Increasing biases can be more efficient than increasing weights

Carlo Metta[0]
ISTI-CNR Pisa, Italy

Marco Fantozzi
University of Parma, Italy

Andrea Papini
Scuola Normale Superiore, Pisa, Italy

Gianluca Amato[1]
University of Chieti-Pescara, Italy

Matteo Bergamaschi
University of Padova, Italy

Silvia Giulia Galfrè[2]
University of Pisa, Italy

Alessandro Marchetti[3]
University of Chieti-Pescara, Italy

Michelangelo Vegliò
University of Chieti-Pescara, Italy

Maurizio Parton[1]
University of Chieti-Pescara, Italy

Francesco Morandin[1]
University of Parma, Italy

Abstract

We introduce a novel computational unit for neural networks that features multiple biases, challenging the traditional perceptron structure. This unit emphasizes the importance of preserving uncorrupted information as it is passed from one unit to the next, applying activation functions later in the process with specialized biases for each unit. Through both empirical and theoretical analyses, we show that by focusing on increasing biases rather than weights, there is potential for significant enhancement in a neural network model's performance. This approach offers an alternative perspective on optimizing information flow within neural networks. See source code [5].

1. Introduction

Historically the structure of the perceptron, the artificial neural network's fundamental computational unit, has rarely been questioned. The biological inspiration is straightforward: input signals from the dendrites are accumulated at the soma (with a linear combination), and if the result is above the activation threshold (that is, the opposite of some bias) there is a nonlinear reaction, as the neuron fires along the axon (with the activation function).

In time, the early sigmoid activation function was replaced by ReLU and variants, and the biological analogy became less stringent, shifting focus on the desirable mathematical properties of the class of functions computed by the networks, like representation power and non-vanishing gradients.

This has brought us to the current situation in which most units output their signal through a nonlinear activation function which effectively destroys some information. In fact, ReLU is not invertible, as it collapses to zero all negative values. Though some of its variants may be formally invertible (leaky ReLU [21] and ELU [4] for example), the fact that they overall perform in a way very similar to ReLU, suggests that their way of compressing negative values via a small derivative bijection leads to the same general properties of the latter.

In this paper, we investigate a radical rethinking of the standard computational unit, where the output brings its full, uncorrupted information to the next units, and only at this point is the activation function applied, with biases specialized for each unit. From the biological point of view, this is like having the activation at the dendrites instead of at the base of the axon, and correspondingly we call the new unit 'DAC', for 'Dendrite-Activated Connection'.

This kind of reversed view has already proved fruitful in the evolution from ResNets v1 [11] to v2 [12] when a comprehensive ablation study showed that for residual net-

[0]EU Horizon 2020: G.A. 871042 SoBig-Data++, NextGenEU - PNRR-PEAI (M4C2, investment 1.3) FAIR and "SoBigData.it".
[1]Funded by INdAM (groups GNAMPA, GNCS, and GNSAGA).
[2]Partially supported by SPARK Pisa.
[3]National PhD in AI, XXXVII cycle, health and life sciences, UCBM.
[all]Computational resources provided by CLAI lab, Chieti-Pescara.

works it is better to keep the information backbone free of activations for maximum information propagation, and pre-activate the convolutional layers in the residual branch.

Here this view is taken forward: not only the units are pre-activated, but the biases become specific to each input-output pair, as the weights are. We refer to this as having *unshared biases*.

The main result of this paper is evidence that sometimes incorporating more biases can increase accuracy more than adding weights, without altering model complexity (see Section 5, SGEMM subsection). The fact that DAC consistently outperforms the baseline models across diverse architectures and datasets strengthens this finding, see the rest of Section 5. Hence, DAC emerges as a viable strategy for enhancing neural network performance when increasing weights proves ineffective.

The proposed model is introduced in Section 2 with a first theoretical discussion. Related works are listed in Section 3. The main practical details for replacing shared with unshared biases are discussed in Section 4, and the empirical experiments can be found in Section 5. Finally, further theoretical questions can be found in Section 6 and in [23].

2. Model

In a standard neural network, units are often post-activated: they compute a linear combination of their inputs and then apply a nonlinear filter to the result.

$$\begin{cases} z_i = \sum_{j \in \mathcal{I}_i} w_{i,j}\, y_j & \text{linear aggregation} \\ y_i = \varphi(b_i + z_i) & \text{nonlinear filter} \end{cases} \tag{1}$$

where, for unit i, $\mathcal{I}_i$ denotes the set of input nodes, b_i the bias and φ the activation function.

We propose to consider units that are pre-activated with unshared biases:

$$\begin{cases} y_{i,j} = \varphi(b_{i,j} + z_j) & \text{nonlinear filter} \\ z_i = \sum_{j \in \mathcal{I}_i} w_{i,j}\, y_{i,j} & \text{linear aggregation} \end{cases} \tag{2}$$

In (1) there is one weight $w_{i,j}$ for each connection between units and one bias b_i for each unit. In (2) each connection still has its own weight $w_{i,j}$, but it also has its own nonlinear filter with a specific bias $b_{i,j}$. (Compare Figures 1 and 2.)

Connections between units correspond to synapses or dendrites in biological neurons, and it is known that activation at the level of dendrites can actually occur in biological neural networks [18, 22]. With this motivation we refer to a connection as in (2) as a Dendrite-Activated Connection (DAC). See [23, Section A] for additional details on the biological inspiration.

A DAC unit is more general than a standard unit, in fact it can have almost twice as many parameters, and a correspondingly greater representation capacity, see [23, Section C]. For this reason, in experiments, neural networks with DACs should be compared to standard ones with similar number of parameters or computational complexity, and not with the same number of units or channels. In this regard it is important to note that, in convolutions larger than 1×1, DAC biases can be partially shared, yielding a much lesser increase in the number of parameters (see Section 4).

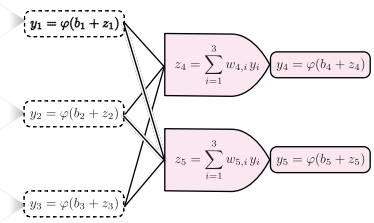

Figure 1. Standard connection between two consecutive layers. The output layer (pink) is fully connected and has two units labelled 4 and 5. The input layer has three units: $\mathcal{I}_4 = \mathcal{I}_5 = \{1, 2, 3\}$. Bullets and rectangles represent linear aggregation and nonlinear filters from (1), respectively. Units 4 and 5 must share the same biases b_1, b_2, b_3 in the activation of their inputs.

In this paper we investigate the properties of DAC units and the behaviour of typical network structures when standard connections are replaced with DACs.

Remark 2.1. DAC and the standard post-activation may coexist in the same connection (in analogy with what happens in the biological neuron, where the standard post-activation of the axon is always present):

linear combination $\to$ shared bias post-activation
$\to$ unshared biases pre-activation $\to$ linear combination

For ReLU activations, the composition of post-activation and pre-activation is equivalent to a pre-activation with modified coefficients, so we do not investigate further this generalization.

3. Related work

The multi-bias activation (MBA) from [19] replicates K times the input features z_j, applies a different bias parameter $b_j^{(k)}$ to each of them, filters them with ReLU, and then computes a linear combination over j and k for every output node i:

$$z_i = \sum_{j \in \mathcal{I}_i} \sum_{k=1}^{K} w_{i,j,k}\, \varphi\big(b_j^{(k)} + z_j\big). \tag{3}$$

Equation (3) resembles our equation (2) of pre-activated units. However, in (3) the pre-activation biases do not depend explicitly on the output i, so they are multiplied in number, but still effectively shared from the point of view of

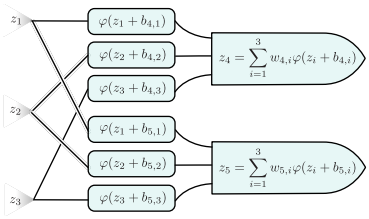

Figure 2. Same structure as in Figure 1 with post-activation replaced by pre-activation with unshared biases. Rectangles and bullets represent nonlinear filters and linear aggregations from (2), respectively. The biases in the activation between the input and the output layer depend both on the input node (1, 2 or 3) and the output node (4 or 5), and so, from the point of view of the output units, we refer to them as *unshared*.

the linear operator. Another consequence of this is that parameters and computations are increased K-fold. Squeeze MBA [8] is a variation of MBA that still shares biases among outputs but tweaks the network structure to partially reduce the number of parameters.

Other approaches to mitigate the loss of information intrinsic in the ReLU activation, such as Maxout [10], adaptive piecewise linear activation functions [1], Concatenated ReLU [31], and Activation ensembles [16], generalize the activation function by using multiple biases (among other parameters) but they all maintain a single nonlinear filtered output per node, and hence shared biases.

There are also ways to design a network that *inherently* mitigate or avoid the loss of information of activations with shared biases. ResNet v2 architecture [12] keeps the information backbone free of nonlinearities that are only on residual branches (with pre-activation). All the nonlinear blocks take their inputs from the backbone, so each input is the pure linear sum of all the previous nonlinear branches. ConvNeXt [20] not only uses the same linear backbone as above, but then applies depthwise convolutions, brought to popularity by [3] and common to other recent successful architectures. In depthwise convolutions, every input channel is unique to one output kernel: this is a simple solution to avoid sharing biases, though the consequent low capacity requires that depthwise convolutions are used together with other types of layers that will typically have shared biases.

4. Methods

To use DAC in a given neural network structure, one replaces the usual post-activations of computational units with pre-activations for the downstream units, using unshared biases. To this end one can design dense and convolutional units that include the required pre-activation.

For a dense layer with n units and m inputs, (2) becomes:

$$f_i(z) = \sum_{j=1}^{m} w_{i,j}\,\varphi(b_{i,j} + z_j), \qquad i = 1, 2, \ldots, n \tag{4}$$

In the case of a 2d convolutional layer with n units/kernels and m input channels, we get instead:

$$f_{h,k,i}(z) = \sum_{a,b=-l}^{l} \sum_{j=1}^{m} w_{a,b,i,j}\,\varphi(b_{i,j} + z_{h+a,k+b,j}), \quad i = 1, 2, \ldots, n, \quad (h,k) \in \text{grid} \tag{5}$$

where $L = 2l + 1$ and $L \times L$ is the kernels size.

In the latter equation, if one were to follow the principle in (2), for which there should be one pre-activation bias for every weight, then these totally unshared biases would take the more general form $b_{a,b,i,j}$ depending on channel, kernel and position in the kernel. There is however the possibility, in this case, to partially share biases and have $b_{i,j}$ depend on input channel and output kernel only, to get a better trade-off between flexibility and number of parameters.

In most cases, pre-activated layers (4) and (5) can replace the usual ones (by removing the post-activation of the layer before the one considered, but see Remark 2.1). In a few cases this change could have no real effect: in fact there are situations in which the standard shared biases are effectively unshared, notably when the subsequent layer has only $n = 1$ unit/kernel, or if it is a depthwise convolution [3]. In these cases DAC reduces to a standard connection. In all other situations, DAC will use more parameters and require more computations than a standard connection.

Parameters and number of operations. A dense layer with n units and m inputs, has mn weights. When it is pre-activated with unshared biases as in (4), a total of mn DAC biases are used, in place of the m shared biases in the post-activation of the layer before that one, for a relative increase factor $2 - 1/n$, that is, almost 2 when n is not very small. For a convolutional layer as in (5), the number of biases involved is the same, but the weights are mnL^2, so the relative increase factor is $1 + 1/L^2 - 1/nL^2$. In the case of a 3×3 kernel, this leads to a modest increase of approximately $+11\%$ in the number of parameters.

In calculating FLOPs, we adopt the usual convention to ignore activation function costs. Though pre-activation involves significantly more calls to the activation function than post-activation, for simple functions like ReLU, the computation cost is very small and can be neglected. This assumption may not hold for other, more computationally expensive, activation functions.

Considering again a dense layer with n units and m inputs, FLOPs are mn multiplications and mn additions. When using pre-activation, another mn additions are required, that replace the m additions in the post-activation

of the layer before that one. The relative increase factor is $1.5 - 1/2n$, that is, almost $+50\%$ when n is not very small. In the case of a convolutional layer, FLOPs per unit/kernel are mL^2 multiplications and mL^2 additions, totaling $2mnL^2st$ operations for an output shape of $s \times t$. When using pre-activation as in (5), since DAC biases $b_{i,j}$ do not depend on the particular kernel position, initial activation results $\varphi(b_{i,j} + z_{\cdot,\cdot,j})$ can be *cached*, requiring $mnst$ additions, in place of the mst additions in the post-activation of the layer before that one. This results in a relative increase factor of $1+0.5L^{-2}(1-1/n)$, or about $+5.5\%$ for 3×3 convolutions and $+50\%$ for 1×1 convolutions.

More complex structures. While the above discussion explains how to use DACs in the case of a plain network, i.e. a regular sequence of basic layers, it is not always clear what to do in more realistic situations, that include, for example, normalization layers and skip connections. The guiding principle then, is to identify activations followed by linear operators, and check whether, from the point of view of the linear operators, those activations are using shared biases. If this is the case, using DAC means to remove said activations and add pre-activations with unshared biases to the linear operators.

Hence for example, in a periodic sequence like

$$\ldots \to \text{ReLU} \to \text{linear} \to \text{BN} \to \text{ReLU} \to \text{linear} \to \ldots$$

where BN stands for batch normalization and "linear" might be either fully connected or convolutional layers, a DAC version would be something like

$$\ldots \to \text{DAC} \to \text{BN}' \to \text{DAC} \to \ldots$$

where BN' is batch normalization without the trainable shift parameter β and DAC might be either (4) or (5).

5. Experiments

We tested empirically the effect of using pre-activations with unshared biases, on several common tasks. All the experiments can be reproduced using the source code [5], that includes a test implementation of the pre-activated layers in (4) and (5). For the training we used L^2 regularization instead of weight decay, applying it only to weights and not to biases. DAC biases were initialized to zero. For full details on training hyperparameters see [23, Section 5].

SGEMM performance regression. We experimented with a regression task from the UCI repository [2, 30]. The task is to predict the execution time of a matrix multiplication on a highly-tuneable SGEMM kernel for GPU. The input variables are 14 kernel parameters that are either binary or take values that are powers of 2, and the response variable ranges between 13.25 and 3397.08 milliseconds. Non-binary variables, including the response, were $\log_2$-transformed. Input variables were normalized and the dataset, consisting of all the 241600 combinations, one replica each, was split with a 70%, 15% and 15% proportion between training, validation and test. We assessed performances as mean squared error (MSE) between predicted and real response (in $\log_2$ scale).

The nature of the task suggests to use fully connected neural networks. We tested several hundreds architectures, all with the same basic structure but variable size. The structure is a simple sequence of fully connected layers, each followed by batch normalization and ReLU. The output is a fully connected layer with 1 unit and no activation. The size ranges in depth (from 5 to 16 layers before the last one), in width (from 16 to 250 units) and in the progression of widths with layers, that was either rectangular (constant width) or pyramidal (decreasing width). The number of trainable parameters ranged between 1700 and 1.4M.

Network structures were defined in pairs, consisting of one network with post-activations and shared biases (baseline), and one network with pre-activations and unshared biases (DAC) with similar number of trainable parameters. To this end, in the DAC version, the number of units in each layer was reduced by roughly a factor $\sqrt{2}$, in such a way that the number of weights was halved, and adding the unshared biases, the total number of trainable parameters was almost exactly the same as for the baseline network.

Training failed to reach a suitable error level in 15/576 cases for the baseline and 2/449 cases for DAC. The occurrence of this problem is significantly lower for DAC (bilateral p-value 0.0063). Failed training experiments were excluded from further analysis as outliers. See [23, Figure 8] for the results of all the other experiments.

We found that there is weak dependence on depth for this task, and hence we aggregated experiments by width, see Figure 3. Since DAC networks with the same number of parameters have reduced number of units, we computed their "equivalent width", i.e. $\sqrt{2}$ times the actual width: for example rectangular DAC networks with 141 units per layer have as many parameters as base networks with the same depth and 200 units per layers, and so we say they have equivalent width 200. For rectangular networks, there are four groups with widths (or equivalent widths) 25, 50, 100 and 200 for base (or DAC). For pyramidal networks, the width of layers is not uniform, so we estimated an average width (or equivalent width) as $\sqrt{\text{parameters}/\text{depth}}$. Obtained values formed six natural clusters that were used as groups.

From these initial analyses we observed that passing from shared to unshared biases can be inefficient if the network is too small and hence has insufficient capacity. To better explore the matter of efficiency of biases with respect to weights, we considered a situation in which one wants to enlarge a standard (shared-biases) model that has N weights, either by adding weights or by adding biases.

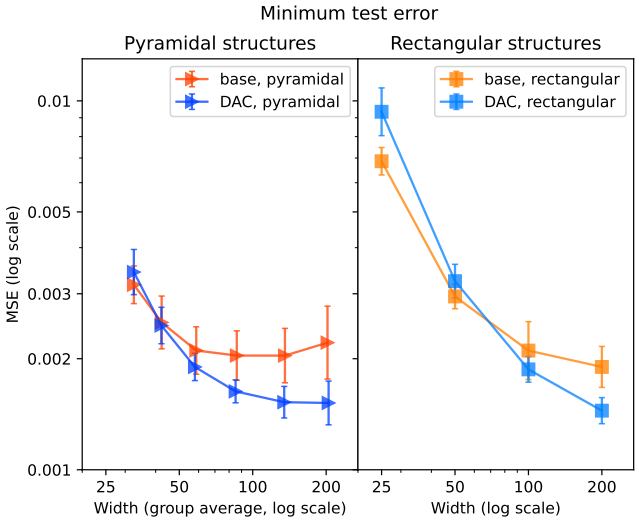

Figure 3. Aggregated and averaged results for the SGEMM regression task. Experiments are grouped by network shape (pyramidal or rectangular, see text) and width. Error bars represent the sample standard deviation of the values concurring to the average. Fully connected networks with DACs perform better than the baseline for larger widths, and similarly or worse for smaller widths, when the general performance of the network is far from optimal.

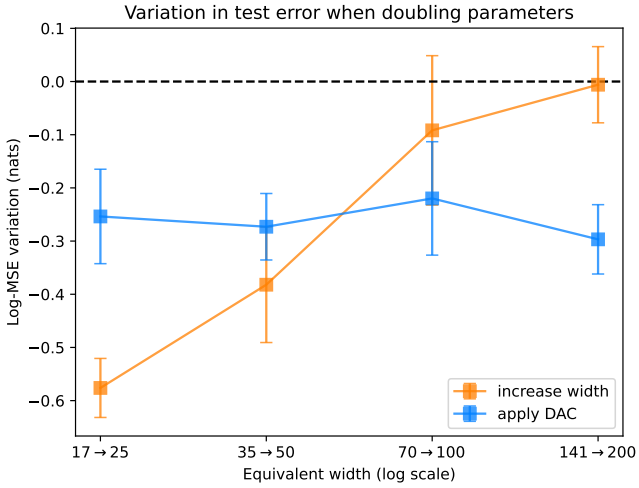

Figure 4. Efficiency analysis of unshared biases for the SGEMM regression task. Rectangular baseline networks were compared with models with double the parameters: either by adding weights (orange) or by making biases unshared (blue). The resulting variations of the MSE are shown (negative means improvement). Error bars represent the sample standard deviation of the values concurring to the average (see text).

We measured the error reduction obtained when the number of parameters doubles in these two ways: by increasing the number of units by a factor $\sqrt{2}$ (adds about N weights), or by making the biases unshared (adds about N biases). To estimate this reduction, first we obtained for each experiment the averages of log-MSE across replicates; then we computed the differences between models of size normal and double, for all widths and depths; then, since these differences did not show dependence on the depth, see [23, Figure 9], for each width we collected the different depths and computed the average of the values.

The results are shown in Figure 4. Increasing the number of weights (orange) has diminishing returns when the width increases, with large benefits for small sizes and almost no improvement passing from 141 to 200 units per layer. Passing from shared to unshared biases instead (blue), gives a uniform improvement of about 0.25 nats in the test error. This confirms that small networks, with error levels that are far from optimal, benefit more from increased width than from unshared biases, but when further width increase is no more beneficial, then using unshared biases gives an additional boost of performances and leads to otherwise unreacheable error levels.

Image classification tasks. We further experimented with convolutional architectures for image classification tasks, like VGG and ResNet. In all experiments we trained a standard structure with post-activations and shared biases (baseline) and a similar network with pre-activations and unshared biases (DAC). In the case of convolutional layers, using unshared biases only increases marginally the number of parameters and the number of operations (see Section 4), so we did not reduce the number of units in the DAC version.

We chose three classification tasks with datasets of diverse nature: CIFAR-10 and CIFAR-100 [17]; two subsets of ImageNet called Imagenette and Imagewoof [14]; and ISIC 2019, a medical images dataset, consisting of skin lesion images [13]. Below we give further details and results of the various experiments.

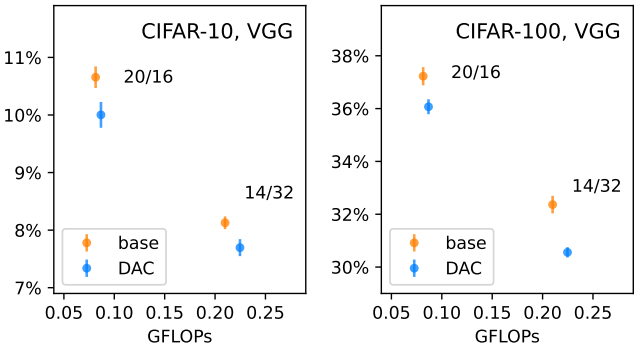

Figure 5. VGG, average test error. Compared performances of VGG 20 layers, 16 channels, and VGG 14 layers, 32 channels with shared biases (baseline, orange) and unshared biases (DAC, blue) on CIFAR-10 and CIFAR-100. Test error (vertical axis) is averaged over 5 replicates and over 5 epochs (see text). Error bars are 95% confidence intervals for the true mean value. Complexity (horizontal axis) is measured in GFLOPs per forward pass.

Plain convolutional networks on CIFAR. We designed two VGG-like architectures for CIFAR-10 and CIFAR-100, in line with modern revisitations like [6]. The structure is a simple sequence of 3×3 convolutional layers, each followed by batch normalization and ReLU. The output is a global average pooling (GAP) layer, followed by a fully connected layer with 10 or 100 units and softmax activation. The first structure (referred as 20/16) has 20 layers including the output, and the 19 convolutional layers start with 16 kernels that become 32 and 64, after 7 and 13 layers. The second structure (14/32) has 14 layers, starts with 32 kernels and doubles after 5 and 9.

Each experiment was replicated 5 times with a 5-fold cross-validation scheme. The best test accuracy was estimated by averaging over the 5 replicates and over 5 epochs centered on the best epoch on the validation dataset, see [23, Section B] for details on this robust statistical estimator.

The networks with pre-activations and unshared biases obtained a systematic improvement in the test accuracy with respect to the baselines. The improvement is statistically significant in all cases. For the two architectures 20/16 and 14/32, the improvement was $0.65\% \pm 0.17\%$ and $0.43\% \pm 0.09\%$ on CIFAR-10, and it was $1.16\% \pm 0.22\%$ and $1.81\% \pm 0.19\%$ on CIFAR-100, see Figure 5. The models with unshared biases require only a marginal increase in complexity, with 11% more parameters and 5.5% more FLOPs than the baseline, so these improvements cannot be explained just by the larger size of the models.

Remark 5.1. To measure the size of the models, in this and other plots, we favor floating point operations (FLOP) over the number of parameters, as was advocated among other sources in [29]. We provide plots with the number of parameters in [23].

ResNet networks on CIFAR. We used as baseline the architecture proposed for CIFAR in the original ResNet papers [11, 12], with 20, 32, 44, and 56 layers ($n = 3, 5, 7, 9$) both with the v1 post-activated [11] architecture and the v2 pre-activated [12] architecture. The structure is similar to the VGG of the previous section, but with skip connections every two layers. More explicitly, it starts with a convolution with 16 kernels, followed by three stages of n residual blocks each, with 16, 32 and 64 kernels respectively. The output is GAP plus fully connected, as before.

The residual blocks come in two versions, referred as v1 and v2. For v1 there is a classical post-activation sequence:

$$\text{conv} \to \text{BN} \to \text{ReLU} \to \text{conv} \to \text{BN} \to +\text{input} \to \text{ReLU}$$

For v2, instead, the authors found that ResNet performs better with a pre-activations sequence:

$$\text{BN} \to \text{ReLU} \to \text{conv} \to \text{BN} \to \text{ReLU} \to \text{conv} \to +\text{input}$$

The conversion to pre-activation with unshared biases (DAC) was done as follows:

$$\text{DAC} \to \text{BN} \to \text{DAC} \to \text{BN} \to +\text{input} \quad \text{(v1)}$$
$$\text{BN} \to \text{DAC} \to \text{BN} \to \text{DAC} \to +\text{input} \quad \text{(v2)}$$

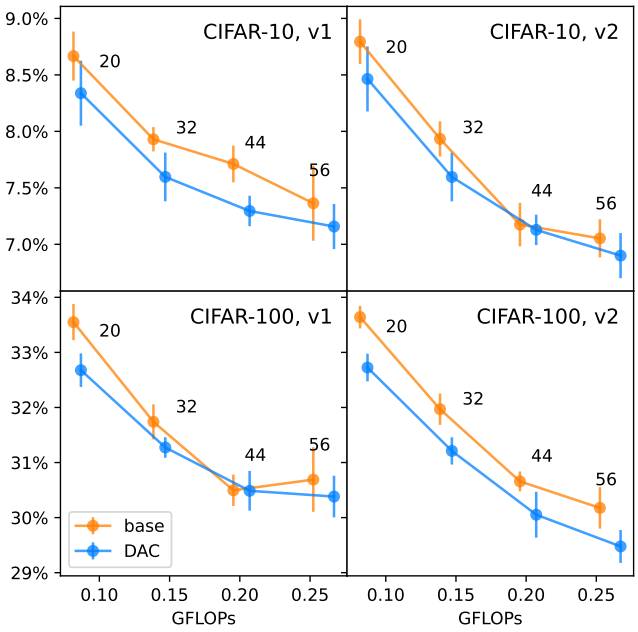

Figure 6. ResNet, average test error. Compared performances of ResNet networks with shared biases (baseline, orange) and unshared biases (DAC, blue), on CIFAR-10 (top) and CIFAR-100 (bottom), with architectures v1 (left) and v2 (right). Floating numbers are the layers count. Test error (vertical axis) is averaged over 5 replicates and over 5 epochs (see text). Error bars are 95% confidence intervals for the true mean value. Complexity (horizontal axis) is measured in GFLOPs per forward pass.

The training hyperparameters were the same as in the plain convolutional networks experiments.

Figure 6 summarizes the results in terms of best test accuracy, estimated robustly as for the VGG experiments, see [23, Section B]. Among the 16 comparisons, 15 are in favor of DAC models, with 9 of them statistically significant (p-value below 5%). This confirms that using unshared biases improves the performances of residual convolutional networks on CIFAR, with only a marginal increase in the model complexity. In fact, one could argue that, if the improvement of the test error between baseline and DAC was due only to the larger sizes of the latter, then in the figure, the decreasing blue and orange lines connecting models of growing depth, would be superimposed. They are partially superimposed for CIFAR-10, v2 and CIFAR-100, v1, but well separated in the other two cases.

Table 1 presents a simpler metric: the lowest test error rate (calculated as the minimum over the epochs of the average of the 5 replicates). The performance of the baselines

Table 1. Comparison of minimum test errors in percent points on CIFAR-10 and CIFAR-100 (± one standard error).

	CIFAR-10, v1		CIFAR-10, v2	
Layers	Base	DAC	Base	DAC
20	8.64±0.12	**8.28**±0.19	8.73±0.10	**8.33**±0.07
32	7.90±0.04	**7.57**±0.11	7.85±0.09	**7.56**±0.15
44	7.69±0.09	**7.27**±0.06	7.14±0.09	**7.10**±0.05
56	7.34±0.17	**7.13**±0.11	7.02±0.09	**6.89**±0.05
	CIFAR-100, v1		CIFAR-100, v2	
Layers	Base	DAC	Base	DAC
20	33.50±0.16	**32.54**±0.04	33.50±0.11	**32.62**±0.10
32	31.66±0.12	**31.09**±0.09	31.79±0.12	**31.05**±0.12
44	**30.40**±0.11	30.42±0.19	30.50±0.07	**29.93**±0.20
56	30.53±0.23	**30.32**±0.19	29.89±0.22	**29.32**±0.17

Table 2. Comparison of minimum test errors in percent points on Imagenette and Imagewoof (± one standard error).

	Imagenette, v1		Imagenette, v2	
Layers	Base	DAC	Base	DAC
20	13.42±0.26	**11.69**±0.11	11.98±0.17	**11.79**±0.13
32	13.14±0.45	**11.75**±0.08	11.40±0.09	**11.28**±0.11
	Imagewoof, v1		Imagewoof, v2	
Layers	Base	DAC	Base	DAC
20	23.20±0.26	**22.61**±0.25	22.61±0.17	**21.70**±0.22
32	23.06±0.24	**21.32**±0.29	21.75±0.17	**20.65**±0.14
GFLOPs	0.509	0.542	0.509	0.542
	0.865	0.918	0.865	0.918

aligns with typical values for networks of similar complexity found in the literature. Both the figure and table suggest that using unshared biases in ResNet architectures leads to measurable performance improvements across most versions and benchmark datasets. In some instances, using unshared biases improves the corresponding baselines despite having fewer layers.

We performed also an experiment on CIFAR-10 using a four times wider ResNet20 v2 architecture, with 64, 128 and 256 kernels in the three stages, resulting in a total of 4.3M parameters for the baseline model. Using unshared biases instead of shared biases, the minimum test error dropped from 5.69% to 5.16%, while the number of parameters increased as usual by 11%, and the FLOPs by 5.5%.

ResNet networks on Imagenette and Imagewoof. Imagenette and Imagewoof [14] are subsets of ImageNet that are frequently utilized for model benchmarking. They offer a simpler and faster alternative to ImageNet while preserving many of its challenges. Imagenette comprises about 10k images belonging to 10 easily distinguishable classes, whereas Imagewoof includes 10 classes that are more challenging to classify due to their similarities, as they represent 10 different dog breeds.

Table 3. Ablation study isolating the contribution made by pre-activation only (with shared biases) for ResNet20 architectures on CIFAR. Reported values (absolute) are the estimated improvements in test accuracy when replacing the standard post-activation with pre-activation. Relative values are relative with respect to the estimated improvement when using full DACs, with pre-activations and unshared biases.

	Absolute		Relative to DAC	
	v1	v2	v1	v2
CIFAR-10	−0.13%	+0.08%	−41%	+31%
CIFAR-100	+0.08%	+0.07%	+9%	+8%

For each dataset, we selected the 160-pixel version and resized it to 80×80 pixels. We used the 20 and 32 layers ResNet structure of the CIFAR-10 experiments, with the same training hyperparameters.

Table 2 summarizes the resulting lowest test error rates (calculated as the minimum over the epochs of the average of the 5 replicates). The performance of the baselines aligns with typical values for networks of similar complexity found in the literature. Using unshared biases (DAC) shows marked performance improvements, again at the cost of a marginal increase in size and complexity. In particular it can be observed that the baseline ResNet v1 (post-activated) is much worse than baseline v2 and DACs (both pre-activated).

ResNet networks on images for melanoma diagnosis. To investigate performances in image classification tasks that are both more difficult and more applied, we conducted an experiment using a ResNet v1 model on a real-world dataset from *International Skin Imaging Collaboration* (ISIC). We used ISIC 2019 [13], a collection of 25330 quality-controlled dermoscopic images of skin lesions, divided into 8 classes. With 20 layers, the baseline error is 27.05%, and with DAC is 26.47%, for an improvement of 0.58%. The same figures for 32 layers are 26.30% and 25.04%, for an improvement of 1.26%.

Ablation study: pre-activation with shared biases. Since unshared biases cannot be realized without pre-activations, in all previous DAC experiments we used both together. We decided then to investigate briefly with pre-activation only (hence, with shared biases). We trained ResNet20 v1 and v2 architectures, on CIFAR-10 and CIFAR-100.

We found that these models performed comparably to the baseline model, but distinctly worse than using full DACs, with pre-activations and unshared biases, see Table 3. In one case the result was slightly worse than the baseline, and

even in the other cases the improvements were never statistically significant.

Comparison with MBA. Since multi-bias activation (see Section 3) introduced in [19] is based on a principle close to DAC, we tested it on the same ResNet20 architecture on CIFAR-100. When converting a baseline structure to MBA, the number of parameters increases K-fold, so these models are much larger than DAC, that increases the parameters by 11%. In spite of this, we found improvements over the baseline smaller than using DACs (0.52%, 0.64%, 0.14% for MBA with $K = 2, 4, 8$, and 0.97% for DAC). It is possible that MBA might require a specific tuning of hyperparameters to reach better performances, but since we did not do this for DAC models, we did not investigate the matter further.

6. Theoretical discussion

This section explores theoretical arguments that support the efficiency of using pre-activations with unshared biases, highlighting the fact that this improves considerably the flexibility of the model.

Last and first layers. In a plain fully connected neural network with ReLU activations and standard units, one would expect a ReLU activation at the output of the very last layer. However, this is undesirable as the final output should be informative and compatible with the loss function (e.g., logits for cross-entropy). So one usually has to remove that last activation, which, using pre-activations, would not have existed in the first place.

Symmetrically, consider the first layer of a similar network: with standard units, it would not make sense to filter the input nodes with ReLU. Nevertheless, with unshared biases, it is instead very reasonable to apply the nonlinearity to the input nodes, because different units in the first layer might benefit from tailored filtering of the input.

In both cases pre-activated units with unshared biases seem more natural.

Input replication. Filtering the input as just described might be even more useful if the input is replicated multiple times. Consider the toy example of a one-dimensional input x and a shallow network with only one layer of one unit aiming at approximating some function $f : \mathbb{R} \to \mathbb{R}$. A pre-activated unit '0' with input replicated n times $\overline{x} = (x, x, \dots, x)$ and unshared biases gives:

$$\hat{f}_{\text{DAC}}(\overline{x}) = \sum_{j=1}^{n} w_{0,j}\, \varphi(b_{0,j} + x), \tag{6}$$

which is a universal approximator of a large class of functions $\mathbb{R} \to \mathbb{R}$, for $n \to \infty$. On the other hand, a standard post-activated unit with replicated inputs would give:

$$\hat{f}_{\text{std}}(\overline{x}) = \varphi\Big(b_0 + \sum_{j=1}^{n} w_{0,j}\, x\Big) = \varphi(b_0 + \tilde{w}_0\, x),$$

equivalent to the same without replicating the inputs, regardless of n. To gain expressivity we can add a hidden layer with n standard units (with or without replicated inputs is the same), obtaining:

$$\hat{f}_{2\times\text{std}}(\overline{x}) = \varphi\Big(b_0 + \sum_{j=1}^{n} w_{0,j}\, \varphi(b_j + \tilde{w}_j\, x)\Big).$$

To show that $\hat{f}_{2\times\text{std}}$ has a representation power similar to $\hat{f}_{\text{DAC}}$, we reparametrize, putting $\tilde{w}_{0,j} = w_{0,j}|\tilde{w}_j|$ and $\tilde{b}_j = {}^{b_j}\!/_{|\tilde{w}_j|}$ in the above expression, which gives:

$$\hat{f}_{2\times\text{std}}(\overline{x}) = \varphi\Big(b_0 + \sum_{j=1}^{n} \tilde{w}_{0,j}\, \varphi\big(\tilde{b}_j + \text{sign}(\tilde{w}_j)\, x\big)\Big).$$

Thus, in this toy problem, one needs a two-layers standard neural network to get a representation power similar to a single pre-activated unit.

Backpropagation. We consider a DAC multi-layer perceptron, and compute the derivatives of a loss E with respect to the parameters. Let y_i, $w_{i,j}$ and $b_{i,j}$ denote the outputs, weights, and DAC biases of layer k, respectively, and let m denote the number of units and $y_j^\diamond$ denote the outputs of layer $k-1$. Then $y_i = \sum_{j=1}^{m} w_{i,j}\, \varphi(b_{i,j} + y_j^\diamond)$, and we get:

$$\begin{cases} \frac{\partial E}{\partial w_{i,j}} = \varphi(b_{i,j} + y_j^\diamond)\, \frac{\partial E}{\partial y_i} \\ \frac{\partial E}{\partial b_{i,j}} = w_{i,j}\, \varphi'(b_{i,j} + y_j^\diamond)\, \frac{\partial E}{\partial y_i} \\ \frac{\partial E}{\partial y_i} = \sum_{l=1}^{n} w_{l,i}^*\, \varphi'(b_{l,i}^* + y_i) \frac{\partial E}{\partial y_l^*} \end{cases} \tag{7}$$

Here n, $w_{l,i}^*$, $b_{l,i}^*$ and y_l^* are the units, the weights, the pre-activation biases and the outputs of layer $k+1$, respectively.

If we were to use post-activated units with shared biases instead, then $\varphi'(b_i^* + y_i)$ in the last equation would not depend on l and could be taken out of the sum. This would result in a single 0-1 factor regulating the entire derivative. Therefore, unshared biases provide a more granular masking of the different contributions to the gradient.

7. Conclusions

This paper is intended as a foundational study on architectures that leverage pre-activation with unshared biases. It provides qualitative arguments and empirical evidence that this choice has measurable advantages and promising potential and that there are situations in which trading some weights for additional biases is more efficient. As a future development, one could investigate if and how DAC units can be integrated on diverse architectures, like for instance mobileNet [28], EfficientNet [33], Transformers [34], ViT [7], generative models [9, 15] or in a reinforcement learning setting like the ones described in [24–27, 32, 35].

Acknowledgements. Many thanks to Rosa Gini for her important intellectual contribution.

References

[1] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning Activation Functions to Improve Deep Neural Networks, 2015. arXiv:1412.6830 [cs, stat]. 3

[2] Rafael Ballester-Ripoll, Enrique G. Paredes, and Renato Pajarola. Sobol tensor trains for global sensitivity analysis. *Reliability Engineering & System Safety*, 183:311–322, 2019. 4

[3] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. pages 1800–1807. IEEE Computer Society, 2017. 3

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), 2016. arXiv:1511.07289 [cs.LG]. 1

[5] CurioSAI. Increasing biases can be more efficient than increasing weights, 2023. https://github.com/CuriosAI/dac-dev. 1, 4

[6] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. RepVGG: Making VGG-Style ConvNets Great Again. pages 13733–13742, 2021. 6

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021*, 2021. 8

[8] Leyuan Fang, Guangyun Liu, Shutao Li, Pedram Ghamisi, and Jón Atli Benediktsson. Hyperspectral Image Classification With Squeeze Multibias Network. *IEEE Transactions on Geoscience and Remote Sensing*, 57(3):1291–1301, 2019. 3

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020. 8

[10] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1319–1327, 2013. 3

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645. Springer, 2016. 1, 3, 6

[13] ISIC. Isic 2019: Skin lesion analysis towards melanoma detection. https://challenge.isic-archive.com, 2019. Accessed: 2023-05-03. 5, 7

[14] Jeremy Howard. Imagenette and Imagewoof datasets, 2019. https://github.com/fastai/imagenette. 5, 7

[15] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014. 8

[16] Diego Klabjan and Mark Harmon. Activation Ensembles for Deep Neural Networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 206–214, 2019. 3

[17] Krizhevsky, A. and Nair, V. and Hinton, G. CIFAR-10 and CIFAR-100 datasets, 2009. https://www.cs.toronto.edu/~kriz/cifar.html. 5

[18] Matthew E. Larkum. Are Dendrites Conceptually Useful? *Neuroscience*, 489:4–14, 2022. 2

[19] Hongyang Li, Wanli Ouyang, and Xiaogang Wang. Multi-bias non-linear activation in deep neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 221–229, 2016. 2, 8

[20] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11966–11976, 2022. 3

[21] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, volume 30, page 3, 2013. 1

[22] Jeffrey C Magee. Dendritic integration of excitatory synaptic input. *Nature Reviews Neuroscience*, 1(3):181–190, 2000. 2

[23] Carlo Metta, Marco Fantozzi, Andrea Papini, Gianluca Amato, Matteo Bergamaschi, Silvia Giulia Galfrè, Alessandro Marchetti, Michelangelo Vegliò, Maurizio Parton, and Francesco Morandin. Increasing biases can be more efficient than increasing weights - extended version. https://arxiv.org/abs/2301.00924v3, 2023. 2, 4, 5, 6

[24] Francesco Morandin, Gianluca Amato, Marco Fantozzi, Rosa Gini, Carlo Metta, and Maurizio Parton. SAI: A sensible artificial intelligence that plays with handicap and targets high scores in 9×9 go. In *ECAI 2020*, volume 325, pages 403–410, 2020. 8

[25] Francesco Morandin, Gianluca Amato, Marco Fantozzi, Rosa Gini, Carlo Metta, and Maurizio Parton. SAI: a Sensible Artificial Intelligence that plays with handicap and targets high scores in 9x9 Go (extended version). AAAI21-RLG workshop, 2021. arXiv:1905.10863 [math.CS]. 8

[26] Francesco Morandin, Gianluca Amato, Rosa Gini, Carlo Metta, Maurizio Parton, and Gian-Carlo Pascutto. SAI a Sensible Artificial Intelligence that plays Go. In *IJCNN*, pages 1–8, 2019. 8

[27] Luca Pasqualini, Maurizio Parton, Francesco Morandin, Gianluca Amato, Rosa Gini, Carlo Metta, Marco Fantozzi, and Alessandro Marchetti. Score vs. winrate in score-based games: which reward for reinforcement learning? In *21st IEEE International Conference on Machine Learning and Applications, ICMLA 2022*, pages 573–578. IEEE, 2022. 8

[28] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 4510–4520, 2018. 8

[29] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. ACM*, 63(12):54–63, 2020. 6

[30] SGEMM. SGEMM GPU Kernel Performance. https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance, 2018. Accessed: 2023-05-03. 4

[31] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2217–2225. PMLR, 2016. 3

[32] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016. 8

[33] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019. 8

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, 2017. 8

[35] David J Wu. Accelerating Self-Play Learning in Go. AAAI20-RLG workshop, 2020. https://arxiv.org/abs/1902.10565. 8