Self-adaptive neural network for edge devices.

Adapting the secondary functional aspects for image classification models.

submitted in partial fulfillment for the degree of master of science

Matteo Rapa
14449811

Master Information Studies
data science
Faculty of Science
University of Amsterdam

Submitted on 24.06.2023

	UvA Supervisor
Title, Name	Dolly Sapra
Affiliation	UvA Supervisor
Email	d.sapra@uva.nl

ABSTRACT

Vision tasks using deep models need to be performed at multiple venues, namely at the cloud and the edge due to privacy requirements, platform reliability and latency requirements among others. However many state-of-the-art deep models cannot be feasibly deployed at the edge due to memory limitations and inference speed. Additionally each application has a distinct combination of such requirements based on the environment it operates in. The proposed research question asks how a self-adaptive neural network could be designed to cater for the different requirements of each execution venue in the cloud-to-edge continuum. Different trade-offs can be present for accuracy against other requirements such as inference speed or power efficiency.

KEYWORDS

Cloud-to-edge continuum, Computer Vision, Extra-functional aspects, Pruning

GITHUB REPOSITORY

https://github.com/matteorapa/self-adaptive-neural-networks

1 INTRODUCTION

Computer vision tasks using deep machine learning models gave way to a number of real-world applications such as autonomous vehicles, facial recognition and object detection. However, state-of-the-art deep models have grown rapidly in number of parameters running in the billions. Given the numerous applications of vision tasks, it may not be feasible to perform the inference only at data centers on the cloud. Certain tasks which have latency and privacy requirements may be more practical and reliable to be executed at the edge device; near the source of the data. This prevents the need of sensitive data being transferred over the network, which helps with reducing security vulnerabilities and privacy concerns as the data is kept at source. In this manner the best execution venue should be based on the requirements of the use case [10]. This concept is known as the cloud-to-edge continuum, which defines the spectrum of locations and connected devices at which the processing can be performed, namely at the cloud or at multiple edge devices (refer to figure 1).

Some tasks and processes have some requirements, such as real-time processing and/or limited resources such as battery life, processing power and memory. These are known as the extra-functional aspects as they influence how the system needs to work but do not change the function of the model. Applications deployed at scale on the cloud may also have strict latency requirements, i.e the amount of time that an inference is expected to take to be able to provide good quality of service while also be capable of handle the rate of requests needed. An edge device is usually near the source of data and may have limited resources when compared to the cloud. These are referred to as the execution venue, which is the place or type of device that the model will run the inferences on. The following are some motivational examples of the specific requirements needed by different execution venues.

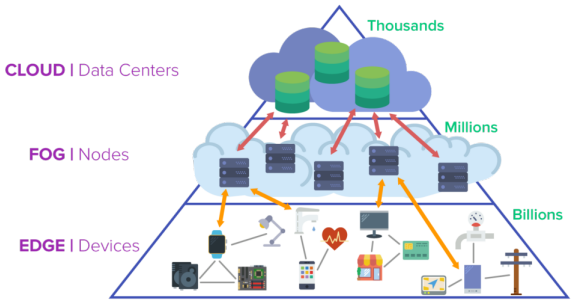

Figure 1: Cloud-to-edge continuum, inference can be made at the cloud or at edge devices which collect data from sensors at the edge. Taken from Pubnub. [1]

1.0.1 Execution Venues. Execution venues for models refer to the various environments or platforms where machine learning models can be deployed and executed. These venues can include a range of computational systems, each with its own characteristics, advantages, and limitations.

1.0.2 Resource varying platforms. Devices that have varying levels of available resources or computational capabilities. These platforms often operate under resource constraints, meaning they have limitations in terms of processing power, memory, energy consumption, or other resources.

Medical edge devices need to keep patient data private hence the processing (inferences) needs to be done at the edge, on the device which may be running on batteries or have limited processing capabilities. Autonomous vehicles need to process large input from an array of cameras surrounding the car. Due to the real-time aspects of this it may not be reliable to process this on the cloud due to network interruptions, it is more practical to be processed at source. Hence, processing all this efficiently is a must, as a high power usage from these driving capabilities comes at a detriment to the car's driving range.

Current techniques like branching to reduce inference time (measured in milliseconds), reduce processing load (measured in FLOPS) and reduce memory usage (number of weights in thousands/millions). However most of these techniques aim to maximise efficiency without taking into consideration the environment for each of the use cases.

The primary research question of this work is: **How can we design a neural network that can self-adapt its extra-functional aspects depending on requirements of the execution venue?** The related sub-questions are the following:

- *What are the trade-offs when for different extra-functional requirements?*
- *Can the adaptability be fine-tuned into multiple levels when focusing on an extra-functional aspect?*
- *How can we limit the impact on accuracy after adapting for an extra-functional requirement?*
- *Which combination of extra-functional requirements can be practically used together?*

In the next section, relevant state of the art work are discussed, focusing on techniques used to reduce resources used during inference.

2 RELATED WORK

In this section, a critical review is made on state-of-the-art work that focus on the adaptability of extra functional aspects for platforms with constraint or varying resources

The majority of earlier literature in dynamic neural networks; networks that are capable of dynamically changing the way that an inference is performed to satisfy one or more extra functional aspects, focus on reducing extra-functional aspects, such as multiply-accumulate operations (MACs) and memory usage (measured in MB) statically. A big innovation to address this issue was done by Howard et al with the development of MobileNet [6]. The primary change was that image classification could be done in a feasible manner on devices with less computation resources, such as edge devices like smartphones while keeping the accuracy high.

Further work done in dynamic neural networks; networks that are capable of dynamically change the way that an inference is performed to satisfy one or some extra functional aspects, focus on adapting the model based on the input. A key finding that motivated such types of dynamic networks is the fact not all examples to be classified, need to utilise the full complexity of the model to be classified with high accuracy. Multiple works [[12], [8]]show that a large proportion of examples can be classified with accuracy within the first few layers of a deep model, hence are considered easy examples. This led to works using branching techniques to dynamically decide when an image can be classified with confidence without needing to pass through all the layers. Teerapittayanon et al developed BranchyNet [12], a model that is capable of early-exiting for most images fed into the model for inference, which led to reduction in processing required. A drawback of such an approach is that in scenarios where the execution environment becomes more resource restrictive, the model cannot guarantee reduction in inference speed for all examples.

Earlier works do not consider adaptability of these aspects based on the changing environment where the task is being performed [3]. More recent work address explore the need for adaptability in deep models, due high number of parameters and dynamic resource environments.

In contrast, recent works [11],[13], [15] propose methodology capable of adapting the neural network, thus altering the numbers of weights/parameter, subsequently reducing the numbers of multiple-accumulate operations.

Sun et al. propose SteppingNet [11], a novel approach that dynamically increases the accuracy of neural networks by constructing a series of subnets with incremental enhancements in accuracy as more MAC operations become available. This design enables a trade-off between accuracy and latency during inference, offering flexibility based on specific resource constraints and varying platforms.

A key advantage of SteppingNet is that it maximizes computational reuse among subnets. By sharing weights among subnets, only one copy of the neural network needs to be stored. The larger subnets in SteppingNet are built upon smaller subnets, allowing for direct reuse of intermediate results without recomputation. This dynamic reuse of computations enables SteppingNet to make real-time decisions on whether to execute further MAC operations, enhancing inference accuracy on-the-fly.

Furthermore, SteppingNet adapts the structures of subnets based on the allowed numbers of MAC operations, incrementally improving their accuracy. This incremental nature makes SteppingNet well-suited for scenarios where an initial decision needs to be made promptly and refined further with additional computational resources or execution time. Experimental results demonstrate that SteppingNet effectively improves accuracy while optimizing computational resources. It consistently outperforms state-of-the-art methods, achieving higher accuracy under the same limit of computational resources.

In summary, SteppingNet introduces a dynamic approach to enhance neural network accuracy by constructing a series of subnets with incremental improvements. It maximizes computational reuse, adapts to resource constraints, and achieves superior accuracy with limited computational resources compared to existing methods.

Vu et al. demonstrated the favorable performance of Any Width Networks (AWNs) compared to existing methods, offering maximal control during inference [13]. AWNs are a family of adjustable-width convolutional neural networks that allow fine-grained control over the width of operations at inference time. The study showcased that AWNs effectively eliminate the impact of varying batch statistics on multi-width functionality. They provide maximal control while ensuring smooth and consistent performance across different widths, without the need to store multiple versions of network layers. Additionally, the use of triangular convolutions in AWNs was found to be naturally suited for multi-width networks. Their results reveal the improved performance of the triangular convolutional layer design compared to standard convolution as the network width is decreased. The performance degradation demonstrates a graceful behavior, making it well-suited for applications involving multiple network widths. The findings highlight the potential of AWNs as a promising research direction for resource-constrained inference. They offer granular control and maintain performance consistency, making them an attractive solution for adaptive network width requirements.

Yu et al present Slimmable networks [15]. present a simple and general method to train a single neural network executable at different widths, permitting instant and adaptive accuracy-efficiency trade-offs at runtime. Instead of training individual networks with different width configurations, we train a shared network with switchable batch normalization. At runtime, the network can adjust its width on the fly according to on-device benchmarks and resource constraints, rather than downloading and offloading different models. To train slimmable neural networks, a naive approach was initially employed, directly training a shared neural network with different width configurations. However, this approach yielded extremely low top-1 testing accuracy of around 0.1% on the 1000-class ImageNet classification task. The primary issue with this approach was identified as the inconsistency in batch normalization statistics due to varying means and variances of aggregated features caused by different numbers of input channels in the previous layer. These inconsistencies propagated layer-by-layer, resulting in inaccurate batch normalization statistics during testing.

Table 1: Summary of techniques used in related works.

Paper	Technique	Year
Earlier Works		
Teerapittayanon et al [12]	Branching	2017
He et al [4]	Channel Pruning	2021
Laskaridis [8] et al	Early-Exiting	2021
Wang et al [14]	Adaptive Input Scaling	2021
State-of-the-art		
Yu et al [5]	Switchable Batch Normalization	2019
Vu et al [7]	Triangular convolutional layer	2020
Sun et al [11]	Subnets	2023

In an attempt to address this issue, an incremental training approach, also known as progressive training, was investigated using Mobilenet v2 on the ImageNet classification task. A base model (A) was trained, and extra parameters (B) were added to create an extended model (A+B). The additional parameters were fine-tuned along with the fixed parameters of A. Although this approach exhibited stability during training and testing, the top-1 accuracy only increased marginally from 60.3% for A to 61.0% for A+B. In contrast, individually trained Mobilenet v2 with 0.5x width achieved an accuracy of 65.4% on the ImageNet validation set. The lack of joint adaptation of weights A and B, due to the incremental training approach, significantly deteriorated the overall performance.

Motivated by these investigations, a simple and highly effective approach called Switchable Batch Normalization (S-BN) was proposed for slimmable networks. S-BN utilizes independent batch normalization for different switches in a slimmable network, thereby addressing the issue of feature aggregation inconsistency. Each switch has its own batch normalization layers, independently normalizing the feature mean and variance during testing. S-BN also enables joint training of all switches at different widths, allowing for the simultaneous update of all weights to achieve improved performance. It offers the advantages of negligible additional parameters and minimal runtime overhead for deployment. S-BN resolves the problem of feature aggregation inconsistency, and the scale and bias parameters in S-BN can potentially encode conditional information regarding the width configuration of the current switch. Additionally, the number of extra parameters in S-BN is negligible, and the runtime overhead for deployment is minimal. Batch normalization layers can be fused with convolution layers during inference, and for slimmable networks, the fusion can be performed on-the-fly at runtime without incurring additional costs in terms of time or memory. The results for slimmable neural networks are state-of-the-art, which achieve similar (and in many cases better) ImageNet classification accuracy than individually trained models of MobileNet v1, MobileNet v2, ShuffleNet and ResNet-50 at different widths respectively.

A summary of papers for earlier and state-of-the-art related works, refer to table 1.

This motivated the thesis project to address the gap in literature on neural networks that can adapt (the extra-functional aspects) themselves based on the requirements of the execution venue environment using a technique called channel pruning which aims to reduce parameters deep neural network. In the next section a practical example is given for why model adaptability is key for specific applications.

3 MOTIVATIONAL EXAMPLE

In this section, the benefits of forming a self-adaptive methodology is explored through the use of an example. For models that consist of convolution layers, used throughout state-of-the-art computer vision task increase accuracy by increase the number of multiple-accumulate operations (such as ResNet). In real scenarios the available resources can vary due to the changes in its environment. The example application performs CNN-based image recognition on a battery powered smartphone.

The smartphone can have different power modes which dictate the maximum amount of watts that the processor can use. This impact the inference speed when running a model. For real-time applications such as object detection the impact in inference speed would be to great.

- **Scenario A: No constraints** The current environment of the device meets the computational requirements of the model to provide an inference within a target inference time range.
- **Scenario B: Reduced MACs due to restricted CPU** The idea is to have a single model that can adapt its processing requirements to reach a target inference speed. When the power state changes to for example *Low Power Mode* (30% reduction in processing power), the model uses less multiple-accumulate operations to perform an inference.
- **Scenario C: Limited available memory** It may be the case that due to other applications running on the device, the available memory is less that the memory used for inference. In this case the model can adapt to reduce its memory usage by further reducing the amount of parameters needed to complete the inference.

4 METHODOLOGY

In this section, TrimmingNet is presented, an adaptability design capable to changing computational resources used at run-time based on the environment based on pruning and fine tuning techniques.

4.1 Background

Channel pruning is a technique used in neural network compression and optimization to reduce the number of channels (also known as filters) in convolutional layers. Convolutional neural networks (CNNs) typically consist of multiple convolutional layers, each comprising a set of filters that extract different features from the input data. However, not all channels contribute equally to the network's performance, and some channels may be redundant or less informative. The goal of channel pruning is to identify and remove unnecessary channels while preserving or even improving the network's performance. This process involves analyzing the importance or relevance of each channel and selectively pruning those that have the least impact on the network's accuracy. By removing these redundant channels, channel pruning reduces the computational complexity and memory footprint of the network, leading to faster inference speed and lower resource requirements. There are several approaches to channel pruning, including magnitude-based

pruning, importance-based pruning, and optimization-based pruning. Magnitude-based pruning involves ranking the channels based on their magnitudes (e.g., weights or activations) and removing those with the smallest magnitudes.

Neural networks with multiple layers, state-the-art vision models such as CNN can have potentially hundreds of layers. These layers are used for data engineering, in order to extract features that could be used to classify the image. One of the prominent challenges of deep learning in image classification is to do it efficiently due to the high number of parameter in state-of-the-art models proposed.

Existing research shows a number of techniques that may be to reduce resources needed to run the models at the edge [3], which are prone to limited resources. The goal of this methodology is to design an adaptive technique to reduce the amount of resource use depending on where the model will run and the state of it's operating environment (refer to figure 2).

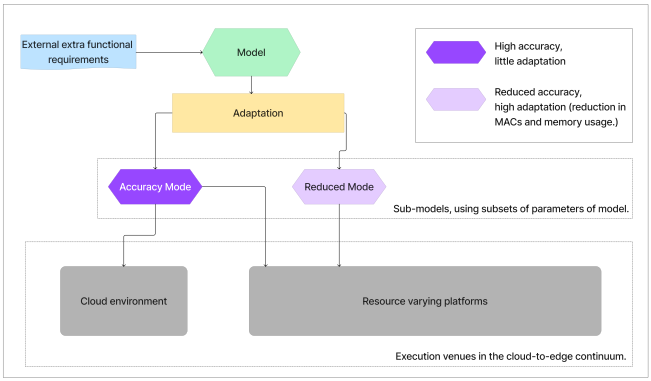

Figure 2: Diagram of the showing adaptive channel pruning for different execution venues in the cloud-to-edge continuum. In this example, multiple devices and places where the image classification models run have different extra-functional requirements. It is expected that adapting the model for less computation, memory and energy usage might impact classification accuracy at different degrees.

4.2 Training of TrimmingNet

The focus will be made on channel (also known as width) pruning; which may be used to reduce the size of a neural network by removing connections or weights. Pruning techniques can be classified into two categories: unstructured and structured pruning. The paper will focus on adapting a structured pruning technique, channel pruning where weights of kernels from the output of a two dimensional convolution layers are removed (or the value is set to zero). The aim is to reduce multiply-accumulate operations in order to use less time, energy and memory resources. For convolution layers the amount of MACs can be expressed as follows:

$$MACs = (K^2) * C_{in} * H_{out} * W_{out} * C_{out} \quad (1)$$

where K is kernel width and height (for square kernels), C_{in} is number of input channels, C_{out} is number of output channels,H_{out} and W_{out} height and width of the output matrix respectively.

In this paper, the number of filters or channels are reduced dynamically in order to reduce computational requirements, thus reducing multiple-accumulate operations as shown in equation 1. To achieve this the output of the layer is reshaped, omitting certain channels. This is shown to yield little drop in accuracy while reducing the amount of computations needed by a significant margin[**?**]. The implementation of the adaptive design will be applied on existing state-of-the art ResNet model composed of multiple convolution layers. Additionally they have been selected due to their widespread use in computer vision tasks. Furthermore, the weights found in each of the layers of the model are downloaded and visualized. This step is important as it provides insights into the amount of parameters, memory usage, and number of multiply-additions (related to inference speed) needed for each layer. The channel pruning is applied to all convolution layers, but experimentation is needed to identify which layers in parts of the model contribute the least accuracy drop, since accuracy is the main functional requirement. By selecting the right layers for adaptation, it minimizes the impact on the overall performance of the model.

4.2.1 Pruning the channels. The training process starts using a set of pre-trained weights. Then pruning is applied in multiple steps, where the history of which channels were pruned are maintained. When a layer is pruned, the dependant layers need to be also pruned in a similar manner since the weight matrix for a convolution layer is dependant from previous layer. Additionally, the input width other layers depend on the output width of the previous layer. To manage these dependencies during pruning, this paper uses a dependency graph introduced in [2]. Fang et al propose a recursive process where they start with a layer, such as w1, and examine its dependencies on other layers, such as w1 w2. Then, they use the dependent layer (w2) as a new starting point and recursively expand the dependency, which leads to w2 w3. This process continues until a transitive relation is established, in this case, w1 w2 w3. Moreover deciding which channels to prune contributes to retaining the performance of the pruned model. In this paper we select the channels to remove based on the magnitude of importance.

Fine-tuning after pruning After pruning the result number of parameter across the network are reduced, however the accuracy drops significantly. Hence fine-tuning was performed to regain back the majority of the accuracy. Fine-tuning involves retraining the model on a subset of the training data to improve performance as shown by existing works [**?**]. Fang et al show that for the ResNet-50 model they reduce the MACs from 4.13 GMACs to 1.99 GMACs, with a modest drop of Top-1 accuracy from 76.15% to 75.83%, a change of -0.32% after 80 epochs of fine-tuning [2].

4.2.2 Rebuilding the model. For the model to be adaptable, the full model needs to work in tandem with the pruned version by sharing the weights. To achieve this, two sets of weights are used together to populate the full width of channel of the original model; the tuned weights from the pruned model and the pre-trained weights from the original model. The tuned weights of pruned model used for channels that were not originally pruned, while the weights from the pre-trained original model for the dropped channels during the pruning. During the pruning steps, a history was kept of the dropped channels for each layer. Using this history, the layers are

rebuilt in reverse; starting from the last pruning step, until the original size of the model.

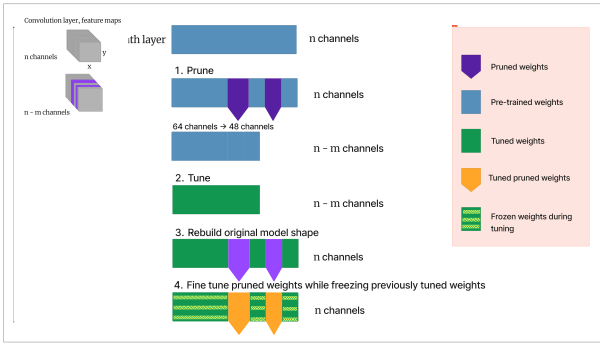

Figure 3: Diagram of the steps taken to setup the adaptability of the models. Different colors indicate channel groups with different weight functions, explained in the legend.

4.2.3 Fine-tuning after rebuilding. In this study, the parameters are optimised to be used with multiple modes of adaptability. After rebuilding a degradation in validation accuracy is observed due to the use of multiple parameter sets. Fine-tuning may be necessary to achieve the best results however if all the parameters are retrained, then performance of pruned model is affected. To avoid this, only the channels which are pruned; and not part of the smaller pruned model, are fine tuned in order to achieve adaptability

Add reference to the results here

This process can be repeated many times to achieve multiple level of adaptability depending on the use case (refer to figure 6).

4.3 Performing Inference

Depending on the resource constraints of the environment where the model is running, the model adapts the multiple-accumulate operations needed for inference in iterative steps excluding specific drop-able channel sets (during inference time). What amount of sets are appropriate to be dropped is decided by a third party controller that has access to the relevant environment metrics, whenever the environment changes either more sets get dropped or added as shown in figure 4. To increase the granularity of adaptability, more fine-tuning of sets needs to be prepared before hand. This behaviour models scenarios, inspired by Minakova et al [9], based adaptability depending on the specific use case of the classification model.

4.4 Experimental Setup

In this paper, the results are demonstrated using the ResNet18 and ResNet50 models.

The experiments were set up as follows: Firstly to find the feasibility of pruning, and find what amount of pruning, in terms of percentage yield the best trade. Using a pre-trained model, $\mathcal{T}$ channels are removed in a single step, where the top-1% and top-5 % validation scores are used to understand at what point the validation accuracy drops more significantly. Refer to 6. Pruning is done in step until the target pruning amount is reached. It was decided to test for the following pruning amounts (measured in percentage

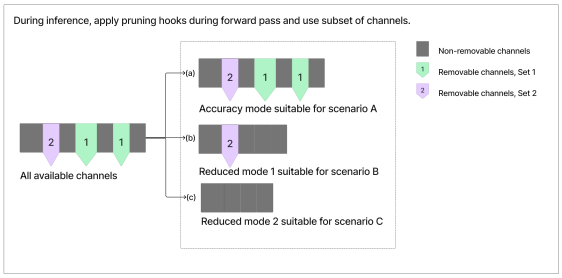

Figure 4: Diagram illustrating how the different channel groups which may be dropped to use a smaller model during inference. All channels are used for maximum accuracy at the highest computational cost.The green set may be removed to reduce computation to a specific level. Additional the pink set may be dropped for a lower level.

of channels) p = 1.5625, 3.125, 6.25, 12.5, 25, 50. SGD for optimizer. momentum, learning rate = 0.01, run for epochs 10

After pruning the resultant model has significant drop in validation accuracy as the number of channels pruned increase. As shown in previous work [references here] fine-tuning is used to retrain the parameter to recover back the lost accuracy. In this study, two form of fine-tuning strategies are explored: fine tuning in one go and fine-tuning after each pruning step.

The following experiment consists of validating the rebuilt model with the original model of the same parameter count. The rebuilt model is then fine-tuned without updated the tuned weights of the previous size.

4.4.1 ***The dataset.*** ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonymset" or "synset". There are more than 100,000 synsets in WordNet; the majority of them arenouns (80,000+). The ImageNet dataset, organized based on the WordNet hierarchy, comprises an extensive collection of images. Each synset, representing a meaningful concept in WordNet, is associated with multiple words or word phrases. With more than 100,000 synsets in WordNet, predominantly nouns, ImageNet aims to provide an average of 1000 images to depict each synset. These images undergo quality control and human annotation, ensuring accuracy. The ultimate goal of ImageNet is to offer a comprehensive collection of tens of millions of well-labeled and categorized images that correspond to the concepts in the WordNet hierarchy (Russakovsky et al., 2015). The ImageNet dataset, it consists of over 14 million images distributed among 1,000 distinct classes for image classification purposes. Firstly, it is important to understand the dataset's size and structure. The dataset comprises a training set, which contains more than 1.2 million images, and a validation set, containing over 50,000 images. Each image in the dataset is assigned a single label from a pool of 1,000 possible categories.

4.5 Evaluation

Using the validation set, comprising of 50,000 images and their ground truth, the results of the experimentation are recorded.

The performance in terms of classification of the multiple modes adapted model is evaluated by measuring the top 1% accuracy, top 5% accuracy. Top-1 accuracy focuses on the model's ability to predict the single most likely class correctly. It measures the percentage of images for which the model's highest-ranked prediction matches the ground truth label.

Top-5 accuracy is a more relaxed metric that allows for some flexibility in the model's predictions. It measures the percentage of images for which the correct class is present within the top 5 predictions made by the model.

However further evaluation is to be made for the extra-functional aspects which memory usage, mult-adds and energy usage. By evaluating the performance of the adapted model, we can identify the best adaptability level for a specific use case (refer to figure 2).

The performance of the adapted model is plotted against each of the metrics mentioned to identify the optimal adaptability levels that achieve the highest efficiency gains in terms of memory, energy and inference speed while minimizing impact of the classification accuracy.

5 RESULTS

In this section, the essential outcomes and significant findings that emerged from the experimentation will be presented. Following this, a comprehensive analysis and key findings that contributed to the research questions of the will be provided.

The performance in terms of classification is evaluated by measuring the top 1% accuracy, top 5% accuracy as shown in existing works . However further evaluation is to be made for the extra-functional aspects which memory usage, mult-adds and inference speed.

The performance of the adapted model is plotted against each of the metrics mentioned to identify the optimal adaptability levels that achieve the highest efficiency gains in terms of memory, energy and inference speed while minimizing impact of the classification accuracy.

First we look at the results before fine-tuning, after we apply structured pruning. The experiments carried out have tests with: both the linear and convolution layers pruned (refer to figure **??**) and only the convolution layers pruned (refer to figure **??**). When comparing the two plots together as shown in figure 5 we notice worst validation scores when pruning both, which is expected as there are more layers in the model being pruned.

We then look at the validation scores when fine tuning is performed. We also measure the reduction in memory usage, inference speed and multiplication-additions operations.

What are the trade-offs when for different extra-functional requirements?

It is observed that when pruning the memory usage during inference drops rather linearly with the pruning amount, this is due to the reduction of trainable parameters. The same effect is shown for the MACs as shown in figure .

6 DISCUSSION

In this section the results of the study will be discussed in comparison with other state-of-the art works. To add discussion here...

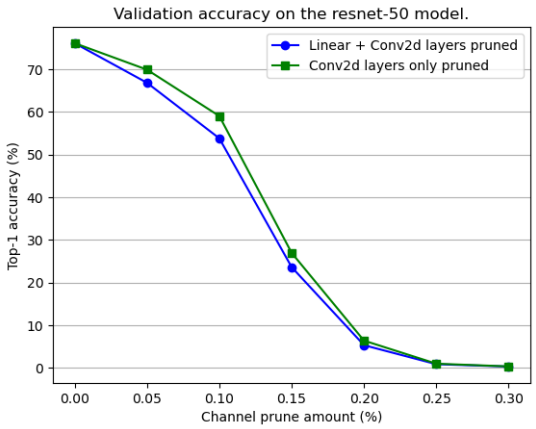

Figure 5: Line chart showing the top 1 % classification accuracy (y-axis) against the pruning amount in percentage (x-axis). The chart illustrates the performance differences when pruning: the linear and convolution layers vs only the convolution layers.

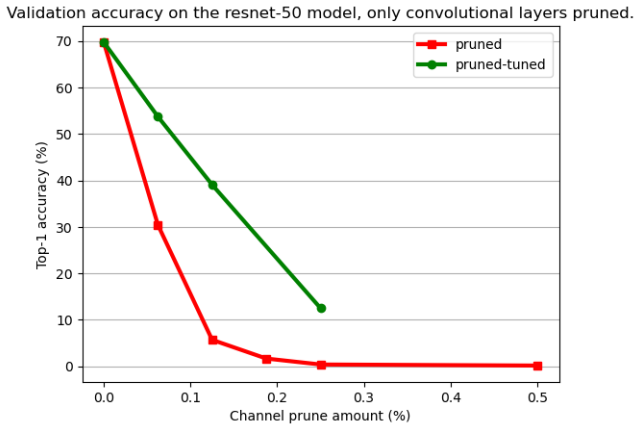

Figure 6: Line chart showing the top 1% classification accuracy (y-axis) against the pruning amount in percentage (x-axis). The ResNet50 model [?] with pre-trained weights is pruned using structured pruning on the linear and convolution layers. After 10% of pruning a steeper drop in accuracy is noticed. The model has not been fine-tuned after pruning.

6.0.1 Limitations. Due to the methodology applied, in order to reduce the multiple-accumulate operations multiple rounds of fine tuning is needed to be performed. beforehand.

While the aim is to minimize reduction in Top-1% accuracy, such technique might not be useful for applications that are used within safety systems, such as object classification for autonomous vehicles.

6.0.2 Future Work. Further work is needed to explore different strategies to select the channels, and study which are the most suitable match for the technique used.

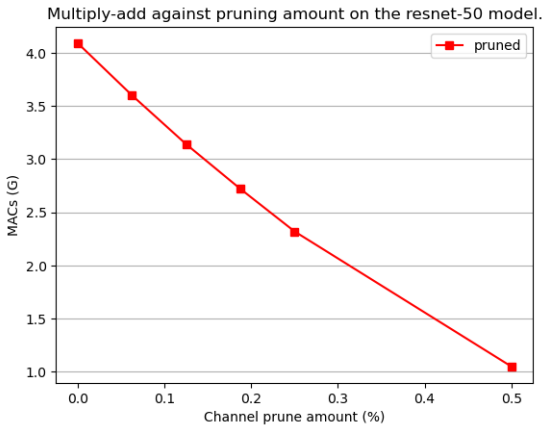

Figure 7: Line chart showing the number of multiple-accumulate operations% (y-axis) against the pruning amount in percentage (x-axis).

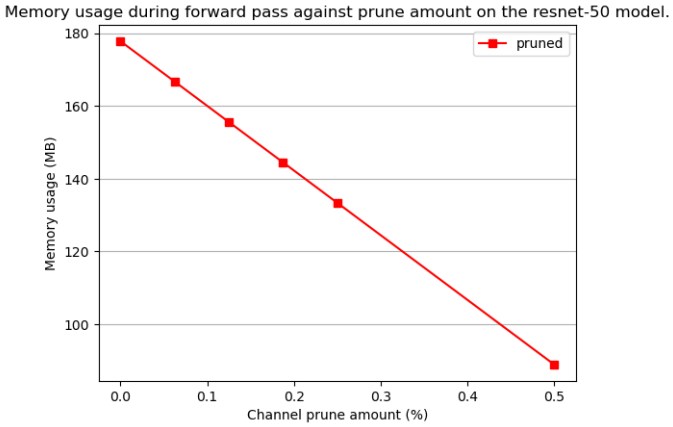

Figure 8: Line chart showing the memory usage (MB) during inference% (y-axis) against the pruning amount in percentage (x-axis).

7 CONCLUSION

In conclusion, this paper has explored various techniques and strategies to reduce the number of multiple-accumulate (MAC) operations during inference time. The significance of MAC operations in deep learning models cannot be understated, as they contribute significantly to where such models can be used, and how they can handle platforms with varying environment. By dynamically minimizing these operations, we can achieve faster inference or increase inference speed when parts of the resources are no longer available due to a change in the environment.

Throughout this study, we have discussed several approaches for reducing MAC operations, including network architecture design, quantization, pruning, and low-rank approximation. Each method has its strengths and limitations, but when combined or applied selectively, they can yield substantial improvements in terms of computational efficiency without compromising the model's performance. Network architecture design has proven to be an effective strategy for reducing MAC operations by carefully designing the structure of the neural network. Techniques such as skip connections, depthwise separable convolutions, and attention mechanisms help in reducing redundant calculations and improving the overall efficiency of the model.

Initial results show that such a methodology design is suitable to build a self-adaptable method. It is also shown that multiple adaptability levels is also possible as long as fine-tuning is done for each mode.

Pruning is another valuable technique that involves removing unnecessary connections or filters from a network, leading to a reduced number of MAC operations. By iterative pruning less important weights or filters, the model can be made more compact and computationally efficient, without significant loss in accuracy.

In conclusion, reducing multiple-accumulate operations during inference time is crucial for achieving efficient and faster deep learning models. By employing a combination of network architecture design, quantization, pruning, and low-rank approximation, significant improvements in computational efficiency can be achieved without sacrificing accuracy. It is important to consider the trade-offs between computational cost, model size, and performance to determine the most suitable combination of techniques for a given application. Future research should continue to explore novel approaches and optimizations to further enhance the efficiency of deep learning models during inference.

REFERENCES

[1] [n. d.]. Moving the cloud to the edge with edge computing.

[2] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. [n. d.]. DepGraph: Towards Any Structural Pruning. https://github.com/VainF/Torch-Pruning

[3] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. 2022. Dynamic Neural Networks: A Survey. , 7436-7456 pages. Issue 11. https://doi.org/10.1109/TPAMI.2021.3117837

[4] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel Pruning for Accelerating Very Deep Neural Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 1398–1406. https://doi.org/10.1109/ICCV.2017.155

[5] Torsten Hoefler, Dan Alistarh, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. , 124 pages. http://jmlr.org/papers/v23/21-0366.html.

[6] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (4 2017). http://arxiv.org/abs/1704.04861

[7] Adarsh Kumar, Arjun Balasubramanian, Shivaram Venkataraman, and Aditya Akella. [n. d.]. Accelerating Deep Learning Inference via Freezing.

[8] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D. Lane. 2021. Adaptive Inference through Early-Exit Networks: Design, Challenges and Directions. (6 2021). https://doi.org/10.1145/3469116.3470012

[9] Svetlana Minakova, Dolly Sapra, Todor Stefanov, and Andy D. Pimentel. 2022. Scenario Based Run-Time Switching for Adaptive CNN-Based Applications at the Edge. *ACM Transactions on Embedded Computing Systems* 21 (2 2022). Issue 2. https://doi.org/10.1145/3488718

[10] Christian Renaud. 2021. Special Report | The Edge-to-Cloud Continuum II About the Author.

[11] Wenhao Sun, Grace Li Zhang, Xunzhao Yin, Cheng Zhuo, Huaxi Gu, Bing Lil, and Ulf Schlichtmann. 2023. SteppingNet: A Stepping Neural Network with Incremental Accuracy Enhancement. In *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1–6. https://doi.org/10.23919/DATE56975.2023.10136943

[12] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2017. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. (9 2017). http://arxiv.org/abs/1709.01686

[13] Thanh Vu, Marc Eder, True Price, and Jan-Michael Frahm. 2020. Any-Width Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 3018–3026. https://doi.org/10.1109/CVPRW50498.2020.00360

[14] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. [n. d.]. Not All Images are Worth 16x16 Words: Dynamic Transformers for Efficient Image Recognition. https://github.com/

[15] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. [n. d.]. SLIMMABLE NEURAL NETWORKS. https://opensignal.com/reports/2015/08/android-fragmentation/

Appendix A FIRST APPENDIX