Increasing CNN Representational Power Using Absolute Cosine Value Regularization

William Singleton
Electrical and Computer Engineering
Purdue School of Engineering and Technology
Indianapolis, USA
william.singleton2010@gmail.com

Mohamed El-Sharkawy
Electrical and Computer Engineering
Purdue School of Engineering and Technology
Indianapolis, USA
melshark@iupui.edu

***Abstract*—The Convolutional Neural Network (CNN) is a mathematical model designed to distill input information into a more useful representation. This distillation process removes information over time through a series of dimensionality reductions, which ultimately, grant the model the ability to resist noise and generalize effectively. However, CNNs often contain elements that are ineffective at contributing towards useful representations. This paper aims at providing a remedy for this problem by introducing Absolute Cosine Value Regularization (ACVR). This is a regularization technique hypothesized to increase the representational power of CNNs by using a Gradient Descent Orthogonalization algorithm to force the vectors that constitute their filters at any given convolutional layer to occupy unique positions in $\mathbb{R}^n$. This method should in theory, lead to a more effective balance between information loss and representational power, ultimately, increasing network performance. The following paper examines the mathematics and intuition behind this Regularizer, as well as its effects on the filters of a low-dimensional CNN.**

***Index Terms*—Absolute Cosine Value Regularization, Convolutional Neural Network, Gradient Descent Orthogonalization**

I. Introduction

Given an effective CNN architecture, it is often the case that the architecture contains representational power that is misused or underutilized. This can be understood through the processes of pruning [1] and network quantization [2]. Pruning is a method through which a Neural Network is reduced in size by selectively removing individual weights, filters, or even entire layers. Quantization on the other hand, is a process through which weights present in a network are given a reduction in precision, corresponding to a lower bit representation. In both cases, elements are entirely removed from the CNN. Rather unexpectedly however, the networks can retain their effectiveness, and in some cases, improve their ability to provide accurate outputs [3]. This process provides evidence that any CNN may have elements that are underutilized, or in extreme cases, entirely useless. While methods such as pruning and quantization appear to be effective in practice, they lead in the direction of a conclusion that is challenging and counterintuitive. By removing elements from a network, representational power is lost. Given this understanding, there must surely be some method through which the network architect can wield this underutilized representational power without excising entire elements.

The search for greater CNN representational power begins with one simple idea: if each filter in a layer learns a similar combination of weights, the information passed to the following layers will be similar as well. However, if each filter learns a weight pattern that is significantly different, the following layers will receive a richer representation, and in turn, be able to produce their own. The goal then, is to learn filter weights that will preserve the greatest amount of information throughout the network, and allow the distillation process the greatest probability of obtaining useful information. This idea can not be taken too literally however, as learning only unique filter values removes any ability of the network to learn from its training data. The search for greater representational power is therefore, a search for an effective balance between filter diversity, and data generated knowledge. This regularization process, unlike pruning and quantization, does not remove potentially useful elements, but instead, seeks to redistribute them into a diverse space where they can begin to provide significant value.

The diversity of any two filters in this case, is given by their respective Absolute Cosine Value given in 1:

$$\left|Cos(\theta)\right| = \left|\frac{x \cdot y}{\|x\|\|y\|}\right| \tag{1}$$

This number is mathematically convenient to represent as an addition to the loss function, as well as intuitive in the fact that it is a representation of the similarity of the vector spaces of each filter. Understanding the problem of underutilized network elements, and the belief that novel filters provide richer representations, this paper aims at providing evidence that forcing CNN filters to learn weights to position themselves into unique spaces in $\mathbb{R}^n$, may improve their ability to generalize well to new data, and increase their representational power.

II. Related work

The concept of increasing representational power by enforcing diversities in Support Vector Machines using the Cosine Formula was discussed in [4]. In addition, regularizing filter diversities for the purpose of increasing network accuracy in CNNs is discussed by [5]. However, this publication is chiefly concerned with constructing smaller architectures through pruning methods, and combined the activity of diversity regularization in an ensemble with other regularization terms. A rigorous explanation for the existence of the Regularizer, its operation, and implementation, is not given.

III. Mathematical Description of ACVR

In order to understand Absolute Cosine Value Regularization (ACVR), it is necessary to define its implementation. Equation 2 represents the total network loss function with the addition of the regularization term ACV, which is a function of the weights W, and summed over the layers in which it is added. This term is

This is the author's manuscript of the article published in final edited form as:

Singleton, W., & El-Sharkawy, M. (2020). Increasing CNN Representational Power Using Absolute Cosine Value Regularization. 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), 391–394. https://doi.org/10.1109/MWSCAS48704.2020.9184503

returning a value which corresponds to the similarity of the filter vector spaces at each layer, by minimizing it, the system will be attempting to generate high-diversity filter vectors. The scalar gamma, represents the hyper-parameter that adjusts the contribution of the Regularizer, a factor which scales the ratio of filter diversity to data generated knowledge.

$$L(\hat{y}) = \sum_{(\hat{y},y)} loss(f(\hat{y}, W), y) + \gamma * \sum_{(l \in L)} ACV(W^l) \tag{2}$$

This formula appears simple to implement, however care must be given to its implementation as the Regularizer must compare all filters against one-another at each layer. This results in a N^2 matrix for every regularized layer. However, since this comparison matrix is symmetric, and the filters need not be compared against themselves, the portion of the ACV matrix containing relevant information is upper-triangular. This leaves the number of ACV calculations done at each layer to be given by 3:

$$comparisons_l = \frac{N_l^2 - N_l}{2} \tag{3}$$

This formula states that this algorithm will become increasingly time consuming with large values of N, necessitating an efficient vectorized computation method.

A. Calculating Regularizer's Contribution to Loss Function

Each convolutional layer contains N convolutional filters parameterized by their height, width, and number of channels. This leaves a 4-Dimensional Tensor that can be unrolled for each filter into a 2-Dimensional tensor for efficient computation as follows in 4:

Unrolled 4-Dimensional Tensor $\equiv$ X, $each filter \equiv$ f$_i$

$$\frac{X \cdot X^T}{\|X\| \cdot \|X\|^T} \equiv \frac{\begin{bmatrix} - & f_1 & - \\ - & f_2 & - \\ & \vdots & \\ - & f_N & - \end{bmatrix} \begin{bmatrix} | & | & | & | \\ f_1 & f_2 & \cdots & f_N \\ | & | & | & | \end{bmatrix}}{\begin{bmatrix} \|f_1\| \\ \|f_2\| \\ \vdots \\ \|f_N\| \end{bmatrix} \begin{bmatrix} \|f_1\| & \|f_2\| & \cdots & \|f_N\| \end{bmatrix}} \tag{4}$$

$$\frac{\begin{bmatrix} f_1 f_1 & \cdots & f_1 f_N \\ \vdots & \ddots & \vdots \\ f_1 f_N & \cdots & f_N f_N \end{bmatrix}}{\begin{bmatrix} \|f_1\|\|f_1\| & \cdots & \|f_1\|\|f_N\| \\ \vdots & \ddots & \vdots \\ \|f_1\|\|f_N\| & \cdots & \|f_N\|\|f_N\| \end{bmatrix}} \tag{5}$$

$$\begin{bmatrix} \frac{f_1 f_1}{\|f_1\|\|f_1\|} & \cdots & \frac{f_1 f_1}{\|f_1\|\|f_N\|} \\ \vdots & \ddots & \vdots \\ \frac{f_1 f_N}{\|f_1\|\|f_N\|} & \cdots & \frac{f_N f_N}{\|f_N\|\|f_N\|} \end{bmatrix} \tag{6}$$

$$Cos(\theta) = \frac{x \cdot y}{\|x\|\|y\|} \tag{7}$$

After element-wise division in 5, each cell in 6 is identical to the Cosine Formula 7, for respective vectors. It is important to note that the Cosine Formula for two identical vectors does not provide any information, as it always yields a value of one. This means all values along the main diagonal are unimportant. Values below the main diagonal are also unimportant, as they provide the same information as the values above the main diagonal. The only useful information in matrix 6 is contained above the main diagonal, as such, all values below are set to zero in 8:

$$\begin{bmatrix} 0 & \cdots & \frac{f_1 f_1}{\|f_1\|\|f_N\|} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \tag{8}$$

This leaves a final Tensor containing useful information which is summed and multiplied by the hyper parameter gamma shown in 9:

$$ACV(W^l) = \gamma * \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \left| \frac{f_j f_i}{\|f_j\|\|f_i\|} \right| \tag{9}$$

Equation 9 represents the loss returned for each regularized layer, and is a singular instance of the ACV function from 2.

B. Expansion of Partial Derivative For Back-propagation

The following steps demonstrate the Calculus involved in determining a closed form solution to the partial derivative of the loss function with respect to a given weight in f_1, from the perspective of the ACV Regularizer:

$$ACV(W^l) = \gamma * \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \sqrt{\left(\frac{f_j f_i}{\|f_j\|\|f_i\|} \right)^2} \tag{10}$$

Definition of Absolute Value

$$\frac{\partial ACV(W^l)}{\partial f_{1k}} = \gamma * \frac{\partial}{\partial f_{1k}} \sum_{i=2}^{N} \sqrt{\left(\frac{f_1 f_i}{\|f_1\|\|f_i\|} \right)^2} \tag{11}$$

Partial Derivative of Loss Function With Respect to k_{th} Element of Vector f_1

$$\frac{\partial ACV(W^l)}{\partial f_{1k}} = \gamma * \frac{\partial}{\partial f_{1k}} \sum_{i=2}^{N} \sqrt{\left(\frac{\sum f_1 f_i \sum f_1 f_i}{\sum f_1^2 \sum f_i^2} \right)} \tag{12}$$

Expanding Definition of ACV

$$\frac{\partial ACV(W^l)}{\partial f_{1k}} = \gamma * \sum_{i=2}^{N} \left[\frac{f_{ik} \sum f_1 f_i}{\sqrt{\left(\sum f_1 f_i \sum f_1 f_i \right)} \sqrt{\left(\sum f_1^2 \sum f_i^2 \right)}} - \frac{f_{1k} \sum f_i^2 \sqrt{\left(\sum f_1 f_i \sum f_1 f_i \right)}}{\sqrt{\left(\sum f_1^2 \sum f_i^2 \right)^3}} \right] \tag{13}$$

Final Analytic Expression

C. Discussion of Analytic Solution

Equation 13 is representative of the rate of change of the loss function from the perspective of the ACV Regularizer, with respect to the vector element f_{1k}. In equation 13 any index of f_1 can be substituted for, and the complete contribution of the Regularizer to the loss function for f_1 can be found by solving 13, for all elements k. Equation 13 by itself is a large multi-variable expression, however its form yields insight into potential operation of the Regularizer. The sign of the right hand side of the equation is given solely by the sign of f_{1k}, and in particular, it is opposite to the sign of f_{1k}. This

signifies that the right hand side of the equation is actively producing a value that is attempting to minimize the value of f_{1k}, and will make this value $N-1$ times for each instance in 13. The significance of this is that in addition to attempting to explicitly push all vectors into unique positions, the ACV Regularizer may be implicitly working to suppress their L_1 Magnitudes, further stabilizing the network.

IV. Experiments on Low-Dimensional CNN

The ACV Regularizer aims at forcing filter vectors into unique positions in $\mathbb{R}^n$. The success of this objective can be easily visualized in $\mathbb{R}^2$ or $\mathbb{R}^3$. Therefore, in order to verify the effectiveness of this Regularizer, a trial CNN is constructed that excluding the fully connected base, consists of two convolutional layers. The purpose of this CNN is not to achieve competitive accuracies, but to demonstrate that by applying ACVR at high gamma values, the filter vectors within the network will spread apart from each other in a manner that is near maximal, a result of minimizing the loss function given in 2. In this case, the differences between vectors is given by 7. Two experiments are conducted: one with the purpose of visualizing the movements of the CNN filter vectors with, and without regularization. The second, to determine the Regularizer's effect on the network when trained to validation set convergence. The filters of the network consist of only three elements and therefore exist in $\mathbb{R}^3$, rendering visualization of the Regularizer's activity trivial.

A. Network Architecture and Training Conditions

The architecture of the network devised to test the ACV Regularizer consists of two convolutional layers, consisting of three and five filters respectively, and one fully-connected layer. The dataset used is the CIFAR-10 small images dataset [6], with data set augmentation consisting of horizontal transfer, and random width and height shifting. The model is constructed using the Keras API [7], and the Regularizer is developed from the TensorFlow API [8]. Post processing is done using SciPy [9] and MatLab [10].

B. Visualizing Effects Of Regularization

To determine the effectiveness of the ACV Regularizer, the network is run with a batch size of 32 for 8 epochs with, and without its presence. The gamma value for the Regularizer is found empirically, and is determined to be $5.0 * 10^{-1}$. From the network, the initial filter vectors are saved, along with the vectors from the network after training in both instances. To determine vector similarity, the Cosine Formula 7 is employed, and six sets of similarity matrices are produced. The following three tables (I, II, III) display the data provided by the second convolutional layer filters:

filters	K2	K3	K4	K5
K1	0.749633	-0.524069	0.501112	-0.985155
K2	-	-0.951118	-0.191307	-0.8513
K3	-	-	0.445442	0.657606
K4	-	-	-	-0.345152

TABLE I
Cosine Value Comparison of Filters Before Training

filters	K2	K3	K4	K5
K1	-0.780891	-0.989761	-0.121355	0.789708
K2	-	0.743862	0.70981	-0.989937
K3	-	-	0.0573636	-0.772605
K4	-	-	-	-0.666673

TABLE II
Cosine Value Comparison of Filters After Training Without ACVR

filters	K2	K3	K4	K5
K1	-0.000174307	0.000104767	-1.35988e-05	-1
K2	-	-1	-0.000178602	0.000214833
K3	-	-	0.000311102	-0.000145298
K4	-	-	-	-0.000135528

TABLE III
Cosine Value Comparison of Filters After Training With ACVR

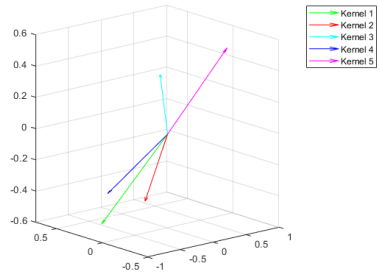

Fig. 1. Convolutional Layer Two Filters Before Training

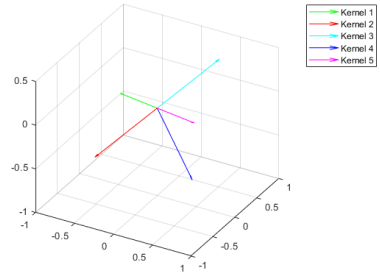

Fig. 2. Convolutional Layer Two Filters After Training With ACVR

C. Discussion of Experiment One

The goal of the first experiment is to demonstrate that when the activity of the Regularizer dominates the loss function, it is capable of spreading the filter vectors out in a near maximal manner. The matrix of cosine values is displayed to provide a numerical representation of the diversity among filter vectors. The filter vectors as they are initialized can be seen in Fig. 1, along with their cosine matrix in Table I. In the case of the non-regularized model, the filter vectors numerically maintain a large degree of similarity after training, as seen in Table II. In the case of the Regularized model seen in Fig. 2, the filters of the convolutional layer visually spread apart in a near maximal manner, which is confirmed by the cosine value matrix in Table III. It is clear from the experiments that the ACV Regularizer is capable of providing immense vector diversity, whereas the non-regularized model does not. The behavior exhibited by the Regularizer is exactly that which was hoped, lending substantial evidence to its correct theory and implementation.

D. Long-Term Behavior of Regularization

The second experiment is run under the same conditions as the first, for 150 epochs, with the purpose of determining the long-term behavior of the ACV Regularizer on the model. From these trials, the training and validation set accuracies of both models are obtained, as well as the L_1 Magnitudes and net sum of the ACV values from each of the convolutional layers.

E. Discussion of Experiment Two

The second experiment is designed to evaluate the behavior of the model when trained to validation set convergence. While the

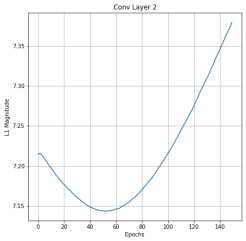

Fig. 3. L_1 Magnitudes of Convolutional Layer Two Without ACVR

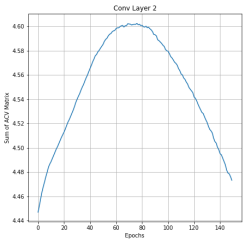

Fig. 4. Sum of ACV Values of Convolutional Layer Two Without ACVR

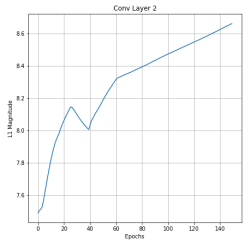

Fig. 5. L_1 Magnitudes of Convolutional Layer Two With ACVR

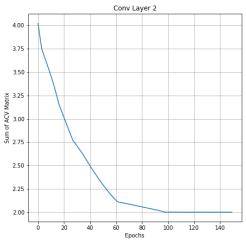

Fig. 6. Sum of ACV Values of Convolutional Layer Two With ACVR

ACV Regularizer was hypothesized to provide an L_1 Regularization effect, the evidence provided from Fig. 3 and Fig. 5 is inconclusive. This can be clarified from equation 13: it is understood that the right ratio is attempting to provide an L_1 Regularization effect, but little can be predicted regarding the left ratio. Given this mathematical definition, and the inconclusive nature of the evidence provided, it can be said that ACVR can not be guaranteed to provide any L_1 Regularization effect. The graphs in Fig. 4 and Fig. 6 demonstrating the sum of ACV values, align exactly with that which was determined from Experiment One, as the regularized model approaches the minimum possible ACV value given the number of filters in each layer, whereas the non-regularized model maintains a significantly higher degree of vector similarity. Overall, the evidence gathered is an excellent proof of concept, and the effect of the Regularizer on a full-scale CNN remains an open research question.

V. Conclusion

Absolute Cosine Value Regularization is a method designed to employ the underutilized elements in a CNN, and promote optimal information distillation. This paper has discussed the mathematics behind this regularization technique, and described its practical implementation. Furthermore, the first experiment conducted on a low-dimensional CNN has provided evidence that this Regularizer is capable of generating high diversity filter vectors. The second experiment confirmed this result, but also produced evidence that the ability of ACV Regularization to act as an L_1 Regularizer is not guaranteed. The evidence gathered concerning filter diversity is promising, as it provides both visual and numerical data demonstrating the effectiveness of the implementation, and reinforces the veracity of the theory. It is hypothesized that at optimum values of gamma, the inclusion of this Regularizer in a full-scale CNN with many layers, and filters per-layer, such as VGG-19 [11], will increase its representational power. In addition, this Regularizer is hypothesized to be of even greater benefit to CNNs that are targeted towards mobile applications, such as MobileNet [12].

VI. Acknowledgment

We thank the ECE department, Purdue School of Engineering and Technology, and our colleagues from IoT Collaboratory lab.

References

[1] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2016. last accessed on 3/21/2020.

[2] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018. last accessed on 3/21/2020.

[3] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," vol. 89, NIPS, 1989.

[4] Y. Yu, Y. Li, and Z. Zhou, "Diversity regularized machine," pp. "1603–1608", IJCAI, 2011.

[5] T. Wang, L. Fan, and H. Wang, "Simultaneously learning architectures and features of deep neural networks," 2019. last accessed on 3/21/2020.

[6] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

[7] F. Chollet *et al.*, "Keras." https://keras.io, 2015. last accessed on 3/21/2020.

[8] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org, last accessed on 3/21/2020.

[9] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, I. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, 2020.

[10] MATLAB, *9.6.0.1072779 (R2019a)*. Natick, Massachusetts: The MathWorks Inc., 2019.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. last accessed on 3/21/2020.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. last accessed on 3/21/2020.