Surround the Nonlinearity: Inserting Foldable Convolutional Autoencoders to Reduce Activation Footprint

Baptiste Rossigneux
Université de Rennes, CEA, LIST, F-91120
Palaiseau, France
baptiste.rossigneux@cea.fr

Inna Kucher
CEA, LIST, F-91120
Palaiseau, France
inna.kucher@cea.fr

Vincent Lorrain
CEA, LIST, F-91120
Palaiseau, France
vincent.lorrain@cea.fr

Emmanuel Casseau
Université de Rennes, CNRS, INRIA
35042 Rennes, France
emmanuel.casseau@irisa.fr

Abstract

Modern deep learning architectures, while highly successful, are characterized by substantial computational and memory demands due to their large number of parameters or the storing of activations. That is why it is hard to adapt a neural network to the constraints of hardware, especially at the edge. This paper presents an investigation into a novel approach for activation compression, which we term 'Projection-based compression on channels' or 'ProChan'. Our method involves interposing projection layers into a pretrained network around the nonlinearity, reducing the channel dimensionality through compression operations and then expanding it back. Our module is made to be then totally fused with the convolutions around it, guaranteeing no overhead, and maximum FLOPs reduction. We studied its absorption of the cost of quantization, to combine the two approaches for footprint reduction. Our findings indicate that the projections likely perform an 'adaptive stretching' operation on the feature space, enabling the preservation of essential information when constrained by dimensional limitations. We also perform an ablation study on the different possible strategies for a stable and quick training, and analyse the interactions with different quantization paradigms, namely PACT for activations and post-training quantization (PTQ) methods for weights.

1. Introduction

Deep learning has achieved tremendous successes across a broad range of applications, especially in computer vision with Convolutional Neural Networks (CNNs). However, the power of deep learning models often comes with the cost of high computational and memory demands due to the large number of parameters involved. As such, model compression has become a critical area of study, with the goal of reducing the size of deep learning models without significantly compromising their performance.

Most existing model compression methods, including pruning, quantization, and knowledge distillation, have their unique strengths and limitations. For instance, pruning methods reduce the size of models by eliminating less important connections, while quantization methods decrease the numerical precision of the weights. Although these methods can achieve considerable model compression, the losses each technique brings make it difficult to combine dimension reduction and quantization.

In this study, our key contribution is a novel approach to activation compression with a new module, ProChan. ProChan consists of a two-step operation: an initial compression of the input through a reduction over channels, followed by an expansion back to the original size. We demonstrate that our module can smoothly be coupled with activation quantization on the nonlinearity, while being entirely foldable, to ensure no overhead. We integrate our module into Resnet18 and Efficientnet-lite, demonstrating the robustness of our approach across a wide range of CNNs.

2. Related works

Most existing works in model reduction focus on parameters reduction, but as [4] puts it, since data movement account for a majority of the energy consumption, limiting data movement through activation compression is key for portability. We will focus on the approach of compressing the activations by adding a trained layer [9] [7] [8]. It is proposed in [9] a projection into a lower-dimensional man-

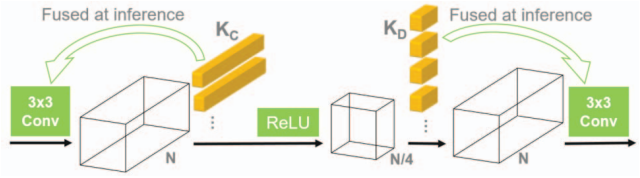


Figure 1: Our method, based on the insertion of an autoencoder around the nonlinearity. K_C compresses the activations to meet the memory requirements and decompresses K_D to adapt the tensor for the next convolution. N here is the initial input depth, so a $\times 4$ compression is applied.

ifold and a quantization that are reversible. [8] improves on this idea by quantizing in mixed precision - since it generally shows the best results in average bits per value [10] [6] - with reinforcement learning, allowing 2.5 bits per value in Resnet18. While the most used metrics are the number of elements and bits per elements, [7] proposed the ceiling compression paradigm, aiming at reducing the maximum memory requirements of a network, by targeting only the necessary layers. [4] also sought to distribute peak memory consumption, but through new architecture design. To the best of our knowledge, all of these approaches are implemented before the ReLU, and our work is the first to propose a compression-decompression paradigm surrounding the ReLU to ensure its foldability, and FLOPs reduction.

Recent advancements in filter pruning also strive to lower the computational and memory demands of CNNs. The hardware-friendliness of these methods lies in avoiding the irregular memory access patterns typically associated with unstructured pruning. Extensive experiments [11] [5] show that filter pruning stands on par with unstructured methods, showing that compression in the channel direction, albeit simple, is a promising research direction.

3. Methodology

3.1. Projection-based compression over channels

The ProChan modules are implemented using a pair of 1x1 convolutions, which serve to compress and decompress the feature maps within the network. The change from a basic CNN structure to one with our module is captured by the equations 1 and 2. The first convolution K_C reduces the number of channels based on a specified ratio, while the second convolution K_D restores the feature map back to its original dimensions. The matching activation function is inserted between the convolutions. The compression ratio of one module is determined by the ceiling compression paradigm from [7]. For example, when we state using ProChan $\times 4$, it means that the maximum size of activations throughout the whole network was reduced by 4. Indeed if an activation was already 2x smaller than the maximum, it would be reduced by 2. It smooths the memory requirements of each layers throughout the network.

$$F_{\text{CNN}} = \text{Conv2}(\text{ReLU}(\text{Conv1}(X))) \quad (1)$$

$$F_{\text{ProChan}} = Conv2(K_D(\text{ReLU}(K_C(\text{Conv1}(X))))) \quad (2)$$

3.2. Fusing our module

Let us consider the previous convolution K_P and the next one K_N with a compression ratio of k, surrounding our module. Contrary to similar works, our ProChan design ensures no computational overhead no matter the compression ratio, and no changes in the layer architecture. The first 1x1 convolution, K_C, having k times fewer filters post-fusion, amounts to kernel pruning, thus promising k times less computation. The second convolution, K_D, reduces even more computation by reducing the depth of the kernel.

K_P has a kernel with size $N_o \times N_i \times C \times C$, respectively for the input size (depth of kernels), the output size (number of kernels) and for the kernel shape (supposed square, for clarity). K_C and K_D each have a set of kernels of size $M \times N_o \times 1 \times 1$ and $N_o \times M \times 1 \times 1$ respectively, M being the compressive fraction of N_o : $M = \frac{N_o}{k}$. After the fusion of K_C, we have a new set of kernels $K_P^{'}$ with size $M \times N_i \times C \times C$, which is calculated as a weighted sum of the previous kernels K_P weighted by one kernel's K_C values:

$$K_P^{'}[m,n,c,c'] = \sum_{i=1}^{N_o} K_P[i,n,c,c'] \cdot K_C[m,i,1,1] \quad (3)$$

$$\forall m \in [1,M], n \in [1,N_i], (c,c') \in [1,C]^2.$$

The fusion of K_D with the next convolution K_N of size $N_o^{'} \times N_o \times C \times C$ operates by weighing the depth of each kernels to get the new kernels of $K_N^{\prime}$ of size $N_o \times M \times C \times C$:

$$K_N^{'}[m,n,c,c'] = \sum_{i=1}^{N_o} K_N[m,i,c,c'] \cdot K_D[i,n,1,1] \quad (4)$$

$$\forall m \in [1,N_o^{'}], n \in [1,M], (c,c') \in [1,C]^2.$$

3.3. Training and testing

We introduce projection layers into a Resnet18 and Efficientnet-lite architectures. This choice serves a twofold purpose - not only to exhibit the effectiveness of our approach on a wide range of CNNs, but also to underscore its capacity to deliver substantial compression results even on highly efficient networks. Efficientnet was chosen in its lite version because the absence of the squeeze-and-excite layers made it more friendly to our technique.

Compression (Resnet18)	Compression rate	Top 1
Original	1×	69.7%
Tai & Al. [9]	2.14×	69.3%
Price & Tanner [7]	2.3×	68.2%
ProChan ×4 (ours)	2.3×	69.0%
ProChan ×8 (ours)	4.2×	67.2%

Table 1: Results of Top 1 accuracies on ImageNet with Resnet18 on ImageNet for state of the art activation compression techniques. The compression rate is the ratio of the total number of activation elements with compression to the total without. The compression rate can not exceed the ceiling as the ceiling represents the maximum compression applied on a layer.

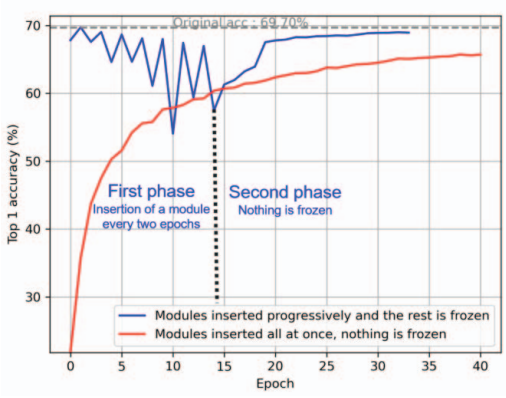


Figure 2: Different training regimes when inserting ProChan ×4 on Resnet18.

3.3.1 Study on training procedures

Every training mentioned in this paper is done on ImageNet [2]. We use Hinton distillation [3] after noting that it slightly accelerates convergence, because it is the simplest and yet most effective one to add to the loss. We perform a study on the usual training methods when adding compression modules for activations. Figure 2 illustrates the differences in training. Since we are also looking for a technique that would not add a significant training overhead, the number of epochs it would take to converge is also taken into account. We limit ourselves to only 2 epochs per module on the first phase (frozen phase) and the same amount of epochs on the second phase (unfrozen phase). We test different hyperparameters and use Adam, with 0.001 learning rate and divide it by 10 during the second phase.

We initially tried to incorporate all projections at once, but this proved unstable due to complexities from concurrent projection layers during training. We began by freezing the entire model except for progressively added projections, establishing this as our baseline. Next, we attempted selective freezing, targeting only layers deeper than the deepest projection. This method, aiming for both stability and adaptability, yielded results similar to full freezing. Lastly, an entirely unfrozen strategy showed comparable performance, even with the model's increased adaptability.

Across these varying strategies, the key insight is that the freezing regime does not significantly impact the final performance, as long as the projections are introduced progressively. This finding suggests a level of robustness in models with projections against different freezing regimes.

Model	Original	ProChan ×4, PACT 2 bits
Effnet-lite0	75.1%	73.0% (-2.1%)
Effnet-lite1	76.7%	74.1% (-2.6%)
Effnet-lite2	77.6%	74.0% (-3.6%)
Effnet-lite4	81.5%	75.1% (-6.4%)

Table 2: Results of Top 1 accuracies on ImageNet with Efficientnet-lite compressed with ProChan ×4 modules.

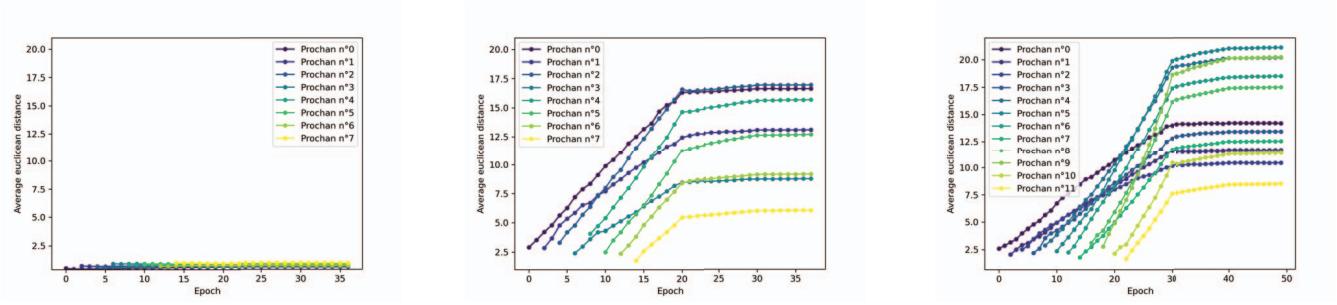
3.3.2 The compatibility of our technique with quantization

We evaluate our method's robustness against weights and activations quantization using state-of-the-art PTQ techniques like FDDA [12], replicating realistic edge inference conditions. Our method maintains accuracy with up to 8-bit quantization but suffers significant loss beyond.

For this reason we opt for a comprehensive approach where we combine our projection-based technique with PACT (Parameterized Clipping Activation) [1] to quantize the activations during the ReLU operation inside our ProChan module. We experiment with reducing precision to 8 bits, 4 bits, and 2 bits. We use our technique combined with PACT and FDDA for weights and remaining activations quantization to 8 bits. As shown in figure 4, training with PACT allows us to compress beyond 8 bits with more accuracy than with a PTQ technique alone. This integrated strategy aims to leverage the strengths of each method.

4. Results

As shown in table 1, our method gets better accuracy than [7] for exactly the same compression rate, and extends the study to 8× compression. Figure 4 shows how our technique can be leveraged in a real low-precision setup, while also showing its limits. Indeed it highlights that PACT can help quantize further (up to 2 bits) where usual quantization methods force a drop in accuracy for 4 bits quantization. For example, our Resnet18 compressed with ProChan ×4 will lose only 0.8% in Top1 accuracy after being com-



(a) ProChan $\times 4$ (b) ProChan $\times 4$ + PACT 2 bits (c) ProChan $\times 8$ + PACT 2 bits

Figure 3: The euclidean difference between inputs and outputs of every ProChan module, for three different training setups, averaged on all the ImageNet validation set. The y-axes were set to highlight the difference in magnitude.

pressed in 8 bits (weights and activations), but in a mixed precision setup, further reducing memory costs by adding PACT 2 bits on already compressed activations will only cost a 1% drop in Top1. However for bigger ProChan compression like $\times 8$, the damages from PACT are amplified.

We analyze Efficientnet-lite to demonstrate our technique's effectiveness on better optimized networks. As detailed in table 2, applying a ProChan $\times 4$ compression, we observe a larger drop in performance, albeit analogous to those encountered with the Resnet18 architecture. While our technique manifests commendable resilience for smaller configurations like Efficientnet-lite0 - a 2.1% drop, it seems to reach a bottleneck with larger models such as the Efficientnet-lite4, suggesting that room exists for further optimizations.

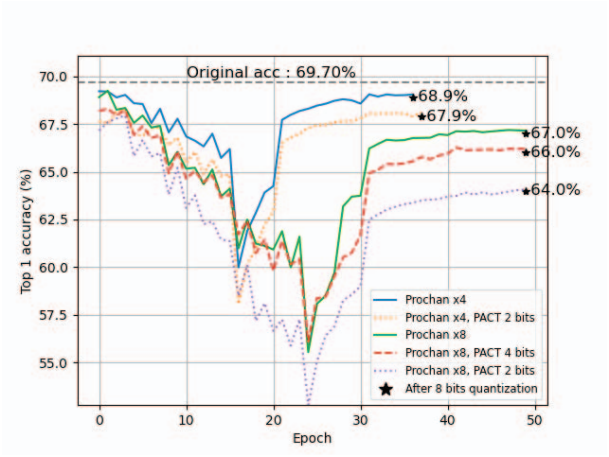


Figure 4: Training of different configurations of ProChan with PACT and FDDA on Resnet18. To show the impact of the different parts of our compression scheme : plain lines are only compressed with ProChan, dashed lines also have 2 or 4 bits activation quantization, and stars are the results of 8 bits quantization of weights and activations on the last checkpoints of the dashed lines with a PTQ algorithm.

5. Discussions

5.1. Theoretical investigations : The stretching hypothesis

We propose the 'Stretching Hypothesis' to explain the functioning of our module with quantization. We theorize that our module, which includes compression and decompression stages, manipulates the feature space to enable efficient learning. The stretch in the feature space assists in disentangling feature representations, simplifying the learning and decision-making processes of coming layers.

Support for our hypothesis comes from our investigation of the Euclidean and Cosine distances between the inputs and outputs of our modules. We observe an increase in the Euclidean distance during training, which stabilizes upon convergence, indicating a 'stretching' operation (figure 3). Meanwhile, the cosine distance remains high and constant, implying the preservation of relative orientation of features by the projections. As shown in figure 3, the more aggressive the compression or quantization, the greater the 'stretching' of the feature space. Indeed, a comparison between (a) and (b) highlights that adding PACT drastically increases the stretching, and between (b) and (c) indicates that lowering the ceiling also increases it. These observations are also verified in Efficientnet-lite.

6. Conclusions and future works

We present a novel method for compressing activations that's foldable and pairs effectively with techniques like PACT activation quantization and PTQ for weights. This method can be effortlessly integrated into hardware. Future efforts will focus on optimizing the balance between compression and quantization for each layer during training. We also aim to utilize our insights from this module to develop improved initializations for faster convergence.

References

[1] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[4] Ji Lin, Wei-Ming Chen, John Cohn, Chuang Gan, and Song Han. Mcunet: Tiny deep learning on iot devices. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[5] Zili Liu, Peisong Wang, and Zaixing Li. More-similar-less-important: Filter pruning via kmeans clustering. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[6] Clément Metz, Thibault Allenet, Johannes Thiele, Antoine Dupret, and Olivier Bichler. Lattice quantization. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–2. IEEE, 2023.

[7] Ilan Price and Jared Tanner. Improved projection learning for lower dimensional feature maps. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[8] Yu-Shan Tai, Cheng-Yang Chang, Chieh-Fang Teng, and An-Yeu Andy Wu. Learnable mixed-precision and dimension reduction co-design for low-storage activation. In *2022 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6. IEEE, 2022.

[9] Yu-Shan Tai, Chieh-Fang Teng, Cheng-Yang Chang, and An-Yeu Andy Wu. Compression-aware projection with greedy dimension reduction for convolutional neural network activations. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 56–60. IEEE, 2022.

[10] Linjie Yang and Qing Jin. Fracbits: Mixed precision quantization via fractional bit-widths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10612–10620, 2021.

[11] Xin Zhang, Weiying Xie, Yunsong Li, Jie Lei, and Qian Du. Filter pruning via learned representation median in the frequency domain. *IEEE Transactions on Cybernetics*, 2021.

[12] Yunshan Zhong, Mingbao Lin, Mengzhao Chen, Ke Li, Yunhang Shen, Fei Chao, Yongjian Wu, and Rongrong Ji. Fine-grained data distribution alignment for post-training quantization. In *European Conference on Computer Vision*, pages 70–86. Springer, 2022.