Draft – Paper accepted (preliminary) to the ECAI-2024 conference https://ecai.ro/ June 2024

VRES-CNN: A Tiny Convolutional Image Classifier with Versatile Choice of Hyperparameters

1st Radu Dogaru
Dept. of Applied Electronics and Information Engineering
National University of Science and Technology Politehnica Bucuresti
Bucharest, Romania
radu.dogaru@upb.ro

2nd Ioana Dogaru
Dept. of Applied Electronics and Information Engineering
National University of Science and Technology Politehnica Bucuresti
Bucharest, Romania
ioana.dogaru@upb.ro

***Abstract*— A novel lightweight convolutional neural network architecture is proposed, denotes as VRES-CNN. Unlike most of the actual lightweight architectures VRES-CNN provides a versatile set of hyper-parameters associated with variable number of macro-blocks. Moreover, the macro-block definition is rather simple and allows very good accuracies by emulating nonlinear convolutions. The use of residual connections, dropout, batch normalization and separable convolutions ensure very good accuracies with a relatively small complexity of the model. With a proper choice of the hyper-parameters, it is demonstrated that models with less than 100 kilo parameters and near state-of-the-art accuracy can be identified for a wide range of datasets (with low or high input image resolutions). In terms of accuracy performance, it is shown that VRES-CNN performs in the state-of-the-art range and much better than a recently proposed lightweight architecture (EtinyNet) for the same or smaller model size.**

Keywords—convolutional neural network, TinyML, hyper-parameter search, image processing, lightweight architecture

I. INTRODUCTION

In the growing context of IoT and EdgeAI a diversity of tiny neural models were recently proposed [1]. Given the resources-constrained specificity of the targeted computational platforms, such models are challenged to get competitive accuracies while minimizing at most the occupied memory for both parameters and activations. In the TinyML [2][3] context, many MCU platforms (e.g. ESP32-CAM, Arduino Nano 33 BLE Sense, etc.) require neural models with less than 1Mbyte of occupied memory (usually Flash) for parameters and less than 0.5 Mbyte memory (usually RAM) for activations.

Although there are already well established architectures included in major machine learning frameworks (for instance EfficientNet [4], MobileNet [5] etc.) many other compact solutions are recently proposed [6-9] aiming to squeeze the most from the available memory. Quite interestingly, the conceptual design of these networks shares some common aspects: They are built as a stack of several (macro)-layers each defined by some predetermined parameters. For instance, the recently proposed EtinyNet model [7] is composed of 2 LB blocks followed by 2 DLB blocks. However, there is no versatility in choosing the number of (macro)-layers nor in the specific numbers per each macro-block. In most cases, the performance and comparison are given for the widely known ImageNet dataset (using high resolution images) but they lack tuning and performance evaluation for other lower-resolution image datasets that are often employed in practical applications (noting that Tiny-ML deployment environments often make use of low-speed microcontrollers and thus their computations are restricted to low resolution images).

Herein we focus on a different conceptual design of the architecture, where we also use macro-blocks (here denoted as macro-layers) but with a simpler definition exploiting the idea of a non-linear convolution, recently proved to increase performance [10][11]. By allowing a versatile set of hyper-parameters the model can be easily tuned for various datasets i.e. by varying the number of macro-blocks in relation to image sizes and the filter profile of the macro-layers to optimize accuracy performance for the given dataset. The goal is to achieve maximal accuracy with a minimal memory cost. The result is called VRES-CNN (Versatile RESidual CNN) architecture, expanding an idea initially proposed in [12] with the additional of residual connections and using separable convolutional layers instead standard convolutional ones.

The VRES-CNN is evaluated on a variety of image recognition problems demonstrating that using a proper choice of the versatile set of hyperparameters, tiny optimal models with very good accuracies and less than 100k-parameters can be identified using heuristics choices for hyperparameters derived from preliminary extensive random NAS. The key ingredient of our model is the existence of macro-layers with emulated nonlinear convolution (proved already in [12] to dramatically improve the accuracy when compared to standard linear convolution) combined with optimally tuned dropout and batch normalization in each macro-layer, thus resulting in a very good balance between training and validation accuracies. Moreover, given a versatile choice of hyper-parameters (number of filters and nonlinear degrees on macro-layers) one can easily adjust the model to any specific dataset in order to get out a well-balanced model with almost overlapping plots of the validation and training accuracies during training. The remaining sections are summarized as follows:

Section II provides a detailed description of the VRES-CNN model (code will be freely provided as an addition to the git-hub project in [12] at the time of publication). Section III provides hints to help rapidly locate a choice of the hyper-parameters for a given dataset while Sections IV and V are devoted to results and comparisons considering representative datasets. For the low-resolution datasets considered in Section IV some comparisons with the modified EtinyNet [7] model was considered (it is now possible to choose 0,1 or 2 DLB blocks, suggesting that VRES-CNN outperforms it and that such tiny models as [6-9] may also benefit from defining and

tuning a versatile set of hyper-parameters for them. Concluding remarks are given in Section VI.

II. The VRES-CNN Model

A. The Macro-layer

Similarly, to [12] the main building block of the VRES-CNN is called a macro-layer. Its structure is depicted in Fig.1. It consists in a permanent layer (traditional convolution followed by batch-normalization and dropout) stacked after NL additional layers composed of convolutions followed by nonlinear (here ReLU) activation. With NL>0 one can emulate the nonlinear convolution [10][11] already proved to be a powerful concept in image processing [13]. Although here we consider emulated convolution, in the future it can be directly implemented as a simplicial layer [14] already proved as quite convenient for hardware implementation with economy of resources as suggested in [15]. Optimal compromise size-accuracy is achieved for 3x3 convolution kernel size, 4x4 pooling size, and 0.6 dropout coefficient. The ReLU function is used in activation layers and MaxPooling acts as nonlinearity in the permanent layer; Each macro-layer down-samples by a factor of 2 the images resulted from convolutional filters.

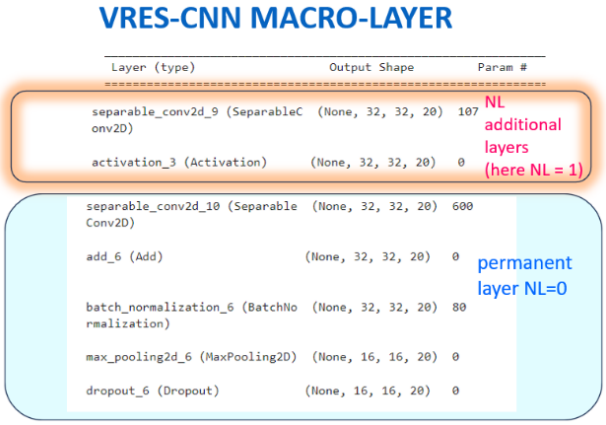

Fig. 1. The structure of the macro-layer emulating a non-linear convolution.

Unlike in [12] VRES-CNN uses depth-wise separable convolution (implemented using the SeparableConv2D Keras layer), giving significant (on average 7 times) size reduction and skip (residual) connections similarly to ResNet (residual networks) topologies. As indicated in [12] the addition of Batch-Normalization and Dropout layers in addition with proper choice for the *drop* parameter are essential for getting good accuracies. The remaining (versatile) set of hyperparameters are:

Fil=[fil1, fil2,…filM] indicating a list of convolutional filters size for each of the 1 to M macro-layers;

NL=[nl1, nl2,..nlM] indicating a list of nonlinear degree for each of the 1 to M macro-layers;

Flat= 0 or 1 – indicating that Global-Average-Pooling (flat=0) or Flatten (flat=1) layers are used to transform the collection of images resulting from the chain of convolutional macro-layers in vectorial input to the output classifier.

hid = [h1,h2..,hH] - indicating a list of up to H hidden dense layers in the output classifier. If the list is void (hid=[]) there is no additional dense layer except the output one. In fact, this choice gives the lightest structure and extensive simulations proved that adding hidden layers for the VRES-CNN gives only dramatic size increases and no substantial performance gain. Consequently, in the next we consider hid=[] (no additional hidden layers in the output classifier).

B. VRES-CNN Model Examples

A first example is considered in Fig. 2 only for the purpose of identifying the relevant building blocks of VRES-CNN.

This particular example considers the (difficult) CIFAR-100 problem and is defined by the simplistic set of hyperparameters: fil= [20, 40]; NL= [1, 0]; flat= 0; hid= [100].

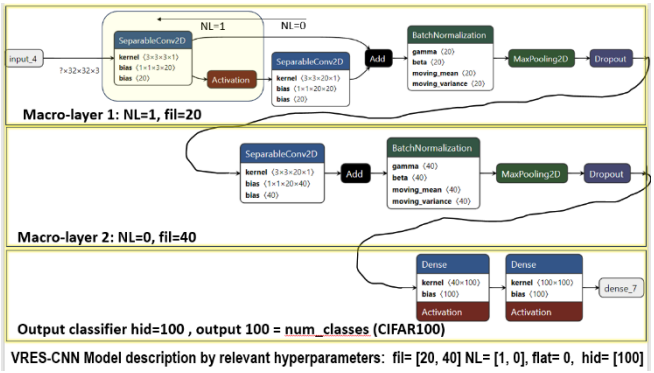

Fig. 2. A simple example of a VRES-CNN model; macro-layers are emphasized. The first macro-layer emulates nonlinear convolution with NL=1 degree while the second macro-layer is a conventional convolution layer (NL=0). The output classifier has one hidden dense layer with 100 units.

It results in a tiny 16k-pars (kilo-parameters) model but the validation accuracy for the indicated dataset is of only 9% with some clear overfit as seen from its training curves in Fig.3. In this particular case there was no optimization for the number of mcaro-layers and for the filter profile. As detailed in section III, using proper heuristics derived after extensive random NAS allows the user to get an optimal-size model. For the CIFAR-100 problem such a model is the one depicted in Fig.4 with 3 macro-layers. The training curves depicted in Fig.5 indicate that the model is a well balanced one, avoiding either overfit and underfit. The model size is still resonable, of only 59 k-pars. When saved as a .tflite model it occupies 136 kbytes. For this model a validation accuracy of 61% is obtained. For comparison, best V-CNN model with a comparable size (65-kpars) in [12] achieved only 42.46% accuracy.

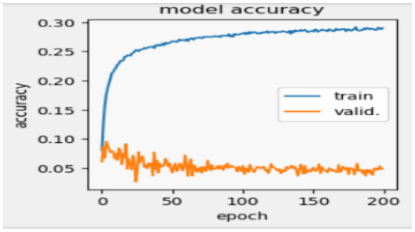

Fig. 3. Training and validation accuracies during the first 200 epochs.

Experiments with the same dataset and other light architectures often indicate that in many cases, with no versatile set of hyperparameters, such architectures are not well tailored to optimal-size balanced models.

For comparison with conventional architectures, Fig. 6 displays training curves for a light ResNet11 (11 deep layers) model with 322 k-pars (about 5 times more than in the previous VRES-CNN model) and although the model is well

balanced the validation accuracy is significantly smaller (50.44%) than for the VRES-CNN.

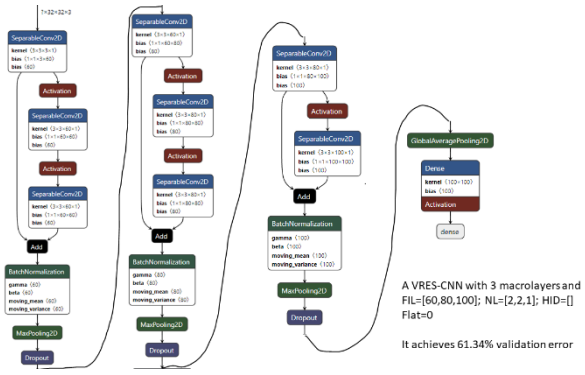

Fig. 4. An optimal VRES-CNN model with 3 macrolayers - capable of achieving 61.34% accuracy on the validation set (CIFAR100)

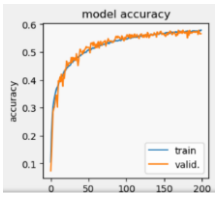

Fig. 5. Training and validation accuracies during the first 200 epochs for the case of a well-balanced VRES-CNN model using CIFAR100 dataset (61.34% best validation accuracy with 59 k-pars) .

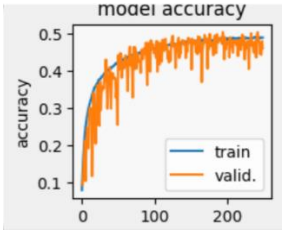

Fig. 6. Training and validation accuracies for the ResNet11 model (50.44% validation accuracy with 322 k-pars)

For the same dataset, a light CNN model, namely the EtinyNet [7] (here with multiplication factor 0.5) resulting in a model size of 150 k-parameters was considered. As seen in Fig. 7 in this case some overfitting occurred, and the validation performance is quite low (19.18%). It is the consequence of a lack of versatility in choosing a tailored EtinyNet model (best performance for it was reported only in the case of ImageNet dataset).

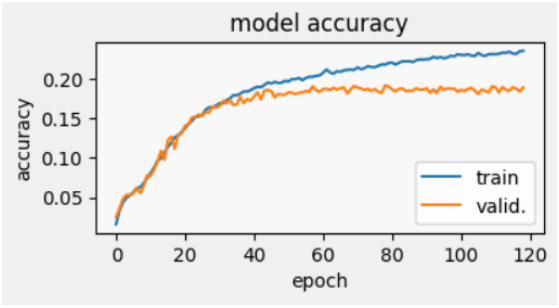

Fig. 7. Training and validation accuracies for the EtinyNet (0.5 multiplier) model (19.18% validation accuracy with 150 k-pars)

III. Versatile Hyperparameters, Their Choice And Aspects Regarding The Training Process

One important advantage of VRES-CNN over other architectures is the versatility in choosing the hyperparameters. It allows a relatively large search space allowing the localization of an *optimally sized model* i.e. one which achieves the best compromise between size and accuracy. Such a large space however comes with a cost, namely the need of some method to rapidly locate the best architecture. Traditionally this is the subject of neural architecture search (NAS) [16].

A. Random NAS

In [12] a detailed investigation for the effect of each hyperparameter was performed and some heuristics were derived, to be detailed next. While such heuristics allow fast settings of the *M* parameter (the number of macro-layers) and *NL* (the nonlinearity profile), it turns out that the choice of the *FIL* profile for optimally sized and well-balanced models has very much to do with the specific dataset. Consequently, some random search within the filter profile helps to identify the best solution. In [17] a random-search NAS for the V-CNN model was performed. Similar searches were done for VRES-CNN in the case of SVHN dataset but also in the case of other datasets. Fig. 8 indicates the validation accuracy versus model size for a selection of best models among all 100 considered. They were searched for the filter profile (herein a filter profile with values ranging from 10 to 150) and training stopped after 10 training epochs. The choice for nonlinearity profile was NL=[3,2,1], and flat=0. As expected, the models identified as efficient after 10 epoch-search improved their accuracies after additional training within 200 more epochs.

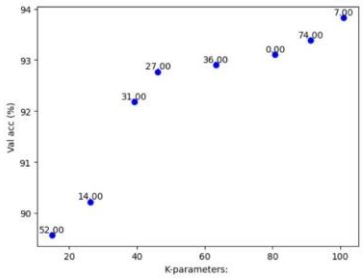

Fig. 8. Validation accuracies for a selection of best models resulting after a random search NAS with 100 model trials. The SVHN dataset was used. IDs of the models are provided in each case.

The filter profile for each of the selected models (depicted with different colors and symbols for each particular ID) is shown in Fig.9 along with the specific profile in each case and the 10-epochs validation accuracy.

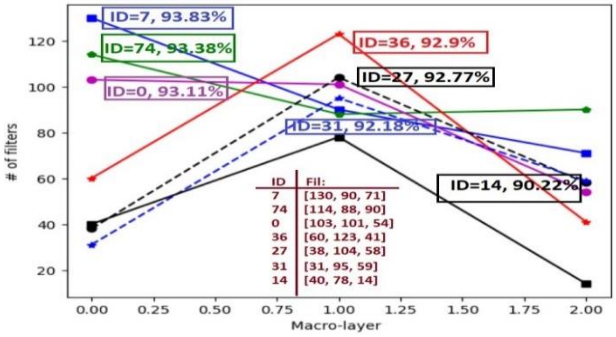

Fig. 9. Filter profiles for the best model selection after random search NAS.

While NAS search times were rather large (in the above example 10 hours are needed for 100 model trials) some heuristics were derived from NAS experiments, thus helping to accelerate the search process while starting with a "good guess" for the hyper-parameters.

B. *Heuristics For Rapidly Choosing Optimal Set Of Hyperparameters*

Extensive NAS for the VRES-CNN model using datasets with various image sizes led to the following conclusions in choosing the starting point for hyperparameters:

a) **choice of M – number of macro-layers**: it turns out that an optimal number of macro-layers is in direct relationship with the input image shape. Usually, well-balanced models were found when the images in the latest convolution layer were 4x4 size. Knowing that each macro-layer reduces 2 times the image size some empirical formulae for M would be: $M = \lfloor -2 + \log_2 sz \rfloor$ where sz is the size of the input image. For instance, in the case of 32x32 sized input images (sz=32) a good choice would be $M=3$. Of course, one could also try other values around (e.g. $M=2$ or $M=4$)

b) **choice of nonlinearity profile:** A good starting point was found to be NL=[2,2,1,1,0,0…0] (keeping only the first M elements in the list) noting that in general only the first 2 macro-layers require a nonlinearity degree.

c) **choice of flat**: In most of the cases choosing flat=0 (Global-Average-Pooling operator applied for the output of convolutional chain) is the best choice. A choice for flat=1 may be necessary for a small (e.g. 10-20) number of filters in the latest macro-layer. Otherwise, choosing flat=1 may give a very small increase in accuracy but with a large increase in the number of parameters (due to the output dense layer).

d) **choice of the filter profile (fil)**: The filter profile is the most sensitive and it must be carefully tuned to match the specificity of each dataset. In general two choices are more specific, depending on the number of classes and the problem complexity: an "inverted L profile" for the case with large number of classes (for example fil=[40, 60, 100, 100, 100] in the case of an 100 class problem) or "an inverted U profile" in the case of small number of classes (for example the profiles for IDs 37,27,31,14 in Fig. 9 or fil = [20, 50, 20] in the case of a problem with 10 classes such as SVHN, CIFAR10, etc..). In any case, the filter profile hyperparameter is the most sensitive to the specific dataset and its proper choice using heuristics combined with random search or educated guess can ensure a well-balanced, light model, under 100 k-parameters, as indicated in the next sections.

IV. Evaluation On Low Resolution Datasets

Several low-resolution image sizes of 32x32 were first considered and using the techniques discussed in Section III several well-balanced models were identified. For comparison a modified EtinyNet model [7] was considered in either its native implementation (the code was adapted from [18]) denoted as ET-NB-m (all NB=2 original output DLB macro-blocks present, while m is the only available hyperparameter called a multiplier) but also variations with NB=0 or 1 DLB macro-blocks at the output. Training the models was done on the Kaggle cloud using the GPU P100 accelerator. The VRES-CNN is implemented in Tensorflow-Keras while the modified EtinyNet model in Pytorch.

A. *CIFAR100*

This widely known dataset [19] has 50k-samples for training and 10k-samples for validation while each sample represents a labeled 32x32 sized colour (3-channels) image. Acquiring high accuracy is often a CNN model challenge and although very good validation accuracies (best reported so far is 96%) are reported [20] for very large CNN models, tiny models with less than 1M-parameters usually perform validation accuracies under 70%. For the VRES-CNN a selection of well-balanced models is given in the next table. Observe that accuracies are quite good given the very small models and have all better performances than EtinyNet versions with various values of the multiplier parameter.

TABLE I. Optimal VRES-CNN Models For CIFAR100

Model	flat	NL	Fil	Val. Acc (%)	.h5 Kbyte	K-pars
1	**0**	**2,1,1**	**143,122,148**	**62.27**	**1784**	**138**
2	0	2,2,1	60,80,100	61.34	862	59.37
3	0	2,1,1	50,100,100	59.36	807	55.87
4	0	2,1,1	50,100,50	52.93	588	37.62
ET-2-1	-	-	-	22.16	-	502
ET-2-0.5	-	-	-	19.18	-	150
ET-2-0.3	-	-	-	14.48	-	65
ET-0-1	-	-	-	28.1	-	277

While the best model is 1, a better compromise (model 2 shaded in yellow) with 2 times less parameters achieves a slightly lower accuracy (about 1% loss). Among the EtinyNet models the best performance of only 28% is obtained by the modified model ET-0-1 with no (0) DLB blocks, thus indicating that a versatility approach could eventually match better this kind of model to various datasets. The native EtinyNet model (ET-2-1) gives lower accuracy with more parameters.

B. *CIFAR10*

This dataset [19] is less difficult than CIFAR 100 while the same size of images belongs now to only 10 classes. Still, achieving good accuracy is a challenge, values around 90% being considered acceptable for light architectures. Some of the best VRES-CNN models are summarized in the next table, along with EtinyNet variants.

TABLE II. Optimal VRES-CNN models for CIFAR10

Model	flat	NL	Fil	Val. Acc (%)	.h5 Kbyte	K-pars
1	0	3,2,1	40,78,14	84.53	479	26.29
2	**0**	**2,1,1,1**	**50,100,100,50**	**86.24**	**815**	**55.43**
ET-0-1	-	-	-	**65.67**	-	**93**
ET-1-1	-	-	-	62.61	-	191
ET-2-0.5	-	-	-	57.43	-	127

Again, the best performance of 86.24% is achieved by VRES-CNN model 2 with only 55 k-pars. Reducing the parameters 2 times results in a slight decrease in accuracy (almost 2% loss) while the best EtinyNet model achieved only 65.67% accuracy in the case of a modified version with 0 (no) DLB blocks.

C. *SVHN*

The SVHN (street view home numbers) [21] dataset contains 32x32 images of house numbers and the aim is to recognize them in one of 10 categories. It is a difficult task and according to [19] accuracies up to 99% can be achieved,

although for tiny models achieving around 96-97% validation accuracy can be considered satisfactory.

TABLE III. OPTIMAL VRES-CNN MODELS FOR SVHN DATASET

Model	flat	NL	Fil	Val. Acc (%)	.h5 Kbyte	K-pars
1	0	2,2,1,1	50,80,100,60	96.47	900	61.56
2	**1**	**2,1,1,1**	**70,80,90,100**	**96.81**	**6298**	**513**
3	0	2,1,1,1	54,44,50,41	96.72	2182	170
ET-2-0.5	-	-	-	83.04	-	127
ET-2-1	-	-	-	87.83	-	455
ET-1-1	-	-	-	89.34	-	161
ET-0-1	-	-	-	**90.08**	-	**93**

The best compromise between accuracy and size is achieved by VRES-CNN model 1 while some minor gain in accuracy (less than 0.4%) would require a much larger model 2. Again, EtinyNet models were proved as not matching well this dataset with the native one achieving only 87.8% while the best modified version (ET-0-1 with no DLB macro-block) gives the best accuracy at a relative low number of parameters.

D. EMNIST – Balanced

This dataset [22] is a difficult one, among those associated with handwriting recognition. It aims to recognize handwritten characters from 47 categories. According to [20] the best validation accuracy achieved so far is 91.06%. Under these circumstances, the results of optimal VRES-CNN models reported in the next table are quite good (best compromise size-accuracy model 2 with 90.61% accuracy) given their very low complexity. Again, this accuracy is much higher than what is best achievable with the modified EtinyNet model (ET-0-1) of 86.3%.

TABLE IV. OPTIMAL VRES-CNN MODELS FOR EMNIST-BALANCED

Model	flat	NL	Fil	Val. Acc (%)	.h5 Kbyte	K-pars
1	**0**	**3,2,1**	**50,150,100**	**90.71**	**1344**	**98.61**
2	0	2,2,1	30,70,100	90.61	633	40.38
3	0	2,1,1	40,90,40	89.17	453	26.19
ET-2-1	-	-	-	85.25	-	474
ET-0-1	-	-	-	**86.3**	-	**168**

V. EVALUATION ON HIGH RESOLUTION DATASETS

Several large image size datasets were considered to demonstrate that VRES-CNN models are also capable of dealing with large image sizes as well. To avoid computational burden, in some cases (the Flowers dataset) TPU-VM accelerators (available on the Kaggle cloud platform) were used in some of these cases.

A. VWW Dataset

This binary classification dataset (whether an image contains a person or not) was proposed in [23] and is often used to test capabilities of TinyML models. Although the original set has more than 100k-samples, in order to speed-up the model search process, herein as smaller set from lab3 of course [24] with only 6.45 k-samples in the training set and 1.6k-samples in the validation set is used. Various image sizes of 64, 128 and 192 were considered, as indicated in the next table (flat=0 column was replaced with image size). Although for the original set, validation accuracies over 95% are reported for this tiny version of the set a reference value is 85.15% obtained by us after training a pretrained EfficientNetB0 model with 4000 k-parameters. The next table reports our best results, indicating that a relatively small model with only 27 k-parameters can achieve an accuracy of 79.8%, close to the best achievable with this small dataset.

TABLE V. OPTIMAL VRES-CNN MODELS FOR REDUCED VWW DATASET

Model	sz	NL	Fil	Val. Acc (%)	.h5 Kbyte	K-pars
1	64	2,2,1,1	40,60,100,50	74.59	704	43.1
2	128	3,2,1,1,1	30,40,70,40,30	78.5	566	27.2
3	128	2,1,1,1,1	20,20,40,20,40	77.7	328	9.03
4	128	2,1,1,1,1	10,10,20,10,10	73.7	251	3.02
5	**196**	**3,2,1,1,1**	**30,40,70,40,30**	**79.81**	**541**	**27.25**

B. FLOWERS 104 Dataset

This dataset [25] aims to correctly classify 104 flowers photos taken in a wide variety of circumstances. It is a relatively difficult dataset, and huge models are often required to get accuracies over 90%. The image size considered herein is 192, found to be the best compromise to get a good accuracy with limited computational resources. The best VRES-CNN models identified so far are given in the next table (here flat=0 for all cases). As seen, their accuracies are close to the state-of-the-art values, given the very low number of parameters.

TABLE VI. OPTIMAL VRES-CNN MODELS FOR THE FLOWERS-104 COMPETITION DATASET

Model	NL	Fil	Val. Acc (%)	.h5 Kbyte	K-pars
1	**2,2,1,1,1,1**	**100,120,140,120, 120,100**	**84.46**	**2610**	**198**
2	2,1,1,1,1	50,70,100,70,70	75.78	1180	79.6
3	2,1,1,1,1	50,70,100,70,50	73.25	934	62

C. TEXTURE Dataset

This dataset was introduced in [26] and contains 8674 high resolution images from 3 other public sources. The surface shape images are categorized into 64 classes. Herein the image size of 128x128 is considered. For reference, a validation accuracy of 93% is achieved after specific training on this dataset of the pretrained EfficientB0 model with 4000 k-parameters. A selection of our best (well-balanced) VRES-CNN models is given in the next table.

TABLE VII. OPTIMAL VRES-CNN MODLES FOR TEXTURE DATASETS

Model	NL	Fil	Val. Acc (%)	.h5 Kbyte	K-pars
1	**2,1,1,1,1**	**80,100,120,100,80**	**95.69**	**1538**	**110**
2	2,1,1,1,1	50,70,100,70,50	94.44	928	60.0
3	2,1,1,1,1	40,48,25,42,48	93.87	507	24.5

The best accuracy (95.69%) obtained by model 1 competes well with state-of-the-art results for this dataset and some better compromise between size and accuracy can be achieved as seen for models 2 and 3.

VI. Concluding Remarks

A compact convolutional neural network model called VRES-CNN is proposed and evaluated on a large variety of datasets in relation with image recognition problems.

What characterizes our approach as novel in contrast with most of the other tiny architectures proposed do far [6-12] is the simplicity in defining macro-layers so that they are emulating a nonlinear convolution operator. On the other hand, our architecture ensures that hyper-parameters can be defined as filter-profiles and nonlinear-profiles thus ensuring a versatile choice of them. This feature allows searching for the best well-balanced model achieving best accuracy with minimal resources. In contrast, the macro-layers of other architectural tiny models are pre-defined not only in their specific filter parameters but also in the number of macro-layers (often called blocks or macro-blocks). As a result, such models, often optimized for difficult datasets (such as ImageNet) would prove not be optimal for other, more common and widely used in specific applications, datasets. As indicated in Section IV, EtinyNet [7] performs worse in terms of validation accuracy than VRES-CNN while requiring a much larger number of parameters. Our preliminary results indicated in Section IV suggests that enabling versatility in such models would allow a better match to the specific datasets. Still, for the investigated datasets VRES-CNN proved to perform better.

Further research would include exploring model-compression techniques such as binarization [27] on the VRES-CNN for their deployment into specialized, e.g. FPGA-based TinyML platforms. Also, since most of the actual TinyML freely available code is published in Pytorch or Tensorflow, an in-depth comparison [28] to other tiny models recently reported in the literature will be considered.

References

[1] F Chen, S Li, J Han, F Ren, Z Yang, "Review of Lightweight Deep Convolutional Neural Networks", in Archives of Computational Methods in Engineering, 2023, Springer.

[2] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki and A. S. Hafid, "A Comprehensive Survey on TinyML," in IEEE Access, vol. 11, pp. 96892-96922, 2023.

[3] J. Lin, L. Zhu, W. -M. Chen, W. -C. Wang and S. Han, "Tiny Machine Learning: Progress and Futures [Feature]," in IEEE Circuits and Systems Magazine, vol. 23, no. 3, pp. 8-34, thirdquarter 2023.

[4] M. Tan and Q. Le. EfficientNet, "Rethinking model scaling for convolutional neural networks", in International Conference on Machine Learning, pages 6105–6114. PMLR, 2019.

[5] A. Howard, M. Sandler, B. Chen, W.J. Wang, L-C. Chen, M.X. Tan, "Searching for MobileNetV3", in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019; pp.1314–1324.

[6] J. Lin et al., "MCUNetV2: Memory-efficient patch-based inference for tiny deep learning," 2021, arXiv:2110.15352.

[7] K. Xu, Y. Li, H. Zhang, R. Lai, and L. Gu, ''EtinyNet: Extremely tiny network for TinyML,'' in Proc. AAAI Conf. Artif. Intell., vol. 36, no. 4, 2022, pp. 4628–4636.

[8] Y. Li et al., "MicroNet: Improving Image Recognition with Extremely Low FLOPs," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 458-467.

[9] R.T.N.V.S. Chappa and M. El-Sharkawy, "Squeeze-and-Excitation SqueezeNext: An Efficient DNN for Hardware Deployment," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0691-0697.

[10] Y. Jiang, F. Yang, H. Zhu, D. Zhou, and X. Zeng, "Nonlinear CNN: Improving CNNs with quadratic convolutions," Neural Comput. Appl.,vol. 32, no. 12, pp. 8507–8516, Jun. 2020.

[11] R. Dogaru and I. Dogaru, "NL-CNN: A Resources-Constrained Deep Learning Model based on Nonlinear Convolution," 2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE), Bucharest, Romania, 2021, pp. 1-4.

[12] R. Dogaru and I. Dogaru, "V-CNN: A Versatile Light CNN Structure For Image Recognition On Resources Constrained Platforms," 2023 8th International Symposium on Electrical and Electronics Engineering (ISEEE), Galati, Romania, 2023, pp. 44-47. (code implementation provided as https://github.com/radu-dogaru/V-CNN)

[13] G. Zoumpourlis, A. Doumanoglou, N. Vretos, and P. Daras., "Nonlinear convolution filters for CNN-based learning". In ICCV, 2017.

[14] R. Dogaru, P. Julian, L. Chua, and M. Glesner, "The simplicial neural cell and its mixed-signal circuit implementation: an efficient neural-network architecture for intelligent signal processing in portable multimedia applications," IEEE Transactions on Neural Networks, vol. 13, no. 4, pp. 995–1008, 2002.

[15] M. Villemur, P. Julian, and A. G. Andreou, "Energy aware simplicial processor for embedded morphological visual processing in intelligent internet of things," Electronics Letters, vol. 54, no. 7, pp. 420–422, 2018.

[16] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS", 2019, https://arxiv.org/pdf/1912.06059.pdf

[17] A.D. Mirică, R. Dogaru and I. Dogaru, "Optimization of a TinyML V-CNN architecture for street number recognition using random search", Doctoral Symposium on Electronics, Telecommunications, & Information Technology, October 4–6, 2023, Bucharest, Romania, october 2023.

[18] Code implementation of EtinyNet in Pytorch: https://github.com/Chadlikouider/TinyML_Det/blob/main/models/nn/etinynet.py , last accessed April 2024.

[19] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", PhD thesis, Computer Science, University of Toronto, 2009.

[20] Papers with Code (datasets section), URL: https://paperswithcode.com/ , last accessed January 2024.

[21] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng., "Reading digits in natural images with unsupervised feature learning", NIPS Workshop, 2011.

[22] G. Cohen, S. Afshar, J. Tapson, et al, "EMNIST: extending MNIST to handwritten letters", 2017 International Joint Conference on Neural Networks (IJCNN). IEEE; 2017.

[23] A. Chowdhery, P. Warden, J. Shlens, A. Howard, and R. Rhodes, "Visual wake words dataset", arXiv preprint arXiv:1906.05721 (2019).

[24] S. Han et al., "TinyML and Efficient Deep Learning Computing", MIT Course URL: https://hanlab.mit.edu/courses/2023-fall-65940 , 2023.

[25] M. McDonald, M. Görner, P. Bailey, P. Culliton, Y. Guo. (2020), "Flower Classification with TPUs", Kaggle url: https://kaggle.com/competitions/flower-classification-with-tpus

[26] Y. Huang, C. Qiu, X. Wang, S. Wang, and K. Yuan. "A Compact Convolutional Neural Network for Surface Defect Inspection". In: Sensors (2020).

[27] R. Dogaru and I. Dogaru, "Fast Training of Light Binary Convolutional Neural Networks using Chainer and Cupy," 2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Bucharest, Romania, 2020, pp. 1-4.

[28] M. C. Chirodea, O. C. Novac, C. M. Novac, N. Bizon, M. Oproescu and C. E. Gordan, "Comparison of Tensorflow and PyTorch in Convolutional Neural Network - based Applications," 2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 2021, pp. 1-6.