Training-Free NAS for Visual Wake Words

Fabrizio Marsico, Kevin Elezi, MohammadSadegh Radmehr
Politecnico di Torino, Italy
{s319359, s316685, s301623 }@studenti.polito.it

Abstract

In recent years, the field of computer vision has made significant strides in advancing object detection and recognition algorithms, but the majority of existing approaches rely heavily on processing large amounts of data, which can be computationally intensive and inefficient for real-time applications To address new necessitates emerged by Internet of Things (IoT) applications, microcontrollers offer a cost-effective computing platform for deploying intelligent IoT applications that leverage machine learning at scale. However, microcontrollers suffer from severe limitations in terms of on-chip memory and computing capabilities, and so deploying computer vision on such devices requires tiny vision models that can operate within a few hundred kilobytes of memory and storage. This paper presents the study of Neural Architecture Search in the context of Tiny Machine Learning. The main goal was to be able to create neural networks that would be able to recognize whether or not there is the presence of a person in the image. The creation of these networks was to be done through automatic pattern generation, a process that is increasingly in vogue in Automated Machine Learning. To find the best architecture, we decided to use a layer-based approach, thus generating, through the simple Random Search algorithm, many different combinations. In order to remain consistent with the Tiny ML framework and thus with the lightness that micro-controllers require in terms of computations, we decided to choose these randomly generated models based on two training-free metrics: NASWOT[20] and ArchScore. We designed the latter metric by exploiting the characteristics and parameters of the network, and then, thanks to the Bayesian Optimization[11][12] algorithm, we were able to judge the efficiency of the model. The code is availabe at : `github.com /Sadegh-rad /NAS _for _TVWW`

1. Introduction

In recent years, convolutional neural networks (CNNs) have become instrumental in generating image representations for a variety of recognition tasks, such as object classification, scene recognition, and object detection. However, it's essential to recognize that not every model is universally applicable, as each specific task requires unique specifications for optimal performance. Deep learning advancements have significantly improved computer vision accuracy, enabling widespread adoption, especially for edge devices with limited computational capabilities, memory, and power constraints. These devices demand neural network architectures with low latency. Mobile platforms have recently seen progress in real-time inference for various vision tasks.

In this work, we utilize the Visual Wake Words dataset to encourage research and development of vision models suitable for deployment on resource-constrained microcontrollers. We focus on a lightweight binary classification task as a representative microcontroller vision use case: detecting if a person is present or not in video frame inputs. The dataset aims to serve as a benchmark for evaluating tiny vision models that can meet microcontrollers' strict constraints in terms of model size, latency, and energy efficiency.

We evaluate this tiny model using a CNN generated through a Neural Architecture Search (NAS) technique. NAS refers to the process of automatically discovering optimal neural network architectures for a given task. It typically involves exploring an extensive search space of possible architectures and evaluating their performance to identify the best ones. We challenge the most "basic" version of the NAS search methodology, Random Search, to demonstrate the power of CNNs. Unlike other NAS techniques that employ complex algorithms and heuristics, random search follows a simple approach. It randomly samples architectures from the search space and evaluates their performance using a predefined metric. This process is repeated for a predefined number of iterations or until a satisfactory architecture is found. The simplicity of the random search NAS technique offers several advantages, such as requiring minimal prior knowledge or domain expertise and being computationally inexpensive compared to other NAS techniques. This efficiency makes it particularly suitable for exploring large search spaces with limited computational resources. To determine the most efficient architecture among the generated combinations, we use two metrics for evaluation: NASWOT and ArchScore. Both can assess the efficiency of the randomly generated architecture through its structural characteristics, allowing us to extract the best networks without training the model. In the research field, these techniques are becoming increasingly popular, as the growing interest in Machine Learning and Deep Learning drives researchers to seek fast and automated methods.

2. Related Work

2.1. Models for microcontroller devices.

Some of the most significant CNNs for VWW[10] include: MobileNetV2[2], this is a lightweight CNN architecture that is optimized for mobile and embedded devices, it has fast inference speed and low memory footprint. Also it was specificaly used for developing VWW[10] MobileNetV3[3], the newest version with high accuaracy/parameters ratio; ResNet50[6][7][8], a complex model of Residual Networks, a family of deep CNN architectures that have been used extensively in computer vision tasks. Other networks, such as Inception, VGG, and DenseNet[9], have also been used in VWW[10] applications with varying degrees of success.

2.2. FreeRea

FreeREA[19] is a custom cell-based evolution Neural Architecture Search (NAS) algorithm. It aims to identify neural network architectures that maximize model accuracy while maintaining small size and computational constraints, particularly for deployment on resource-constrained devices. The researchers selected a combination of training-free metrics that accurately approximate the model's test accuracy, that were used to rank different network architectures during the search process, which was implemented using an evolution-based approach. They also conducted experiments on constrained settings, where the search space was limited due to target hardware constraints, and despite these limitations, FreeREA[19] still produced accurate models. Based on these promising results, future efforts will focus on generalizing the method to other search spaces and tackling more challenging tasks.

3. Visual Wake Words

The Visual Wake Words[10][15] is a dataset that classify images into two categories: whether or not there is at least one person present. This is particularly focused on enabling the development of models suitable for microcontrollers. To serve this purpose, the Visual Wake Words[10] is specifically designed for typical MCU use cases, which involve determining if a person is present in the camera's field of view, and the detection of a person in an image can be used to trigger an alert or collected for further analysis. With the decreasing cost of image sensors, they are increasingly used in IoT devices, spanning various applications from industrial to home automation. However, most image datasets available are not suitable for microcontroller use cases. For example, ImageNet contains a large number of classes that are not relevant for classical microcontroller applications, and it lacks samples specifically labeled as ”person.” The CIFAR-10 dataset, although designed for edge inference, has limitations due to its low image resolution (32x32), which restricts the capacity of models trained on it.

To address these issues, the Visual Wake Word dataset[10] is build utilizing a subset of the public COCO dataset[13], that consists of several images divided into training and validation subsets, where both subsets contain examples of images classified as either ”person” or ”not-person.” By leveraging this dataset, it enables the development and evaluation of models that are suitable for microcontroller-based applications.

3.1. Coco Dataset

The COCO dataset[13], short for Common Objects in Context, is a large-scale object detection, segmentation, and captioning dataset. It was created by Microsoft Research in collaboration with several universities and is widely used in computer vision research.

In 2017 version, it contains more than 330,000 images, each annotated with object bounding boxes, object segmentation masks, and object captions. The images in the dataset are diverse, depicting a wide range of object categories in various scenes and contexts, including people, animals, vehicles, household objects, and outdoor scenes.

COCO[13] has become a standard benchmark for object detection, segmentation, and captioning tasks in computer vision: it has been used to train and evaluate various deep learning models, including Faster R-CNN, Mask R-CNN, and various other models based on CNNs.

We opted for the 2014 version, with around 115k images, to increase data speed processing and reduce eventually over-training.

4. Methodology

4.1. Labels

First then first, we built new labels for COCO dataset[13] The process for of generating new labels the Visual Wake Words[10] dataset from the COCO dataset[13] follows these steps: each image is given a label, either 1 or 0, based on specific criteria; a label of 1 is assigned if the image contains at least one bounding box that corresponds to the object of interest, such as a person, and if the area of that

(a) An example of our train images, where labels represent 1 if a person is present, 0 if not.

bounding box is greater than 0.5% of the total image area. In the context of the person/not-person use case, a label of 1 indicates the presence of a 'person' in the image, while a label of 0 signifies that the image does not contain any objects classified as 'person'. To generate the new annotations and the files in PyTorch records format corresponding to this dataset, we used :

```
create_coco_train_minival_split.py
```

available in open-source at [15] :

```
github.com /Mxbonn /visualwakewords
```

For evaluation, we use the minival on the COCO2014[13] dataset comprising a total of 8k images.

4.2. Data Augmentation

To improve the model's generalization ability, robustness to various image conditions, and overall, performance in object detention tasks, we apply some Data Augmentation technique to our dataset.

1.`transforms.RandomHorizontalFlip(p=0.5)`: This transformation randomly flips the image horizontally. It is commonly used in object detection tasks to introduce variability in the orientation of objects. By randomly flipping the images,the model can learn to detect objects regardless of their left-right orientation.

2.`transforms.RandomRotation(degrees=10)`:This transformation randomly rotates the image by a certain degree. Since in the COCO dataset[13] many scenes of daily life are taken, we often end up with images from any angle, as it happens precisely in the daily routine of taking those photos. Therefore, this augmentation helps the model generalize better and improves its ability to detect objects from different perspectives.

3.`transforms.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1)`: Color jittering applies random perturbations to the brightness, contrast, saturation, and hue of the image. For our dataset, since photos cover any time and place, this transformation can help network learning.

4.`RandomResizedCrop(size=(96, 96), scale=(0.8, 1.0), ratio=(3/4, 4/3))`: The motivations behind this transformation relate to those mentioned above.

5.`transforms.ConvertImageDtype (torch.float16)`: This transformation converts the image data type to float16. Float16 is a lower precision data type compared to the default float32 used in deep learning models. It can reduce memory requirements and potentially speed up computations, especially when working with large datasets like COCO[13]. However, it is important to note that float16 may result in some loss of precision.

We perform these steps to enable the custom dataset class to efficiently load and process the data during training.

4.3. CustomCNN

We manually design a custom CNN model called CustomDNN, with a series of convolutional blocks followed by fully connected layers and their number is determined by the given hyperparameters. These layers collectively perform feature extraction and spatial downsampling, capturing and preserving relevant patterns in the input data, where each block in the model includes a 2D convolutional layer, batch normalization layer, rectified linear unit (ReLU) activation function, dropout layer, and max pooling layer.

Then we have several fully connected layers, where the number is specified by the hyperparameters: each ones is followed by a ReLU activation function, allowing the model to learn non-linear relationships between the extracted features, and includes dropout regularization to mitigate overfitting.

4.4. Search Algorithm

To search for the optimal model, we develop a random search approach, with training-free metrics: in this way we can build accurate models with a significantly lower search time then classic NAS pipelines where, in order to assess the performance, they had to train and evaluate each candidate architecture individually, which constitute the true bottleneck in NAS algorithms.

We define a search space,that checks a set of hyperparameters represented as a dictionary (params) against a series of conditions. These conditions define specific con-

	Search Space
number_of_filters (4,5,6)	[16,64],[32,128],[64,256]
Kernel_size	[3,5]
padding	[1,2]
dropout_rate	[0.0,0.1,0.2,0.3,0.4,0.5]
max_pooling_stide	[2,3]
Max_pooling_stide	[1,2]
Dropout_rate_fc	[0.0,0.1,0.2,0.3,0.4,0.5]
number_of_filters10	[32,64,128,265,512]
number_of_fc_layers	[1,2,3]
number_of_neurons_per_fc_layer	[16,32,64,128]

Table 1. This table show the distribution of values of our Search Space (SS)

straints or requirements for the hyperparameters of a neural network model. The code iterates through the layers of the network and checks various properties such as the number of filters, ecc. If any of the hyperparameters violate the defined conditions, the code appends a message indicating the violated condition to a list.

NASWOT

The metric proposed is Linear Regions (NASWOT)[20], which measures the expressivity power of an architecture and allows the search of competitive candidates in few seconds. NASWOT[20] is a metric that measures the redundancy in a neural network by counting the number of activations that are switched on together (i.e., the number of times two activations are both greater than zero) across the network. It quantifies this value with a score: a low NASWOT score indicates that there are many pairs of activations that tend to be active together, suggesting a higher level of redundancy in the network; a higher value, instead, indicate a more efficient use of activations.

ArchScore

The second metric we used was defined by ourselves: we named it the ArchScore. This method implements a Bayesian optimization[11][12] algorithm to find the optimal combination of weights for a scoring function that evaluates the performance of different neural network architectures. The objective is to find the architecture with the lowest score. Its function is to judge a network by metrics that include FLOPs (Floating Point Operations), number of parameters, number of fully connected layers, depth (number of layers), width (maximum number of channels in any layer), receptive field size, and number of convolutional layers. For each of these metrics, Bayesian Optimization[11[12]] will be used to decide which of these metrics will be more 'important' than the others.

metric generated best model	parameters	accuracy (25 epo)
NASWOT	4116557	76.4 %
ArcScore	444862	75 %

Table 2. Here we can observe the values reported by each best model, generated by our metrics

$$
\begin{aligned}
ArchScore &= A + B + C + D + E + F + G && (1)\\
A &= w_{flops} \cdot s_{flops} && (2)\\
B &= w_{params} \cdot s_{params} && (3)\\
C &= w_{fc_{layers}} \cdot s_{fc_{layers}} && (4)\\
D &= w_{depth} \cdot s_{depth} && (5)\\
E &= w_{width} \cdot s_{width} && (6)\\
F &= w_{receptivefield} \cdot s_{receptivefield} && (7)\\
G &= w_{convlayers} \cdot s_{convlayers} && (8)
\end{aligned}
$$

where $w_{flops}, w_{params}, w_{fc_{layers}}, w_{depth}, w_{width}, w_{receptivefield}, w_{convlayers} \subset [0, 1]$.

To make it more clear, if FLOPs and parameters are heavily weighted, smaller and more efficient models may be favored. On the other hand, higher weights on depth and width would favor models with more layers and channels, indicating increased capacity. Similarly, emphasizing the number of convolutional layers would favor models that perform more spatial processing.The score provides a weighted and normalized evaluation of crucial model metrics, offering insights into the suitability of different architectures based on the valued metrics. The weightings are optimized to best align with the evaluated architectures.

5. Experiment

In this section, we describe some key components of our experiment, which aimed to evaluate the most efficient

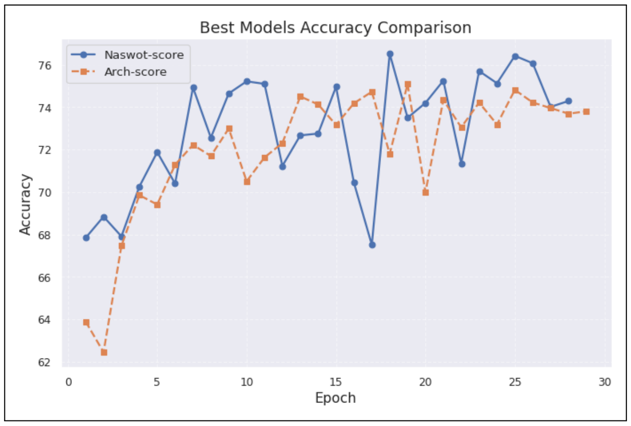

Figure 2. The plot shows a comparison in accuracy between the first best model found with NASWOT[20], and the one found with our metric, ArchScore, after several epoch iterations

model for VWW[10]. Our objective was to develop the model using a NAS (Neural Architecture Search) training-free algorithm and compare performance based on various metrics. All experiments were performed on a google colab, with standard Tesla T4 GPU.

We start the process by generating several models with a random, starting with an hand-made CNN, called CustomDnn. We test each model using different metrics, first with NASWOT[20], then with our metric ArchScore to find the best performing one. After finding the model that gets the highest score in each ones, we implement a train and we obtain our results.

5.1. Training Setup

To prevent overfitting we implemented Early Stopping technique in training process: it monitors the model's performance on a validation dataset and stops the training process once the model's performance stops improving or starts to degrade.

5.2. Accuracy

As we can observe from table 2, after 25 epochs, accuracy in ArchScore is 1.4 % less the NASWOT[20], but with one order of magnitude of parameters less, we can call it a good compromise. After that Early stopping approach interrupt epoch iterations to prevent over-fitting.

6. Conclusions

Based on our results, it is clear that training-free search is an effective approach for rapidly constructing efficient models tailored to specific tasks. Limited on-chip memory is becoming a major problem in deploying CNN models on microcontrollers, and to advance research in this area, we tested the effectiveness of training-free development on specific task, such as vision use case like VWW[10]. We report relevant benefit with respect to State of the Art training-based methods[14], with twenty minutes in the worst case in our search time, much lower than State of the Art algorithms[14]. In future research, there may be a focus on compressing more complex networks, such as MobileNetV3[3], to enable their deployment on resource-constrained devices while maintaining their essential characteristics.

References

[1] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, April 2017.*.

[2] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. *Mobilenetv2: Inverted residuals and linear bottlenecks.*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4510– 4520, 2018.

[3] Andrew Howard, Mark Sandler, Grace Chu, LiangChieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. arXiv *Searching for mobilenetv3 preprint arXiv:1905.02244, 2019.*.

[4] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, Chang Xu *GhostNet: More Features from Cheap Operations*.

[5] Mingxing Tan, Quoc V. Le *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition*.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Identity mappings in deep residual networks.*. In European conference on computer vision, pages 630–645. Springer, 2016

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep residual learning for image recognition.*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770– 778, 2016.

[9] Peng Zhaoa , Chen Lia,, Md Mamunur Rahamana, Hao Xua, Hechen Yanga, Hongzan Sund,Tao Jiangb and Marcin Grzegorzekc *A Comparative Study of Deep Learning Classification Methods on a Small Environmental Microorganism Image Dataset (EMDS-6): from Convolutional Neural Networks to Visual Transformers*. In Proceedings of the IEEE Conference

on Computer Vision and Pattern Recognition, pages 4510– 4520, 2018.

[10] Aakanksha Chowdhery, Pete Warden, Jonathon Shlens, Andrew Howard, Rocky Rhodes *Visual Wake Words Dataset*.

[11] Peter I. Frazier *A Tutorial on Bayesian Optimization*.

[12] Colin White Abacus.AI colin@abacus.ai Willie Neiswanger Stanford University and Petuum Inc. neiswanger@cs.stanford.edu Yash Savani Abacus.AI yash@abacus.ai

BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick *Microsoft coco: ´Common objects in context*. In European conference on computer vision, pages 740–755. Springer, 2014

[14] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. *Regularized evolution for image classifier architecture search.*. In Proceedings of the aaai conference on artificial intelligence, volume 33, pages 4780–4789, 2019.

[15] *Visual Wake Words Dataset*. https://github.com/Mxbonn/visualwakewords

[16] DMITRY ULYANOV *IMAGE GENERATION WITH CONVOLUTIONAL NEURAL NETWORKS*. Doctoral Thesis

[17] Supervisors Prof. Daniele Jahier Pagliari Dr. Matteo Risso Dr. Alessio Burrello Candidate Fabio Eterno *Differentiable Neural Architecture Search Algorithms for MLPerf Tiny benchmarks*. POLITECNICO DI TORINO, Master Degree Thesis, Anno accademico 2021-2022

[18] Sulabh Katiyar, Samir Kumar Borgohain *IComparative evaluation of CNN architectures for Image Caption Generation*. https://arxiv.org/abs/2102.11506

[19] Niccolo Cavagnero, Luca Robbiano, Barbara Caputo, Giuseppe Averta ‘ *FreeREA: Training-Free Evolution-based Architecture Search*. Politecnico di Torino, Italy niccolo.cavagnero, luca.robbiano, barbara.caputo, giuseppe.averta@polito.it

[20] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. *Neural architecture search without training.*. In International Conference on Machine Learning, pages 7588–7598. PMLR, 2021.