Hardware efficient Co-DETR model for mobile applications

Sophie Takken, University of Twente, The Netherlands

. ABSTRACT

Object detection models are evolving and their wide range of applications is expanding as well. Object detection models can be utilized as a tool for sustainability and there are various methods to aid climate change. Included in these benefits is the usage of mobile applications with object detection models. Mobile devices are resource constraint and have limited storage, therefore the object detection model should attain to these limits. However, object detection models can be complex and great in size. A state-of-the-art object detection model is Co-DETR, which is currently too large to deploy for mobile applications. This research will focus on the effects of global unstructured magnitude pruning on the hardware efficiency of the CO-DETR model for mobile applications. As object detection models can be compressed in order to decrease the model size and a method that can be applied is pruning. With pruning, redundant components of a network can be removed and there can be a trade-off between average precision and model compression. However, to the best of my knowledge, there is no research available on the application of model compression on the Co-DETR model. This research applies global unstructured pruning on the ResNet50 backbone of Co-DINO, focusing on pruning the weights of the convolutional layers. The impact on the average inference time, model size and average precision with a range of sparsity levels is evaluated, before re-training with a 20% sparsity level. The results showed a negative impact on the Average Precision (AP), average inference time and model size when applied, demonstrating an increase in average inference time and model size and a decrease in AP. This research aims to investigate the effects of pruning strategy on the Co-DETR model for mobile applications and broaden the knowledge of the effects of pruning on the average inference time, model size and average precision of the Co-DETR model.

Keywords: Co-DETR model, hardware efficiency, model compression, object detection, pruning.

1 INTRODUCTION

Object detection is a computer vision technique that can be utilized to detect instances of objects from one or multiple classes in an image [1]. There are various techniques to perform object detection and this can be observed in different object detection models. These different models can prioritize aspects such as inference speed or accuracy. The results in these areas are increasing as state-of-the-art models are researched and improved. As object detection models and techniques are evolving, so are the possibilities to utilize object detection. One of the ways in which it can be utilized is to aid sustainability.

Climate change is a growing issue and with this sustainability is becoming an increasingly more important aspect of artificial intelligence. An advancement of climate change is renewable energy and a demonstration of this is solar power. Solar power is an exemplary field that can benefit from the usage of object detection. Object detection can be used to detect the presence of solar panels from aerial photographs, which can help estimate the amount of solar energy generated in an area [31]. In this domain of object detection the model can be trained and evaluated specifically for detecting solar panels, however object detection models can be trained for a range of object classes. The MS COCO dataset is a popular dataset containing 91 object categories and containing 2.5 million labelled instances in 328 thousand images [16]. This dataset can be used as a benchmark for object detection, different object detection models tested with the same MS COCO dataset and the results are compared. The Co-DETR object detection model is tested with the highest AP score on the COCO test-dev with 66% AP [27]. This state-of-the-art object detection model is based on the DETR object detection model. DETR approaches object detection as a set prediction problem which is part of the more flexible end-to-end detector that signifies the DEtection TRansformer (DETR). This set prediction problem is a direct approach in predicting bounding boxes and category labels for each object of interest, where DETR predicts all objects simultaneously. To prevent duplicate predictions, a set-based global loss function is implemented which assigns a prediction to a ground truth object, with bipartite matching to ensure an one-to-one assignment [4]. Co-DETR advances this by using a collaborative hybrid assignments training scheme [27]. An enhancement Co-DETR makes to DETR is by training multiple parallel auxiliary heads in the encoder supervised by one-to-many label assignments, where a ground truth object can be matched to multiple predictions. As this is done during training and disregarded during inference, it does not add additional parameters to the model [27]. The amount of parameters used by Co-DETR on the COCO test-dev is 304 million. This makes the Co-DETR model large to deploy on resource constrained devices for instance for mobile applications. However, deployment on mobile applications can have various benefits, including benefits regarding sustainability, smartphone deployment could offer real-time application, more accessibility and ease of use. In order to reduce the size of the model, model compression techniques can be used as for instance pruning. Pruning focuses on removing inessential components of model [14]. Pruning can have advantages as reduction of model size and increased inference speed, but also downfalls as loss of accuracy [26]. However, to the best of my knowledge there is currently no available literature on pruning of the Co-DETR model and its effects. On the DETR model there is a limited amount of literature regarding pruning. The paper by Sun et al. [2023] discusses the application of sparse structured pruning on the DETR model and Roh et al. [2022]

introduce Sparse DETR, both papers include evaluation of the AP on the COCO 2017 val set. The aim of this research is to give insight into the affect of pruning on the hardware efficiency of the Co-DETR model in relation to mobile applications. Specifically, global unstructured pruning on the weights of the convolutional layers of the ResNet50 backbone is implemented. The Co-DETR model is trained with the MS COCO 2017 dataset and the results before and after pruning are evaluated based on model size, inference time and Average Precision (AP).

2 PROBLEM STATEMENT

Climate change is one of various fields in which object detection models can make a positive impact. There are various examples of research that has utilized object detection to help with climate change. In the research by Khalid et al. [2023] object detection was applied in detecting small pests in field crops. Field crops affect climate change with their impact on greenhouse gases, related through the amount of nitrogen fertilizer and crop growth [28]. By utilizing object detection in detecting pests, it can optimize production, economic costs and usage of pesticide sprays, which benefits climate change and sustainability in agriculture. Mobile application allows the tool to be real-time and more accessible for a large audience, such as farmers. Co-DETR is a state-of-the-art object detection model, nevertheless it is limited in usage for mobile applications. The Co-DETR model is rather large and to the best of my knowledge, there is no research available into compressing the Co-DETR model. For object detection models such as the Single Shot Detector (SSD), the lightweight SSD is introduced to make it suitable for mobile applications [7] and for the YOLOv2 model mobile applications for various purposes are researched [20]. This research aims to look into mobile application for Co-DETR, by applying pruning to the model and investigating the effect pruning has on the model's size, inference time and AP. ‘

2.1 Research question

“How can pruning strategy affect the hardware efficiency of the Co-DETR model for mobile application”

(1) “What is the effect of compression on the model size of the Co-DETR model? ”
(2) “What is the impact of pruning on the inference time of the Co-DETR model”
(3) “What is the impact of pruning on the mean average precision of the Co-DETR model”

3 RELATED WORK

There is literature available on different aspects of the research question on the effects of pruning strategy on the hardware efficiency of the Co-DETR model for mobile applications. The structure behind the object detection model is relevant as well as literature on pruning and testing methods. In the following section information on the background and relevant literature will be given.

3.1 Background

Object detection is a computer vision technique that can be used to identify and locate instances in images. There are different classes to be detected in an image, object detection models construct object classes based on training examples [1]. There are different methods as to how to identify objects. The research paper by Amjoud et al. [2023] discusses that traditional object detection models have three common attributes, firstly evaluating the entire image in steps and generation candidate boxes. Secondly, feature extraction analyses the candidate boxes, paying attention to features or patterns. Lastly, all these features are classified [2]. The traditional models require handcrafted features, but this can become redundant with the usage of Deep learning. Deep convolutional neural networks can be used for feature extraction and have more promising results as the feature extraction is more representative. The CNN surpasses handcrafted-features in discriminability and generalization of the features [25]. The CNN is an end-to-end learning model in which parameters can be trained and an important feature of the CNN is its sharing weights over layers. Convolutional neural networks generally consist of a convolutional layer, pooling layer and fully-connected layer [2]. A standard CNN backbone is ResNet, which can come in different variants containing specific amounts of layers. The ResNet backbone differs itself through residual learning, which increases the depth of the network while maintaining accuracy. It implements shortcut connections, which allow the model to skip one or more layers [9]. The architecture of ResNet contains multiple convolutional and batch normalization layers, additionally there is a fully connected layer that connects to the final layer [2]. The ResNet backbone can be utilized in various object detection models inclusive of the DETR model. The DETR model deviates from the traditional architecture, it consists of three main components, a CNN backbone followed by an encoder-decoder transformer and a simple feed forward network [4]. The key feature in the DETR model is the approach on object detection, it views object detection as a set prediction problem, which is reinforced in a set-based global loss function. The DETR model aims to reduce the need for prior knowledge in the form of, for example, initial guessing with proposals [4]. The DETR model uses bipartite matching to uniquely match predicted to ground-truth objects directly. The DETR model generates an activation map in the CNN backbone which can be fed into the transformer encoder, where a feature map is constructed. The decoder proceeds with the encoder output and decodes the objects in parallel. Each output embedding of the decoder is then decoded by the feedforward network, locating and classifying the outputs. The Co-DETR model extends the DETR model with collaborative hybrid assignments training [27]. The Co-DETR model implements a training scheme that aids the training efficiency and effectiveness of discriminative feature learning in the encoder and training of positive samples in the decoder. An important aspect is the usage of one-to-many label assignment during training. The original one-to-one set matching in DETR assigned unique matching pairs of predicted and ground-truth objects, whereas many-to-one label assignments can match a prediction to multiple ground-truth objects [27]. This one-to-many label assignment method is implemented through auxiliary heads in the encoder which are only used during the training period. The collaborative hybrid assignments training scheme entails the implementation of the one-to-many label assignments through multiple auxiliary heads and customized positive queries during the training of the model. This collaborative hybrid assignments training scheme can be applied to a range of variant DETR models. One of these DETR models is DINO which abbreviates Detr with Improved deNoising anchOr box. DINO improves DETR by using three core methods, contrastive denoising training, mixed query selection and look forward twice scheme [23]. DINO focuses on enhancing query selection during training by implementing the contrastive denoising training. Contrastive denoising training rejects irrelevant anchors, furthering the prevention of duplicate predictions and selection of anchors that lie far apart from the ground truth box. In addition, DINO improves the positional information for feature selection and advances the effectiveness and efficiency of the training process. Out of the DETR variants DINO yields the highest score on the MS COCO test-dev dataset with the ResNet50 backbone, it achieves 49.4 AP in 12 epochs [23]. When testing with MS COCO AP is the most important metric, this reflects the average precision. The COCO test-dev benchmark does not differentiate between the mean average precision (mAP) and average precision (AP)[30]. The AP is averaged over multiple Intersection over Union (IoU) values ranging from 0.5 to 0.95 in 0.05 increases. IoU values represent a metric used for comparing the localization of the predicted and ground-truth object. The MS COCO dataset is a commonly used dataset to train and test object detection models. The latest version is the MS COCO 2017 dataset, this set includes 118k training images and 5k validation images in which a variety of object instances can be found. Each image can contain multiple object instances and the MS COCO dataset prioritizes clear localization of each instance [16]. The MS COCO dataset is also used to evaluate the Co-DETR model, the Co-DETR model scored 66.0% AP with the ViT-L backbone. The Co-DETR set a new record with this AP on the MS COCO test-dev and reported 304M encoder parameters for the process [27]. In order to reduce the model size, different compression techniques can be utilized and a common technique is pruning. Pruning is the method of removing parameters from a network. Pruning can be applied with different amounts of sparsity, which refers to the amount of parameters pruned [17]. Different amounts of sparsity can affect a trade-off between accuracy of the model, however there are a number of distinct pruning methods which induce varying results. There is a distinction between structured and unstructured pruning. Unstructured pruning eliminates individual parameters while retaining the model structure whereas structured pruning removes groups as channels which impact the models structure. The pruning methods can be applied in different locations in the network, pruning can be applied locally or globally. Local pruning removes a certain sparsity of each subset of the group to be pruned, for example dividing the weights per layer and taking a sparsity of each layer, while global pruning removes the defined sparsity from all available locations in the network [5]. In addition, pruning can happen during different parts of the object detection process, there are pruning methods that can be applied before, during and post training the object detection model. Global unstructured magnitude pruning is a pruning method that can be applied after training. Magnitude based pruning is a pruning method that is based on the theory that smaller weights are less important, therefore magnitude based training removes weights ascending from the lowest to highest weights [15]. The weights will be made redundant by setting them to zero, the amount of weights is dependent on the sparsity level. After applying the pruning, it is common to fine tune the pruned model, this improves accuracy of the pruned model by re-training the pruned model with a subset of the dataset. As Li et al. [2023] discusses, unstructured pruning on the network's weights can greatly reduce the amount of model parameters, but when setting the weights to zero, the weights are not removed. Therefore the pruning does not fully utilize the hardware efficiency optimization by keeping the storage of redundant weights unchanged [14]. Further compression can be executed to optimize hardware efficiency. Sparse storage methods aim to optimize storage of sparse matrices, with indexing indicating whether an element in the matrix is non-zero. Hereby removing the need to store the value of the zero weights as the non-zero weights, optimally saving space. In order to implement sparse storage, a format such as the COO format can be used. The COO format is a more general format that stores the value, the index of the row and the index of the column of each non-zero weight [6]. However, this indexing can induce overhead that can counteract the compression [26]. The measure in which the overhead impacts the compression effectiveness of the Co-DETR model will have to be determined, but generally the COO format provides flexible and simple sparse storage.

3.2 Literature review

There is various literature on object detection models, mobile application of object detection models and pruning methods. Regarding the Co-DETR model, the main research paper that will be used is the one by Zong et al. [2022]. An extensive amount of literary research has been done that involves object detection. For example Xiao et al. [2020] gives a general review of object detection algorithms with deep learning methods, also mentioning DETR and Co-DETR, as well as Jain et al. [2014], which analyses the evolution of object detection methods. This information on different object detection models can be used when looking at mobile application of object detection by different models than Co-DETR. In the literature by Sun et al. [2019] a mobile application for food detection using YOLOv2 and DCNN based on Mobilenet is presented and Ng [11] discussed the mobile application for specific plant disease detection. This research paper will be looking into compressing and pruning the Co-DETR model and literature regarding compressing and pruning models is available. Han [20] discusses compression for deep neural networks using pruning, quantization and Huffman coding. Where Zhu et al. [2017] demonstrates research on the relation between model size and accuracy in pruned deep neural networks and concludes an positive outtake on model pruning for compression. In addition, there is more literature regarding pruning using more specific examples of object detection models. Zhang et al. [2021] displays the affects of a channel pruning pipeline implemented on YOLO v3/v5. The literature by Li et al. [2021] introduces a compression pipeline for one-stage detectors, in which pruning, knowledge distillation and quantization were combined to reduce inference time and model size with minimal mAP reduction. The object detection model DETR forms a basis for the Co-DETR model and on the DETR model there exists literature regarding pruning. After the application of sparse structured pruning on the network structure of the DETR model by Sun et al. [2023], the paper concluded an improvement in inference speed and reduction in computational cost. Roh et al. [2022] introduced Sparse DETR which introduces an encoder token scarification method with learnable sparsity, this method prioritizes favourable tokens for training with the ability to prune encoder tokens based on results in the decoder. The results of the Sparse-DETR and the pruning DETR are depicted in Table 3. Even though there is not an exceeding amount of research involving Co-DETR and to my knowledge no literature regarding the application of pruning strategy on the Co-DETR model, these related works approach similar topics.

Method	Epochs	AP	Params
DETR	500	42.0	41M
Sparse-DETR	50	46.3	41M
Pruning DETR	50	42.7	26M

Table 3. Results on COCO 2017 val by Sun et al. [2023] on DETR models [4, 18, 19].

4 METHODS OF RESEARCH

When researching how pruning strategy can affect hardware efficiency of the Co-DETR model for mobile application, three main methods of research were used, research into literature, by experiment and evaluation. In this research three different models are evaluated, an Co-DETR model referred to as the original model, this model pruned and this model pruned and trained. The research was conducted in the following order,

- training the original model
- pruning the original model
- fine tuning the pruned model
- testing the pruned and original model
- training the pruned model
- testing the trained pruned model
- evaluating the results.

Where fine tuning and training utilize the same technique, distinguished by the smaller subset of images that is used for fine tuning. Once the original model was trained, research into the application of pruning strategy was executed. There are various pruning strategies available and the objective was to choose a pruning strategy that is suitable with the Co-DETR model and is rather simplistic, due to time constraints on the research. The unstructured global magnitude pruning is applied to the original model, this can be applied to the model with different levels of sparsity and on various layers of the model. Testing was used to evaluate the effects of different sparsity values when pruning the model. Models for the different sparsity's were computed from the original model and these were fine tuned with a subset of the MS COCO 2017 dataset. The various pruned models had to be evaluated on the metrics of inference time, model size and AP. Therefore additional research for testing these metrics was conducted and implemented. From there the pruned and fine tuned models were evaluated for the sparsity amount that yielded on average the smallest model size, lowest inference time and highest AP. This sparsity amount was used to prune the original model which is then trained on the MS COCO 2017 dataset. This trained pruned model alongside the original model and pruned model are tested for inference time, model size and AP. These results can then be analysed to see how the results can be interpreted in relation to the research question. How the results of the original model in contrast to the pruned and trained pruned model performed on the basis of the inference time, model size and AP and what possible conclusions could be drawn from this.

5 TESTING

5.1 Environment

The research is executed locally on an available laptop, in this case the Lenovo Thinkpad P1 Gen3. This laptop consists of an NVIDIA Quadro T100 Max-Q GPU and up to 64GB of RAM. The model training and testing runs over the GPU and in a python environment using python 3.8. A virtual environment is set up with pytorch 1.12 with CUDA version 1.17 and mmcv-full==1.7.0. The training of the model however is extensive and is unable to run on the Lenovo Thinkpad P1 Gen3, consequently Google Colab is used to train the models. The fine tuning of the pruned models is able to run on the T4 GPU with 12.7 GB RAM via Google Colab. The training more extensively with the MS COCO 2017 dataset requires additional storage and the L4 GPU in Google Colab is able to provide this to a certain limit. Because of the limitations in hardware and time, the training is done in 3 epochs. For testing and training of the models the files provided with the Co-DETR code by Zong et al. [2022] are utilized. These files are based on MMDetection which provides a toolbox for object detection. In addition, pytorch is implemented, this is for instance utilized in applying the pruning method and sparse formatting. The pruning method used is unstructured global magnitude based pruning, this is done after training the Co-DETR model and before fine tuning. There was a series of Co-DETR variants that could be used for testing and training and Co-DINO with a ResNet50 backbone was used. The pruning was applied to the convolutional layers of the ResNet50 backbone. The pruning is applied on the weights of the convolutional layer and is applied globally and unstructured. Fig 2. illustrates unstructured pruning of weights in the convolutional layer. All the convolutional layers in the backbone were pruned with different amounts of sparsity's. In order to compare the different sparsity levels, 5 sparsity's were chosen, accounting for resource constraints. The pruning sparsity's tested are 10%, 20%, 30%, 40% and 60%. The sparsity amounts were chosen with increments of 10%, starting from the baseline, which is the original model with 0% pruning. Additionally, the last increment is made with 20% to the sparsity of 60%, in order to consider a higher sparsity level. After applying the pruning method, a sparse formatting method was applied to all the convolutional layers. This created tensors formatted in the COO format, which only stores the indexes and values of the non-zero elements. This was then exported into a checkpoint file which along with the configuration file is used for fine tuning of the models. Taking into account the resource constraints, a subset of random images from the MS COCO dataset 2017 was selected for fine tuning, 595 testing images and 405 validation images. After evaluation of the fine tuned models, the model with sparsity 20% was selected and trained more extensively with the MS COCO dataset. The configuration and checkpoint file after training were exported to be used for evaluation.

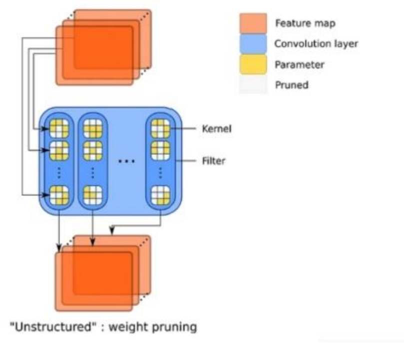

Fig. 2. Unstructured pruning of the weights in the convolutional layer, adapted from Tessier [2021].

5.2 Evaluation methods

In order to capture the effects of the pruning strategy on the Co-DETR model various testing was done to compute results on different metrics. Except for the test on sparsity level, when referencing testing on the models, the following three models are meant; the original Co-DINO model that has been trained, this original model that has been pruned and this original model that has been pruned and trained. Each model consists of a configuration and checkpoint file. The configuration file contains information on the model structure and configurations for testing and training whereas the checkpoint file is aimed toward storing data during training, as weights and biases.

5.2.1 Sparsity levels

To determine which sparsity level to use to prune the original Co-DETR model, testing was done on multiple levels of sparsity. For the model 10%, 20%, 30%, 40% and 60% of the weights of the convolutional layers pruned and the same tests were executed for each sparsity. For each pruned model, the average inference time was computed. This was done by computing the inference time for an image 50 times and averaging these times to determine the inference time for the image. The inference time was computed for 15 random images from the MS COCO val2017 dataset. In total the inference time is calculated over 750 times, 50 times per image and the average inference time per model is computed based on 15 images. In addition, the AP is tested for each model, this is done with the testing function included by Zong et al. [2022] with Co-DETR. Following this, to each model sparse formatting is applied by using the COO format executed by pytorch. Additionally, each model is compressed into a zip file and the file size alongside the amount of non-sparse parameters is computed. The results of the three metrics, inference time, model size and AP, are normalized. For the model size and inference time a low score is optimal so the normalization is computed for the values of each model by

$$normalized\ value = (max - value)/(max - min) \qquad (1)$$

In this function (1) the 'max' is the maximum and 'min' the minimum obtained value among the 5 models. The normalised value of 1 will represent the smallest model size or lowest inference time. Similarly, for the AP the highest score is preferred so the normalization is calculated with

$$normalize\ value = (value - min)/(max - min) \qquad (2)$$

In this function (2) the normalised value of 1 depicts the highest AP amongst the models with different sparsity levels. All three metrics are considered equally important for the evaluation.

5.2.2 Model size

The model checkpoint files are compressed into a .zip file and the file size is observed. For each model the amount of parameters are computed as well as the amount of parameters in the convolutional layers that have weight zero.

5.2.3 Inference per image

15 random images have been selected from the MS COCO val2017 dataset. For each of these images the inference time is computed

50 times consecutively. Then the average time of these 50 times is computed for each image, resulting in 15 inference times per model. The average inference time of a model represents the average of the inference times over these 15 images.

5.2.4 Average Precision (AP)

By using the testing function provided with the Co-DETR code by Zong et al. [2022] the Average Precision (AP) is computed. This correlates with the evaluation metrics of the MS COCO dataset. Where the AP is calculated with the average AP for different IoU thresholds considering a maximum of the 100-top detections [30]. The AP considers all object sizes, which is based on the amount of area in the image that is occupied by the object.

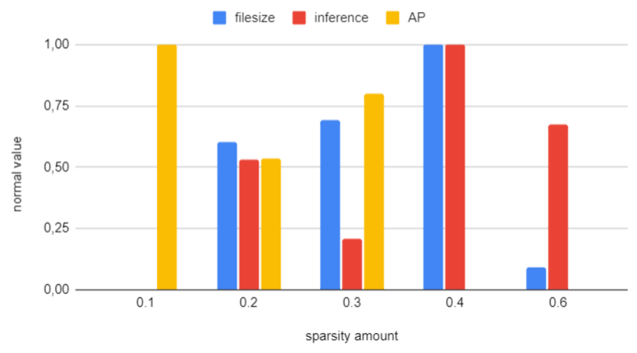

Fig. 1. Normal values of pruned and fine tuned models differentiating in applied sparsity levels.

7 RESULTS

7.1 Sparsity levels

For the sparsity levels, the file size of the compressed checkpoint file, average precision (AP) and average inference time is given in table 1. The differences in results for file size and average inference time between the sparsity levels are minimal, the average deviation between the file sizes is 2,6 MB and between the average inference time is 0,8 milliseconds. There is a visible decrease in AP when the sparsity level increases, however the average deviation of the AP is 0.0056. In fig 1. the normal values for the Average Precision (AP), average inference time and file size of compressed checkpoint file are given. The normal value of 0.1 on the file size and average inference time is 0, this represents the largest file size and average inference time amongst the models with different sparsity levels. Likewise, the normal value for the AP of 0.6 is 0, illustrating the lowest AP. In fig 1. it is visible that the 20% sparsity level yielded the most balanced scores, it is not the highest nor lowest performing in any category, where as 0.4 for example has the fastest inference time but lowest AP. The 20% sparsity level was chosen for further testing.

7.2 Model size, inference time and average precision (AP)

The results from the original Co-DINO model, the pruned Co-DINO model and the trained pruned Co-DINO model on the metrics average inference time, Average Precision (AP) and model size are displayed in table 2. When pruning the convolutional layers of the ResNet50 backbone 20% of the weights were set to zero. This makes that 4137203 of the 65204005 parameters are pruned, which leads to 6,37%. Table 2 displays the AP of the original Co-DINO significantly dropped with pruning.

8 DISCUSSION

8.1 Research question

When interpreting the results of the original Co-DINO model, the pruned model and the trained pruned model, it is observed that pruning in combination with sparse file storage can affect the model size. However, in this research pruning has increased the file size, inference time and decreased the AP.

The results of two different pruning strategies applied on the DETR model are given in Table 3.. Both the pruning strategy implemented in Sparse-DETR and the pruning strategy implemented on DETR by Sun et al [2023] show a decrease in inference time and model size while not observing a significant decrease in AP. This diverges from the results found within this research on the application of pruning strategy on the Co-DETR model, where the model size and inference time increased and the AP decreased. Considering the Co-DETR extends the DETR model, this could indicate that the Co-DETR model might benefit from other pruning strategies.

Nevertheless, the increased file size, inference time and decreased AP could be the cause of multiple factors. A possible cause could be the pruning method, the unstructured global magnitude pruning method is focused solely on the convolutional layers of the ResNet50 backbone. The convolutional layers of the model take up 31,73% of all parameters, whereas pruning with a sparsity of 20% takes up 6,37% of all parameters. This could possible relate to the file size not decreasing in combination with the sparse formatting which can produce overhead. The COO format that is used for sparse formatting, stores both the index of the row and column of the non-zero weight. Even though the COO format is a simple, general and widely-used data format, it can have a negative effect on the memory footprint [6]. Therefore it is possible that a less general and more specific storage format, for example focusing on memory efficiency, can be found more suitable with the pruning of the Co-DETR model. Nonetheless, this would have to be researched.

The sparse formatting method also influences the performance of the CO-DETR model, specifically the operations performed involving sparce matrices [6]. From the results displayed in Table 1. and Table 2. it can be observed that the average inference times are close to remaining constant. The average deviation of the average inference times for the various amounts of sparsity is 0,8 milliseconds, hence, the deviation is perceived to be minor. The cause of this deviation could be due to the models architecture and the efficiency of performing operations with the pruned weights.

When evaluating the results of models with various sparsity amounts on model size, average inference time and AP, it is visible that the file size slightly decreases with the increment in sparsity up to 60% sparsity. At 60% sparsity, the file size seems to climb again and it reaches a similar file size as with 10% sparsity. The cause of this spike is currently unclear, it could be related to the increase in sparsity affecting the amount of overhead relating to the storage of pruned weights. Even so, to determine the cause of the spike in file size, closer inspection into the change in storage between the sparsity amounts will have to be conducted.

In addition, the training could be more extensive, whereas fine tuning happened with a small subset of MS COCO val2017 and there were 3 training epochs executed. It is possible the losses through pruning are not compensated during the fine tuning or training, resulting in a lower AP. Overall, the results show an negative impact on the hardware efficiency of the Co-DETR model.

Sparsity	0.0	0.1	0.2	0.3	0.4	0.6
File size (MB)	649,61 32402	674,77 92425	670,03 20673	669,35 44054	666,91 77456	674,05 36537
AP	0,547	0,021	0,014	0,018	0,006	0,006
Average inference time (s)	0,6906 36007	0,6616 430953	0,6603 747698	0,6611 429084	0,6592 593047	0,6600 406612

Table 1. results of different sparsity levels after pruning and fine tuning.

Model	Zip file size mb	Average inference time	AP
Original	649,6132402	0,690636007	0,547
Pruned	670,0320673	0,6945705763	0,014
Trained pruned	667,651516	0,6986917664	0,011

Table 2. results during different stages of pruning the model.

8.2 Limitations and future work

With the research into the effects of pruning strategy for the Co-DETR model on hardware efficiency for mobile applications there were resource and time constraints. The resource restraint impacted the ability to test and train the models. The research on Co-DETR model by Zong mainly used upwards from 12 epochs for testing and training [27], whereas this research used 3 epochs. In the research by Azadvatan et al [2024] on the real-time deep learning algorithm for object detection MelNet, it displays an improvement in AP when the amount of training epochs is increased. Indicating that training with additional epochs might be beneficial to explore in future work.

Furthermore, the fine tuning uses a rather small subset of the original MS COCO 2017 dataset in which the images were randomly selected, this could be improved in terms of general optimization, investigating optimal images and dataset size. In addition, the research could be improved by more extensive testing, for example to explore a wider variety of pruning strategies, as well as sparsity levels and compression methods. The research by Tian et al. [2024] combines both structure and unstructured pruning with quantization. Unstructured pruning was applied as it generally has a higher compression rate than structured pruning. However, the random pruning of the individual parameters with unstructured pruning may lead to worse hardware acceleration compared to structured pruning [10]. As structured and unstructured pruning both have different strengths, it can be beneficial to evaluate structured pruning on the Co-DETR model in the future. Alternative compression methods as quantization could be applied, with which the hardware efficiency could possibly benefit. Currently 32 bit storage is used, whereas quantization can reduce the bit storage. Using quantization to reduce the bit storage to 8 bits is commonly used and research by Han et al. [2015] shows that quantization in combination with pruning induces significant model compression.

Overall, the compression strategy can be adapted in various aspects, as investigating structured pruning and different sparse formats for future work. Even though, this research provided inside into the impact of pruning strategy on the inference time, model size and average precision of the Co-DETR model, there is opportunity to further explore this.

9 CONCLUSION

In conclusion, when applying unstructured global magnitude based pruning on the backbone of the Co-DETR model, it causes an increase in file size, inference time and a decrease in average precision. Therefore based on the results of this research, exploration into different pruning strategy or pruning pipeline is recommended for compression of the Co-DETR model for mobile application.

10 REFERENCES

1. Yali Amit, Pedro Felzenszwalb, and Ross Girshick. 2021. Object Detection. Computer Vision (2021), 875–883. https://doi.org/10.1007/978-3-030-63416-2_660
2. Ayoub Benali Amjoud and Mustapha Amrouch. 2023. Object Detection Using Deep Learning, CNNs and Vision Transformers: A Review. IEEE Access 11, (2023), 35479–35516. https://doi.org/10.1109/ACCESS.2023.3266093
3. Yashar Azadvatan and Murat Kurt. 2024. MelNet: A Real-Time Deep Learning Algorithm for Object Detection. (January 2024). Retrieved June 23, 2024 from https://arxiv.org/abs/2401.17972v1
4. Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12346 LNCS, (May 2020), 213–229. https://doi.org/10.1007/978-3-030-58452-8_13
5. Hongrong Cheng, Zhang Miao, and Javen Qinfeng Shi. A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. Retrieved June 23, 2024 from https://ar5iv.labs.arxiv.org/html/2308.06767
6. Salvatore Filippone, Valeria Cardellini, Davide Barbieri, and Alessandro Fanfarillo. 2017. Sparse Matrix-Vector Multiplication on GPGPUs. ACM Transactions on Mathematical Software (TOMS) 43, 4 (January 2017). https://doi.org/10.1145/3017994
7. Shi Guo, Yang Liu, Yong Ni, and Wei Ni. 2021. Lightweight SSD: Real-time lightweight single shot detector for mobile devices. VISIGRAPP 2021 - Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications 5, (2021), 25–35. https://doi.org/10.5220/0010188000250035
8. Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings (October 2015). Retrieved June 23, 2024 from https://arxiv.org/abs/1510.00149v5
9. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-December, (December 2015), 770–778. https://doi.org/10.1109/CVPR.2016.90
10. Yang He and Lingao Xiao. 2023. Structured Pruning for Deep Convolutional Neural Networks: A survey. IEEE Trans Pattern Anal Mach Intell 46, 5 (March 2023), 2900–2919. https://doi.org/10.1109/TPAMI.2023.3334614
11. Shreya Jain, Samta Gajbhiye, Achala Jain, Shrikant Tiwari, and Kanchan Naithani. 2024. A Quarter Century Journey: Evolution of Object Detection Methods. 2024 4th International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies, ICAECT 2024 (2024). https://doi.org/10.1109/ICAECT60202.2024.10469334
12. Saim Khalid, Hadi Mohsen Oqaibi, Muhammad Aqib, and Yaser Hafeez. 2023. Small Pests Detection in Field Crops Using Deep Learning Object Detection. Sustainability 2023, Vol. 15, Page 6815 15, 8 (April 2023), 6815. https://doi.org/10.3390/SU15086815
13. Zhishan Li, Yiran Sun, Guanzhong Tian, Lei Xie, Yong Liu, Hongye Su, and Yifan He. 2021. A compression pipeline for one-stage object detection model. J Real Time Image Process 18, 6 (December 2021), 1949–1962. https://doi.org/10.1007/S11554-020-01053-Z/TABLES/8
14. Zhuo Li, Hengyi Li, and Lin Meng. 2023. Model Compression for Deep Neural Networks: A Survey. Computers 2023, Vol. 12, Page 60 12, 3 (March 2023), 60. https://doi.org/10.3390/COMPUTERS12030060
15. Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. 2021. Pruning and Quantization for Deep Neural Network Acceleration: A Survey. Neurocomputing 461, (January 2021), 370–403. https://doi.org/10.1016/j.neucom.2021.07.045
16. Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8693 LNCS, PART 5 (May 2014), 740–755. https://doi.org/10.1007/978-3-319-10602-1_48
17. Gabryel Mason-Williams and Fredrik Dahlqvist. What Makes a Good Prune? Maximal Unstructured Pruning for Maximal Cosine Similarity. Retrieved June 23, 2024 from https://github.com/gmw99/what
18. Byungseok Roh, Jae Woong Shin, Wuhyun Shin, and Saehoon Kim. 2021. Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity. ICLR 2022 - 10th International Conference on Learning Representations (November 2021). Retrieved June 30, 2024 from https://arxiv.org/abs/2111.14330v2
19. Huaiyuan Sun, Shuili Zhang, Xve Tian, and Yuanyuan Zou. 2024. Pruning DETR: efficient end-to-end object detection with sparse structured pruning. Signal Image Video Process 18, 1 (February 2024), 129–135. https://doi.org/10.1007/S11760-023-02719-4/TABLES/2
20. Jianing Sun, Katarzyna Radecka, and Zeljko Zilic. 2019. FoodTracker: A Real-time Food Detection Mobile Application by Deep Convolutional Neural Networks. (September 2019). Retrieved June 23, 2024 from https://arxiv.org/abs/1909.05994v2

21. Danhe Tian, Shinichi Yamagiwa, and Koichi Wada. 2024. Heuristic Compression Method for CNN Model applying Quantization to a Combination of Structured and Unstructured Pruning Techniques. IEEE Access (2024). https://doi.org/10.1109/ACCESS.2024.3399541
22. Youzi Xiao, Zhiqiang Tian, Jiachen Yu, Yinshu Zhang, Shuai Liu, Shaoyi Du, and Xuguang Lan. 2020. A review of object detection based on deep learning. Multimed Tools Appl 79, 33–34 (September 2020), 23729–23791. https://doi.org/10.1007/S11042-020-08976-6/FIGURES/10
23. Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. 2022. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. (March 2022). Retrieved June 23, 2024 from https://arxiv.org/abs/2203.03605v4
24. Jiacheng Zhang, Pingyu Wang, Zhicheng Zhao, and Fei Su. 2021. Pruned-YOLO: Learning Efficient Object Detector Using Model Pruning. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12894 LNCS, (2021), 34–45. https://doi.org/10.1007/978-3-030-86380-7_4/TABLES/2
25. Wang Zhiqiang and Liu Jun. 2017. A review of object detection based on convolutional neural network. Chinese Control Conference, CCC (September 2017), 11104–11109. https://doi.org/10.23919/CHICC.2017.8029130
26. Michael H. Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. 6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings (October 2017). Retrieved June 23, 2024 from https://arxiv.org/abs/1710.01878v2
27. Zhuofan Zong, Guanglu Song, and Yu Liu. 2022. DETRs with Collaborative Hybrid Assignments Training. Proceedings of the IEEE International Conference on Computer Vision (November 2022), 6725–6735. https://doi.org/10.1109/ICCV51070.2023.00621
28. Field Crop Agriculture and Climate Change - MSU Extension. Retrieved June 30, 2024 from https://www.canr.msu.edu/resources/field_crop_agriculture_and_climate_change_e3149
29. Neural Network Pruning 101. All you need to know not to get lost | by Hugo Tessier | Towards Data Science. Retrieved June 30, 2024 from https://towardsdatascience.com/neural-network-pruning-101-af816aaea61
30. COCO - Common Objects in Context. Retrieved June 23, 2024 from https://cocodataset.org/#detection-eval
31. Deep learning for solar panel detection | CBS. Retrieved June 23, 2024 from https://www.cbs.nl/en-gb/about-us/innovation/project/deep-learning-for-solar-panel-detection