsity being of approximately the same shape. This could possibly be due to a dissimilarity in the amount of training epochs or hyperparameter values between the two papers. Furthermore, the theoretical reduction in the number of float point operations described in the original paper, seems to be of similar magnitude for a given sparsity as the real-world reduction in energy consumption seen in this work. From the similarities between the two works, it appears that the theoretical performance of L1-Norm pruning is somewhat similar to its performance in the real world and that real world frameworks such as Tensorflow do not significantly seem to affect the performance of said pruning technique.

Global unstructured magnitude pruning has shown to be highly accurate even at high levels of sparsity, while not having the plateau constraint that Network Slimming has. Furthermore, the method appears to have a an upward concave relation between the amount of accuracy percentage points per MB of the model size and the sparsity of the model, making it an excellent pruning method for effectively reducing the size of a model, while retaining high accuracy. The major downside to the global unstructured magnitude pruning method is however its energy consumption, which appears to stay unchanged at all tested sparsity levels as comapred to the unpruned network. A possible explanation for the lack of energy consumption reduction is the in section 6 previously mentioned theory by Li et al. [17], in which it is proposed that possibly, magnitude pruning of wights might not reduce energy consumption due to the irregular nature of the sparsity. This appears to make the pruning method unsuitable when one's objective is to reduce the energy consumption. The method is however of great use when one intends to decrease the size of a model by the largest possible amount, while retaining a relatively high accuracy on MobileNetV2. In the paper in which the pruning method was originally proposed [10], one can see that the amount by which the number of weights decrease when the sparsity decreases, is three times as large as the amount by which the inference cost in float point operations drops when the sparsity increases by the same amount when applied to AlexNet trained on ImageNet. This behaviour would in itself lead to an increase in energy consumption per MB of model size when the sparsity increases, and is observed in the results of this work. It appears that in this work however, the behaviour that can be seen in Figure 12 is, despite it appearing a reasonable possibility when looking at the theoretical results from the original paper, not only due to the nature of the model as can be seen from its theoretical behaviour in [10], but also due to the theory by Li et al. described above as there appears to not simply be a smaller decrease in model energy consumption as compared to the decrease in model size when sparsity increases, but there seems to be no decrease in energy consumption at all.

Global unstructured random pruning, shows similar energy consumption concerns as global unstructured magnitude pruning. It seems plausible that, looking at the data regarding energy consumption for each of the tested sparsity levels as well as looking at the previously mentioned theory by Li et al. [17], that this might be the case for all unstructured pruning methods on MobileNetV2, and possibly other networks as well. To confirm such hypothesis however, further research into this matter is required. Furthermore, global unstructured random pruning appears to decrease in accuracy very significantly, with at a sparsity level of 40%, the model accuracy being 58%. less accurate than the pruning method with the next lowest accuracy at 40% sparsity, L1-Norm pruning. The model accuracy loss for global unstructured random pruning is of such magnitude, that it appears that beyond 60% sparsity, the method converges to an accuracy of approximately 10%, which for CIFAR-10, with 10 different classes to identify, means that the performance of the pruning method is as good as simply guessing of what category an image might be. As, when increasing the sparsity, the accuracy drops significantly while the energy consumption stays approximately constant, the energy consumption per accuracy percentage point, as is shown in Figure 13 increases in a concave down fashion (as the model accuracy eventually converges to 10%) and is almost an order of magnitude worse performing as compared to all other pruning methods.

Now, global unstructured random pruning was introduced as a baseline for other pruning techniques to compare to. After all, if a pruning method would perform worse than random pruning, it is likely not a very useful pruning method. Therefore, it was expected that all other pruning techniques would outperform this global unstructured random pruning method.

As there appears to be a number of similarities between the theoretical performance of the analyzed pruning techniques on network

architectures other than MobileNetV2 and their real-world performance obtained from the results of this work, there is reason to believe that similar real-world behaviour might be observed on models other than MobileNetV2, especially those with relatively similar network structures, as can be seen in Figure 5, such as MobileNet, Shufflenet and ShuffleNetV2. The effect of the pruning methods analyzed in this work on networks that are more dissimilar to MobileNetV2 might still behave similar to the behaviour seen in this work, although there might be more dissimilarities due to differences in the ability of TensorFlow Lite to compress pruned models in the real world caused by the nature of the sparsity induced by the pruning method on a specific network architecture, which is behaviour that is theorized to be observed in this work when applying Network Slimming on MobileNetV2. Further research on this topic, analyzing the real-world trade-offs of the pruning methods analyzed in this work on network architectures other than MobileNetV2 would however be required to confirm this hypothesis and to possibly generalize any conclusions across multiple network architectures.

8 CONCLUSION AND FUTURE WORK

Firstly, the conducted research has been able to provide one with highly useful insights regarding the trade-offs between accuracy, energy consumption and model size between unstructured global magnitude pruning, Network Slimming, L1-Norm structured pruning and random pruning on MobileNetV2, and hence has been successful in answering the research question at hand. Despite this however, there is a number of questions that are still left to be answered. Firstly, the behaviour observed for the Network Slimming pruning method, where a plateau is reached for both the model size as well as the model energy consumption, should be further investigated. A possible explanation for the observed phenomenon could be, as mentioned before in this work, an adaptation of the theory posed by Li et al. [17] regarding the inference cost of models pruned with a structured pruning methods, namely the possibility that there might be a certain degree of irregular sparsity of channels at which the phenomenon occurs. As compared to the unpruned model, the energy consumption and model size has significantly decreased at 40% sparsity. A possible experiment would be to measure the model energy consumption as well as the model size starting from 0% sparsity and iteratively increase the sparsity by a small percentage, say for example, 2%. The original theory posed by Li et al. regarding the inference cost of models pruned with structured pruning methods could furthermore be investigated further. This work adds a certain degree of credibility to this theory as no significant decrease in model energy consumption is observed when increasing the sparsity of the network for models pruned with any of the two unstructured pruning methods analyzed in this work.

In addition, by conducting the experiments carried out in this work on other model architectures as well, it might be possible to generalize conclusions based on observations done to a large range of architectures, as discussed in section 7.

Lastly, by conducting the experiments carried out in this work a large number of times (i.e. measure the energy consumption of a certain model pruned with a certain pruning method 50 times), which due to the time frame of this research was not possibles, based on the variance of the data, conclusions presented in this work might become more sound and statistically valid.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *Proceedings of machine learning and systems* 2 (2020), 129–146.

[3] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, and Diana Marculescu. 2017. Neuralpower: Predict and deploy energy-efficient convolutional neural networks. In *Asian Conference on Machine Learning*. PMLR, 622–637.

[4] Liyang Chen, Yongquan Chen, Juntong Xi, and Xinyi Le. 2022. Knowledge from the original network: restore a better pruned network with knowledge distillation. *Complex & Intelligent Systems* 8, 2 (2022), 709–718.

[5] Yanjiao Chen, Baolin Zheng, Zihan Zhang, Qian Wang, Chao Shen, and Qian Zhang. 2020. Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. *ACM Computing Surveys (CSUR)* 53, 4 (2020), 1–37.

[6] ONNX Runtime developers. 2022. ONNX Runtime. https://onnxruntime.ai/. Version: 1.12.0.

[7] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. 2019. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611* (2019).

[8] Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574* (2019).

[9] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. 2019. Estimation of energy consumption in machine learning. *J. Parallel and Distrib. Comput.* 134 (2019), 75–88.

[10] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).

[11] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*. 1389–1397.

[12] Mark Horowitz. 2014. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 10–14.

[13] Steven A Janowsky. 1989. Pruning versus clipping in neural networks. *Physical Review A* 39, 12 (1989), 6600.

[14] Michael I Jordan and Tom M Mitchell. 2015. Machine learning: Trends, perspectives, and prospects. *Science* 349, 6245 (2015), 255–260.

[15] Ehud D Karnin. 1990. A simple procedure for pruning back-propagation trained neural networks. *IEEE transactions on neural networks* 1, 2 (1990), 239–242.

[16] Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems* 2 (1989).

[17] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

[18] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*. 2736–2744.

[19] MATLAB. 2021. *9.11.0.1809720 (R2021b)*. The MathWorks Inc., Natick, Massachusetts.

[20] Seyedeh Yasaman Hosseini Mirmahaleh and Amir Masoud Rahmani. 2019. DNN pruning and mapping on NoC-Based communication infrastructure. *Microelectronics Journal* 94 (2019), 104655.

[21] Bert Moons, Koen Goetschalckx, Nick Van Berckelaer, and Marian Verhelst. 2017. Minimum energy quantized neural networks. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 1921–1925.

[22] MG Sarwar Murshed, Christopher Murphy, Daqing Hou, Nazar Khan, Ganesh Ananthanarayanan, and Faraz Hussain. 2021. Machine learning at the network edge: A survey. *ACM Computing Surveys (CSUR)* 54, 8 (2021), 1–37.

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

4510–4520.

[24] Xavier Suau, Luca Zappella, and Nicholas Apostoloff. 2018. Network compression using correlation analysis of layer responses. (2018).

[25] Texas Instruments 2015. *INA219 Zerø-Drift, Bidirectional Current/Power Monitor With I2C Interface*. Texas Instruments. Latest rev..

[26] Swagath Venkataramani, Anand Raghunathan, Jie Liu, and Mohammed Shoaib. 2015. Scalable-effort classifiers for energy-efficient machine learning. In *Proceedings of the 52nd Annual Design Automation Conference*. 1–6.

[27] Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376* (2020).

[28] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. 2021. Emerging paradigms of neural network pruning. *arXiv preprint arXiv:2103.06460* (2021).

[29] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. 2017. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5687–5695.

[30] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. 2018. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9194–9203.