2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT) | 978-1-6654-5262-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCCNT54827.2022.9984572

Optimization of multi-objectives for Adaptive Reverse Recommendation method on edge devices

Winston Dcosta
School of Computer Science and Engineering
KLE Technological University
Hubballi, India
dcostawinston6@gmail.com

Meena S M
School of Computer Science and Engineering
KLE Technological University
Hubballi, India
msm@kletech.ac.in

Sunil V Gurlahosur
School of Computer Science and Engineering
KLE Technological University
Hubballi, India
svgurlahosur@kletech.ac.in

Uday Kulkarni
School of Computer Science and Engineering
KLE Technological University
Hubballi, India
uday_kulkarni@kletech.ac.in

***Abstract*—The Neural Architecture Search (NAS) enables Deep Learning engineers to automatically explore the wide variety of all possible neural architectures to design the solution for a given problem. Though the NAS methods automatically generate the best possible architecture, it is time-consuming, and resource constraint since the search space consists of the best architectures(good accuracy) and includes non performing architectures as well. Researchers have proposed various NAS methods especially based on evolutionary algorithms which consists of diverse search space due to the presence of genetic operations leading to expensive search time cost. These NAS methods also deal with multiple objectives like FLOPS, latency and power usage, and model accuracy, for image classification tasks. Though the generated architectures from NAS have multi-objective, they require significant computing resources and are not suitable for running on edge devices with limited computational resources. The proposed method includes a novel model optimization method designed for the neural architecture generated from NAS methods to make it smaller in size, memory efficient, and faster at inference with minimal loss of accuracy. We optimized the models designed for standard datasets and decreased the parameters by 7.82% for CIFAR-10, 2.95% for CIFAR-100, 6.36% for STL-10, and 3.57% for Food-101 datasets. The resulting models have achieved an accuracy of 97.4% for CIFAR-10, 86% for CIFAR-100, 96.7% for STL-10, and 87.4% for Food-101 datasets.**

***Index Terms*—Deep Learning, Neural Architecture Search, Multi-objective optimization, Pruning**

I. Introduction

Deep learning (DL) [1] has completely transformed the future of Artificial Intelligence (AI). It has addressed several complicated challenges such as pattern recognition, image classification, segmentation and so on, that have plagued the AI community for many years. In past few years, multiple deep architectures with different learning frameworks have been rapidly initiated in order to build machines that can perform correspondingly to or better than humans in a variety of disciplines of application such as diagnosing diseases, self-driving automobiles, natural language and image processing. Additionally, the deep learning model needs additional data to produce accurate findings as the architecture expands. Moreover, because of the complexity of the data models, training deep learning models is an exceedingly expensive endeavour. They can demand expensive GPU [2] and hundreds of computers, which raises the cost for consumers. However, neural networks are difficult for non-experts to utilise since their success is heavily dependent on the appropriate settings of a broad number of hyper parameters and design choices.

Automated Machine Learning(AutoML) [3] strives to deliver effective off-the-shelf learning solutions that liberate professionals and non-experts alike from the time-consuming work of arriving at the appropriate algorithm for a given dataset, as well as provides suitable methods for preprocessing the data and optimizing the model hyper parameters. Transfer learning [4], feature extraction, feature selection, NAS (model selection, hyper-parameter optimization [5], and meta learning), and inferencing are all examples of Auto ML. Although human specialists build the majority of popular and successful model architectures, this does not mean we have thoroughly explored the network architectural space and selected the optimal alternative.

Neural Architecture Search(NAS) [6] [7] is the task of automatically discovering one or more designs for a neural network that would generate models with good results relatively quickly. NAS is defined as a system with three primary components(search space, search strategy and evaluation strategy) which is a clear and succinct description about NAS. The NAS search space provides a collection of operations (for example, convolution, fully-connected, and pooling) and how they may be combined to construct viable network designs. Human knowledge, as well as unavoidable human biases, are generally involved in the creation of search space. A NAS search strategy selects a sample of network architecture candidates from a population. Based on the child model performance characteristics, it produces high-performance architecture candidates (e.g. high precision, response time). There are several search strategies but the prominent among them are Reinforcement Learning(RL) [8], gradient-based optimisation and evolutionary algorithms [9]. The evaluation method evaluates or predicts the performance of several suggested child models in order to provide information for the search algorithm's development. Candidate evaluation may be a costly operation, and numerous novel ways such as parameter sharing, one- shot approach [10] and so on, have been developed to reduce time or computation resources. The combination of search space, search strategy and evaluation strategy has resulted in different NAS methods

such as ENAS [11], DARTS [12] and NSGA-Net [13]. While most existing research, such as ENAS, DARTS and others aims to identify designs that improve prediction accuracy, these structures may be complicated and hence unsuitable for deployment in specific computer environments. Whereas, when searching for neural network topologies, the NSGA-Net [14] [15] [16] [17] approach takes into account prediction accuracy as well as other pertinent objectives (e.g., power consumption, FLOPs, latency). Multi-objective optimization aids in the solution of several interesting issues in a single run.

Multi-Objective Optimization based on Adaptive Reverse Recommendation (MoARR) [18] is a technique that aids in the creation of lightweight architectures that are more adaptable and diverse. Furthermore, a unique multi-objective strategy is developed to effectively evaluate Historical Evaluation Information (HEI) in order to rapidly search for designs with high accuracy and a limited number of parameters.The contributions of the proposed work are:

1) We apply the Soft Filter Pruning (SFP) [19] strategy to the architectures before analysing them using the multi-objective method to generate better HEI and obtain efficient pareto optimum architectures.
2) We compare different datasets such as CIFAR-10, CIFAR-100, STL-10 and Food-101 and analyse how MoARR method performs after applying SFP technique to it.

The paper is divided into the sections listed below. Section 2 focuses on MoARR [18] algorithm. In section 3, we discuss about the methodology used to optimize the MoARR method. Section 4 focuses on the results obtained after optimizing the MoARR method on different datasets.

II. Background Study

To implement the proposed methodology a background study had to be made on the below mentioned topics.

A. Multi-Objective Optimization

Multi-objective optimization [20] is a technique for solving multi-objective optimization problems (MOPs), which have many objective functions that must all be maximised at the same time. As a result, a MOP may be defined as,

$$\begin{aligned} \min_{x} F(x) = (f_1(x), f_2(x), ..., f_m(x))^T, \\ s.t \;\; x = (x_1, x_2, ..., x_n)^T \end{aligned} \tag{1}$$

where n is the dimension of the decision variable and m signifies the number of objective functions. In general, the MOP objectives are in odds with one another. As a result, only if the following requirements are met for solutions x_a and x_b:

$$\begin{aligned} \forall_i = 1, 2, ..., m \;\; f_i(x_a \leq f_i(x_b), \\ \exists_i = 1, 2, ..., m \;\; f_j(x_a \leq f_j(x_b), \end{aligned} \tag{2}$$

Then the option x_a is preferable than option x_b. It's known as x_a dominating x_b, and it's always written as $x_a > x_b$. Furthermore, in decision space, the Pareto optimum solution [21] is the solution that cannot be influenced by any other option. The Pareto optimum set is made up of all Pareto solutions, whereas the Pareto front is made up of all Pareto ideal objective vectors that correspond to the Pareto optimal set. Obtaining complete Pareto optimum solutions is typically quite challenging. As a result, in multi-objective optimization, an evenly distributed Pareto front must be created, which can roughly represent the whole solution set. The non-dominated sorting genetic algorithm (NSGA-II) [22], multi-objective particle swarm optimization (MOPSO) [23], and multi-objective evolutionary algorithm based on decomposition (MOEA/D) [24] are just a few examples of multi-objective algorithms that use population optimization. In continuous optimization tasks, MOPSO and MOEA/D are more efficient. Rapid non-dominated sorting and crowded-distance-based selection are two of the NSGA-main II's technologies, which generate Pareto solutions and uniformly distribute them. In conclusion, multi-objective optimization is an effective strategy for solving neural network optimization issues, especially when neural network pruning [25] is well-done.

B. Pruning

Deep Neural Networks (DNNs) have made significant advances in a variety of fields, including speech recognition [26], computer vision [27], natural language processing, and so on. However, because of the enormous set of parameters in DNNs, model deployment might be expensive at times. Pruning is one strategy for compressing DNNs and reducing the amount of processing that has been proposed to alleviate this problem. Pruning has evolved into a highly effective approach for compressing and accelerating current neural networks. Traditional network pruning techniques have a significant influence on network compression while retaining accuracy. Their network topology is logically built, for example 30 percent of the filters are pruned in each layer, predicting the sparsity ratio [28], and regularisation.

Existing pruning methods are classified into two types: filter pruning (FP) and weight pruning (WP). WP is a quite well pruning approach that prunes specific weights inside the network [29] that have near-zero values, resulting in a minimal network without sacrificing performance of the model.However, because non-zero weight assignments are irregular and unexpected, we require an extra trace of the weight placement, and the minimal network pruned by WP cannot be shown in an organised fashion like FP due to the network's unpredictability, making WP incompatible with general-purpose processors.

FP-based approaches [30] [31], on the other hand, trim filters or channels inside the convolution layers, resulting in a pruned network that is still highly structured and can be readily accelerated on general processors. There are multiple phases in a typical filter pruning pipeline. At the start we need to train the larger model with many parameters till convergence which leads to a better accuracy. Secondly, prune the unimportant and redundant filters. Finally, fine-tune the trimmed network. It is

usually observed that training the pruned model with different initial conditions can also produce satisfactory results. Thus, rather than training weights, it is the network design that matters.

C. Multi-Objective Optimization based on Adaptive Reverse Recommendation(MoARR)

NAS [6] has enabled humans to explore a diverse range of structures [32] [33] that would not have been conceivable otherwise, resulting in a vast search space [34]. Because of the inclusion of genetic processes, most NAS approaches, particularly those based on evolutionary algorithms, have a very vast search space, resulting in a high search cost time. It is not true that NAS only looks for the optimal set of architectures; it can also look for architectures that are not ideal since they have a chance of being found. To save searching costs, most NAS algorithms employ a set outer network level structure and only scan the repeating cell structure. When there are enough cells and channels, this type of fixed design functions well. However, as the design grows more lightweight, performance suffers considerably. More flexible and diverse neural architectures are required to achieve better lightweight designs, and more efficient approaches for bigger search spaces should be created.

As a result, there is a need for an algorithm that attempts to avoid searching for the worst architectures, resulting in a compact and optimum search space and increasing the likelihood of obtaining the most optimal architecture when compared to earlier NAS approaches. As a result, an algorithm known as Mutli-Objective Optimization based on Adaptive Reverse recommendation (MoARR) [18] assists in avoiding the worst architectures by exploiting prior evaluation information to reduce search cost. Based on the historical evaluation information the algorithm tries to avoid worst architectures leading to cost reduction with increased optimization. It studies the potential relationship between parameter quantity and accuracy to select the suitable model with the help of Reverse Recommendation model (RRModel). Thus, pareto optimal models are selected by RRModel when higher accuracy and smaller parameters are inputted to the RRModel. The approach follows four steps, firstly, defining an objective, secondly, defining a new search space, thirdly introducing the multi-objective optimization task to obtain light weight CNNs and lastly, designing acceleration strategy to reduce the computational cost as shown in Figure 1. The objective is defined in such a way that it searches the architectures with the best tradeoff between accuracy and number of parameters.

$$\underset{x \in S}{max} F(x) = [ACC(x), -PAR(x)] \tag{3}$$

s.t. Par(x)$\leq P_{max}$ where S refers to the CNN architecture codes in our new search space, ACC(x) refers to accuracy scores, PAR(x) refers to the number of parameters, P_{max} refers to the upper limit of parameter count.

The new search space focuses on structural flexibility which adjusts the number of cells and channels in each stage and also focuses on cell diversity which generates different structures each different stages. The search space generates an architecture consisting of 5 stages. Stage 1 concentrates on extracting the common level features. Stage 2 to 4 reduces the dimension of the input image using a stride equal to 2. Stage 5 is used to make a final prediction using a global pooling layer and fully connected layer.

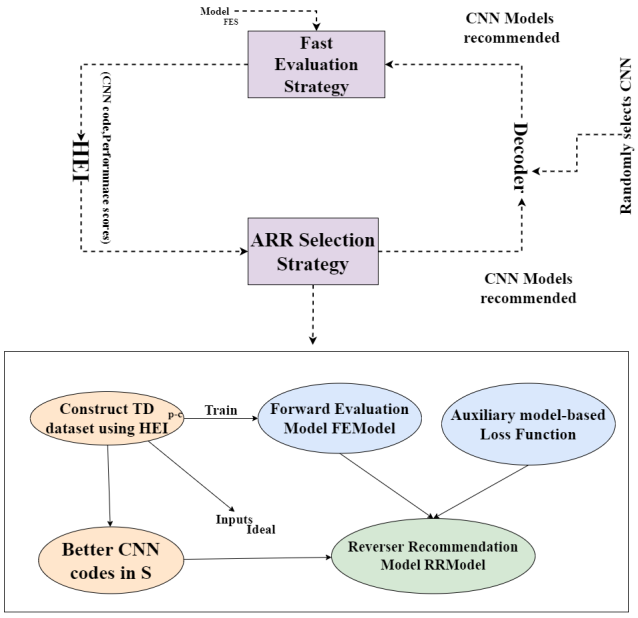

Fig. 1. Overall Framework of MoARR

Algorithm 1 MoARR

$\hat{S} \leftarrow$randomly select N codes from $\hat{S}$
$HEI \leftarrow \{< x, ACC(x), PAR(x) > | x \in \hat{S}\}$
$B(\hat{S}) \leftarrow \{x \in \hat{S} | \nexists y \in \hat{S}, x < y\}$
repeat
  #ARR selection strategy
  $TD_{c-p} \leftarrow \{< x, (acc, par) > | < x, acc, par > \in HEI\}$
  Train FEModel using TD_{c-p}
  Train RRMODel using loss function given in equation(2)
  Get $Inputs_{Ideal}$ using equation(3)
  TargetScores $\leftarrow$ randomly select N performance scores from $Inputs_{ideal}$
  SuperiorCodes$\leftarrow$ RRModel(TargetScores)
  # Update HEI,$\hat{S}$ and $\hat{S}$
  $HEI \leftarrow HEI \cup \{< x, ACC(x), PAR(x) > | x \in SuperiorCodes\}$
  $\hat{S} \leftarrow \hat{S} \cup SuperiorCodes$
  $B(\hat{S}) \leftarrow \{x \in \hat{S} | \nexists y \in \hat{S}, x < y\}$
until no change to B($\hat{S}$) for several iterations

The MoARR algorithm 1 consists of two main components i.e Adaptive Reverse Recommendation(ARR) and RRModel. AAR recommends the most suitable architecture code by

utilizing the historical information of $\hat{S}$. RRModel maps the performance pair (acc,par) to suitable architecture code through

$$x \in \{argmin_{x \in S}(|ACC(x) - acc|^2 + |PAR(x) - par|^2)\} \tag{4}$$

In order to make this RRModel effective Historical Evaluation Information(HEI) is used

$$(HEI) = \{< x, ACC(x), PAR(x) > | x \in S\} \tag{5}$$

HEI helps in constructing the Performance-to-code training data

$$TD_{p-c} = \{< (acc, par), x > | < x, acc, par > \in HEI\} \tag{6}$$

to build an effective RRModel which would output corresponding codes by considering the performance scores as input. Due to the presence of contradictory outputs having similar accuracy score and parameter count the RRModel becomes less effective. For the sake of removing this confusion one code is preserved for each performance,but it was observed that it resulted in information loss and difficulty in selection. Concerning these defects the RRModel made use of an auxiliary based loss function which utilizes HEI to study the suitable output values. The new loss function of RRModel is defined as follows

$$Loss(x, y') = \frac{1}{n} \times \sum_{i=1} ||x_i - FEModel(y_i^{'})||^2 \tag{7}$$

where FEM(Forward Evaluation Model) maps the architecture code to its performance pair, x=$\{x_1, \cdot \cdot \cdot \cdot, x_n\}$ refers to a set of target performance scores. The equation helps in determining the difference between the target performance and performance of architecture codes recommended by RRModel. The feedback provided by the FEModel is useful to the RRModel to improve its output.The HEI also constructs the code-to-performance training data which trains the FEModel. The new loss function not only helps in avoiding defects but also helps in building the RRModel automatically. Now the performance scores with higher accuracy and lower number of parameters are inputted to RRModel to find the pareto architectures not dominated by the evaluated codes of $\bar{S}$ which is provided by

$$Inputs_{Ideal} = \{(acc, par) | 0 < acc < 1, 0 < par < P_{max}, \\ \forall x \in B(\hat{S})\ ACC(x) \leq acc\ or\ PAR(x) \geq par\} \tag{8}$$

. In MoARR the evaluation of CNNs is time consuming due to the large training data and large number of epochs but Fast Evaluation Strategy(FES) estimates the final accuracy with less training data and number of epochs.

III. Proposed Approach

MoARR [18] algorithm utilizes the existing results and historical information to find accurate and lightweight architectures. It also defines a novel multi-objective method to search for accurate model with less parameters. MoARR models have demonstrated excellent performance that is comparable to or even beats the present state-of-the-art developed by domain experts in a variety of demanding tasks, suggesting tremendous potential in automating neural network construction. This cumbersome model's storage, memory, and compute requirements surpass the computational capabilities of today's mobile devices. A cutting-edge model with outstanding accuracy may not be appropriate, or even practicable, for deployment on certain computer devices, such as mobile phones. As a result, it is critical to keep these models compact, as they have a cheap computational cost yet great accuracy in real-world applications. MoARR method has helped us in finding the pareto optimal architectures but we can still reduce the FLOPS through pruning [35].

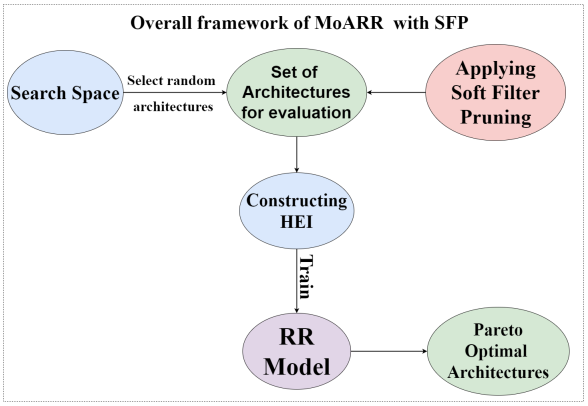

Fig. 2. Combination of MoARR and SFP

Most filter pruning techniques aim to eliminate filters from just one layer of a pretrained model before fine-tuning the model that was left behind to make up for the performance loss. They then trim the model's subsequent layer and adjust it once more till the last layer is clipped. However, once filters are pruned, these procedures will not update them again. As a result, the model capacity is severely decreased because of the deleted filters; and correspondingly such harsh pruning has a harmful impact on the performance of the compressed models. Dynamically removing the filters in a gradual manner is possible using the Soft Filter Pruning (SFP) [19] technique. During the training phase, it is intended to continuously update the pruned filters. It avoids the wasteful layer-by-layer pruning approach and allows practically all layers to be trimmed concurrently while maintaining the model capacity of the created CNN models at the same level as the early ones. More specifically, our technique may prune a model that is currently being trained or one that has previously been trained.

The MoARR approach employs a search space defined by S, which contains a set of all architectures. From S, a collection of architectures $\hat{S}$ is chosen for evaluation. We attempt to apply the SFP algorithm 2 to the architectures under consideration in order to acquire superior HEI, resulting in considerably lighter CNN models than the present MoARR method as shown in Figure 2. We utilise the l_p-norm to estimate the relevance of each filter as shown in eq(6) throughout the evaluation of

Algorithm 2 SFP Algorithm

Input: training data: D, pruning rate: X_i
the parameters of model $P = P^{(i)}, 0 \leq i \leq L$
Initialization of model parameter P
for $e = 1; e \leq e_{max}; e$ **do**
  the model parameters P are updated based on D
  for $i = 1; i \leq L; i++$ **do**
    l_2-norm for each filter is calculated $||F_{i,j}||_2, l \leq j \leq N_{i+1}$
    Zeroise $N_{i+1}X_i$ filters by l_2-norm filter selection
  end for
end for
The dense model with parameters P^* from P is obtained
Output: The dense model and its parameters P^*

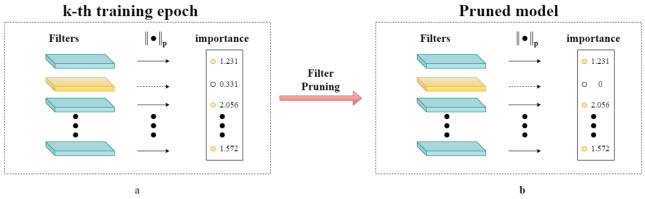

Fig. 3. Steps of soft filter pruning

various architectures. In general, the convolutional outputs of the filter with the smaller $l_p - norm$ result in lower activation levels and hence have a reduced numerical influence on the final estimate of deep CNN models. According to this concept, tiny $l_p - norm$ filters will be pruned first, followed by greater $l_p - norm$ filters. We utilise a pruning rate of X_i in particular to choose $N_{i+1}X_i$ irrelevant filters for the $i - th$ weighted layer.

$$||F_{i,j}|| = \sqrt[p]{\sum_{n=1}^{N_i} \sum_{k_1=1}^{K} \sum_{k2=1}^{K} |F_{i,j}(n, k_1, k_2)|^X} \tag{9}$$

The value of the specified $N_{i+1}X_i$ filters is then set to zero. This can reduce their contribution to network output briefly. Nonetheless, in the subsequent training step, we enable these selected filters to be changed in order to maintain the model's representational capacity and high performance. We train the network for one epoch after pruning to recreate the pruned filters. Back-propagation is used to update the pruned filters to non-zero values, as seen in Figure 3. SFP allows the trimmed model to train with the same capacity as the original model. SFP goes through the filter selection, filter trimming, and reconstruction phases iteratively. We can generate a sparse model with numerous zero filters once the model has converged. A single zero filter corresponds to a single feature map. The feature maps associated with these zero filters will never be zero throughout the inference technique. There will be no effect on removing these filters or the accompanying feature maps. We now have a better HEI than with the previous MoARR method, which is used by the ARR strategy to avoid picking bad architectures at the beginning of the search. A better HEI results in a higher performance score, which is employed by the RR model resulting in a superior pareto optimal models. The RR model picks the architecture which will have the closest performance score.

IV. Results and Discussion

Here we are comparing different datasets and analysing how MoARR method performs after applying SFP technique to it. Table I displays by what factor does the number of parameters reduce and whether it effects the accuracy as well after the application of SFP to MoARR method. For comparison we have considered standard datasets such as CIFAR-10 [36], CIFAR-100, STL-10 [37] and Food-101. We have trained the datasets for 600 epochs with and without applying pruning with a batch size of 96, learning rate of 0.025 and momentum 0f 0.9 on a single GPU. It determines that applying SFP to MoARR method helps in finding much more pareto-dominated architectures and selects the best among those architectures which consists of lesser parameters and also optimal accuracy as compared to the traditional approach.

TABLE I
Comparison of different datasets with and without Pruning techniques

Datasets	Validation Accuracy(%)		Parameters(Millions	
	without Pruning	with Pruning	without Pruning	with Pruning
CIFAR-10	97.4	97.16	2.3	2.12
CIFAR-100	86.0	83.05	3.8	3.18
STL-10	96.7	97.94	4.4	4.12
Food-101	87.4	89.2	3.3	3.182

After applying SFP on MoARR method for CIFAR-10 we have observed 7.82% decrease in the number of parameters and also there is an increase of 0.24% in accuracy. Coming to CIFAR-100 there is 16.31% decrease in the number of parameters but it also resulted in 2.95% decrease in the accuracy. Similarly we saw 6.36% decrease in the number of parameters for STL-10 dataset with an increase in the accuracy by 1.24% . Lastly coming to Food-101 dataset we saw a decrease in the number of parameters by 3.57% along with an increase in the accuracy by 1.8% From table I we can see that we have obtained pareto-dominated architectures i.e increase in the accuracy and decrease in the number of parameters for all the datasets except for CIFAR-100.

V. Conclusion

When we consider any architecture which has been discovered by the MoARR algorithm it needs to be pareto optimal so that it can be easily deployed on any edge device with lesser memory. In this paper we have applied SFP technique to the MoARR algorithm and analysed how it performs on several datasets like CIFAR-10, CIFAR-100, STL-10 and Food-101. It was observed that the parameters of the models after training on each dataset had reduced significantly after applying SFP and even there was no decrease in accuracy except for CIFAR-100. This method has aided in the preservation of model

capacity and resulted in higher performance of the resultant architectures. Surprisingly, SFP has assisted in reducing the parameters of these models to some amount when compared to the state-of-the-art models developed using MoARR.

In our proposed work, we have limited to optimise only two objectives namely accuracy and number of parameters. As a part of future scope, more than two objectives(FLOPs, latency, model size, MACs, etc) can be considered as a part of the neural architecture search strategy for obtaining pareto-optimal architectures.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, et al. Deep learning. nature, 521 (7553), 436-444. *Google Scholar Google Scholar Cross Ref Cross Ref*, 2015.

[2] Amr Kayid, Yasmeen Khaled, and M Elmahdy. Performance of cpus/gpus for deep learning workloads. *The German University in Cairo*, 2018.

[3] Esteban Real, Chen Liang, David So, and Quoc Le. Automl-zero: Evolving machine learning algorithms from scratch. In *International Conference on Machine Learning*, pages 8007–8019. PMLR, 2020.

[4] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[5] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

[6] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[7] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

[8] Keith G Mills, Fred X Han, Mohammad Salameh, Seyed Saeed Changiz Rezaei, Linglong Kong, Wei Lu, Shuo Lian, Shangling Jui, and Di Niu. L2nas: Learning to optimize neural architectures via continuous-action reinforcement learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1284–1293, 2021.

[9] Chao Pan and Xin Yao. Neural architecture search based on evolutionary algorithms with fitness approximation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[10] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559. PMLR, 2018.

[11] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

[12] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[13] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 419–427, 2019.

[14] Zhichao Lu, Ran Cheng, Shihua Huang, Haoming Zhang, Changxiao Qiu, and Fan Yang. Surrogate-assisted multi-objective neural architecture search for real-time semantic segmentation. *arXiv preprint arXiv:2208.06820*, 2022.

[15] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision*, pages 35–51. Springer, 2020.

[16] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Neural architecture transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[17] Andrea Falanti, Eugenio Lomurno, Stefano Samele, Danilo Ardagna, and Matteo Matteucci. Popnasv2: An efficient multi-objective neural architecture search technique. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[18] Chunnan Wang, Hongzhi Wang, Guosheng Feng, and Fei Geng. Multi-objective neural architecture search based on diverse structures and adaptive recommendation. *arXiv preprint arXiv:2007.02749*, 2020.

[19] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[20] Tao Wu, Jiao Shi, Deyun Zhou, Xiaolong Zheng, and Na Li. Evolutionary multi-objective one-shot filter pruning for designing lightweight convolutional neural network. *Sensors*, 21(17):5901, 2021.

[21] Eugenio Lomurno, Stefano Samele, Matteo Matteucci, and Danilo Ardagna. Pareto-optimal progressive neural architecture search. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1726–1734, 2021.

[22] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021.

[23] Jinhua Zheng, Zeyu Zhang, Juan Zou, Shengxiang Yang, Junwei Ou, and Yaru Hu. A dynamic multi-objective particle swarm optimization algorithm based on adversarial decomposition and neighborhood evolution. *Swarm and Evolutionary Computation*, 69:100987, 2022.

[24] Lu Chen, Bin Xin, and Jie Chen. Interactive multiobjective evolutionary algorithm based on decomposition and compression. *Science China Information Sciences*, 64(10):1–16, 2021.

[25] Tao Wu, Jiao Shi, Deyun Zhou, Yu Lei, and Maoguo Gong. A multi-objective particle swarm optimization for neural networks pruning. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 570–577. IEEE, 2019.

[26] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[27] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[28] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018.

[29] George Retsinas, Athena Elafrou, Georgios Goumas, and Petros Maragos. Weight pruning via adaptive sparsity loss. *arXiv preprint arXiv:2006.02768*, 2020.

[30] CH Sarvani, Mrinmoy Ghorai, Shiv Ram Dubey, and SH Shabbeer Basha. Hrel: Filter pruning based on high relevance between activation maps and class labels. *Neural Networks*, 147:186–197, 2022.

[31] Linsong Shao, Haorui Zuo, Jianlin Zhang, Zhiyong Xu, Jinzhen Yao, Zhixing Wang, and Hong Li. Filter pruning via measuring feature map information. *Sensors*, 21(19):6601, 2021.

[32] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

[33] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[34] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

[35] Tai Vu, Emily Wen, and Roy Nehoran. How not to give a flop: combining regularization and pruning for efficient inference. *arXiv preprint arXiv:2003.13593*, 2020.

[36] Rahul Chauhan, Kamal Kumar Ghanshala, and RC Joshi. Convolutional neural network (cnn) for image detection and recognition. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 278–282. IEEE, 2018.

[37] Hardik Singh, Sweta Swagatika, Raavi Sai Venkat, and Sanjay Saxena. Justification of stl-10 dataset using a competent cnn model trained on cifar-10. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1254–1257. IEEE, 2019.