DEFCON 1: Deformable Convolution in 1D for Channel Attention

Vatsal Agarwal Harish Kumar Madhava Paliyam Jerry Qian
University of Maryland, College Park
{vatsalag,hkumar,mpaliyam,jerryq}@umd.edu

Abstract

Attention mechanisms for convolutional neural networks have shown improvement in performance on various tasks. Previous papers have explored methods for spatial attention. In this paper, we present DEFCON-1, an attention module for channel attention. Different objects activate varying sets of channels. We attempt to improve performance by weighing each channel of feature maps by its relevance to the object. We show how deformable 1D convolutions can increase performance against baseline ResNet-50 and compare against existing channel attention modules.

1. Introduction

Deep convolutional neural networks (CNNs) have become a staple in tackling computer vision tasks like image classification and object segmentation, and much work has been devoted to improve their efficiency and performance. Channel attention is one such enhancement that has been explored in recent years. Similar to human tendency to focus on certain features in a scene (such as a face in a portrait), channel attention weighs channels in a CNN feature map by their relevance to the task. Integrating channel attention mechanisms into convolutional blocks and modules have shown great promise in improving the performance of CNNs in various tasks; examples include squeeze-and-excitation network (SENet) [4], and Efficient Channel Attention network [7].

Our proposed method, DEFCON 1, hopes to increase non-local interactions between channels, like in SE-Net, while also keeping a low number of parameters like ECA-Net. To accomplish this, we decided to combine deformable convolutions with the 1D kernel operation from ECA-Net; we also perform channel shuffling to increase detection of non-local interactions.

In summary, our contributions in this paper are:

1. We present a module that uses 1D deformable convolutions and channel shuffling to perform channel attention that captures local and non-local interactions.
2. We review the results of experiments we conducted to evaluate the performance of DEFCON 1 in comparison to other channel attention mechanisms. In addition, we explain our approach to fine-tune hyperparameters.
3. We consider future directions by which to improve the performance and evaluation of DEFCON 1.

2. Related Works

In this section, we discuss previous research that our work takes inspiration from, and the lessons we took from each.

SENet [4] In this paper, a Squeeze-and-Excitation (SE) module is introduced to explicitly model channel interdependencies and adaptively weigh channels. An output feature map is squeezed using Global Average Pooling (GAP) to an one-dimensional feature vector; this is input to a two-layer fully connected excitation network that outputs a vector of channel weights of the same size. This output is used to scale the channels of the original feature map. SENet, when integrated into traditional ResNet architectures, offers greater accuracy, while only slightly increasing computational cost; ResNet-50 with SENet modules achieves nearly the same accuracy as a baseline ResNet-101. We sought to reduce the parameter overhead and eliminate the dimensionality reduction that comes with SENet with our approach; after all, a two-layer excitation network is trained in each SE module.

ECA-Net [7] In an effort to avoid dimensionality reduction from SENet, the Efficient Channel Attention (ECA) module uses an adaptive kernel that captures cross-channel interactions. Rather than input the squeezed channel vector into a fully-connected network, the ECA module uses the kernel to perform a 1D convolution. This results in a significant reduction in parameter overhead, as there is only a single kernel parameter for each module. This network (combined with ResNet) achieves gains in accuracy over ResNet and SENet architectures, while keeping the number of parameters to be almost equivalent to that of ResNet. Since ECA-Net's implementation only allows for local channel interactions to be captured, we asked ourselves a question:

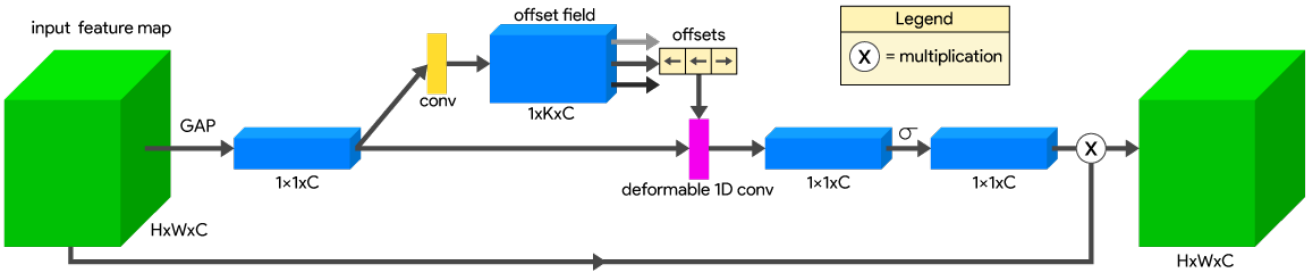

Figure 1. Architecture of our module

What accuracy and computational cost can we expect when considering local and non-local channel interactions?

Convolutional Block Attention Module [8] To incorporate attention along the channel dimension and spatial dimention, CBAM used channel and spatial attention sequentially. First, the input feature is max pooled and average pooled along the channel dimension. Both these tensors are fed through a shared multi layer perceptron similar to SE-Net, and the two outputs are summed and multiplied along the original feature vector to get the attended features. CBAM then applies spatial attention to the channel attended features as well.

Shuffle Net [9] Shuffle Net was created to be very efficient network to be used on mobile devices. On these small networks, they introduced a shuffle net unit block which was used in their networks. Normal outputs for channel wise group convolutions are only related to the input channels near or in the same group. To increase channel interactions, the shuffle net unit incorporated a channel shuffle operation which increased the non-local interactions between channels. We experiment with using the channel shuffle operation with our deformable module to increase non-local interactions.

Deformable Convolutional Network [1] Traditional CNNs are restricted to using fixed geometric structures as kernels (i.e. rectangular grids). The contributions of this paper include introducing deformable convolutional kernels, the spatial sampling locations of which are determined using learnable offsets. The result is free form deformation of the sampling grid, which when incorporated into CNNs leads to better performance and more robustness to spatial transformations. The insights from this paper were helpful in applying deformable convolutions to channel attention, as we seek to capture both local and non-local interactions.

3. Approach

We present a channel attention module that applies weights to feature maps that brings relevant channels into focus, while lessening the impact of irrelevant channels.

The module (Figure 1) begins by running GAP on the input feature map to convert it into a 1D tensor of channel values ($1x1xC$). Next, it calculates channel offsets by running a regular 1D convolution on the pooled 1D tensor. These offset values are passed on to a deformable 1D convolution. With the offsets, the deformable 1D convolution is run on the original pooled 1D tensor to produce another 1D tensor of the same dimensionality. Next, it is normalized with the sigmoid function to produce the channel attention weight tensor. Finally, the weight tensor is multiplied with our input feature map resulting in a feature map with channels weighted by learnt attention. The deformable convolution is described in more detail next.

1-Dimensional Deformable Convolution We start with the offsets that are created using a 1D convolution. A 1xk kernel, with k = 3, is defined as:

$$R = \{-1, 0, 1\}$$

In our case, k was determined the same way [7] determined the kernel sizes for their 1D convolutions. This was through an function on the number of input channels.

In a normal convolution, each output feature y_0 is defined by

$$y_0 = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n)$$

where p_n is each item in R, x is the input feature map (the average pooled tensor in our case), p_0 is the location of the center of the kernel in the feature map, and $w(p_n)$ represents the weight learned by the kernel at position p_n.

In a deformable 1D convolution, the output feature equation is modified by Δp_n which is added when calculating the position for each convolution. Δp_n can be between 0 and the channel size. The equation now becomes:

$$y_0 = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n + \Delta p_n)$$

For example, if we were on the first position in the convolution, $p_0 = 1$, and the convolution was calculating the output for a 1x3 kernel, and the determined offsets were $\Delta p_n = \{3, 5, 20\}$ then the convolution positions would be $\{3, 6, 22\}$ instead of $\{0, 1, 2\}$ under a normal convolution.

Since Δp_n can be non discrete values since it is an output of the previous 1x3 kernel, the values at the new calculated

Method	Backbone	Top-1	Top-5	Parameters
ResNet-50 [3]	ResNet-50	67.060[1]	-	23,705,252
SENet [4]		75.18	92.59	26,220,196
ECANet [7]		74.68	92.65	23,705,332
DEFCON 1 (ECA Hyperparameters)		66.79	88.12	23,705,756
DEFCON 1 (Cosine Annealing with Warm Restarts)		70.3500	89.6200	23,705,756
DEFCON 1 + Channel Shuffle (ECA Hyperparameters)		67.3900	88.4800	23,705,756

Table 1. Comparison of various attention mechanisms with our DEFCON 1 approach. Results from the baseline ResNet-50 are also included.

offset positions x($p_0 + p_n + \Delta p_n$) are linearly interpolated based on which channel values they fall between. Unlike spatial features however, channel features are not known to be continuous and correlated to each other so interpolating might not seem like the optimal solution. However, we hope that by interpolating from the start of training, the channels will be learnt such that they are closely correlated where interpolation would make sense.

y_0 describes the output features of the deformable convolution at position 0. After the deformable convolution operation, y will be the same size as the input average pooled channel tensor and will multiplied with the original feature map, resulting in the attended channel features. We hope that by using deformable convolutions, the dynamic offsets can learn to look at channels that are not close and make better predictions of how much weight to give itself.

Implementation Details We implemented our deformable attention module in Pytorch using a ResNet-50 [3] architecture. The module was added in each block of the model after the convolutions and right before the residual connection, similar to how other channel attention modules are used. Our code is available here.

All of our experiments are using the CIFAR-100 dataset [5]. CIFAR-100 is composed of 32 x 32 images for 100 classes, with 500 training images and 100 test images each. We use the original train test split of the CIFAR dataset with 40,000 training images and 10,000 test images. At training time we augment our training samples by normalizing to the mean and standard deviation of the dataset, applying random crops, and random horizontal flips. We trained our model on a NVIDIA V100 GPU using Google Cloud Compute. We chose this dataset instead of a larger one like ImageNet because of memory, compute and cost limitations.

4. Experiments and Results

To test our channel attention module, we used a ResNet-50 network [3]. We modified the ResNet-50 to make it compatible with the smaller CIFAR-100 images by reducing the first convolution stride to 2. We tested our channel attention module by comparing our validation accuracy against ECA-Net [7], and SE-Net [4]. For ECA-Net and SE-Net, we used the hyperparameters as described by [2]. These were, 200 epochs, Batch-size: 128, Momentum: 0.9, Weight Decay: 5e-4, Learning Rate: 0.1. For comparisons against plain ResNet-50 without any channel attention modules, we used the ResNet-50 benchmark website[1]. Our results are compared in Table 1.

4.1. Local Interactions

Our first experiments were with only local interactions between the channels. Though we were using deformable convolutions, which could learn offsets that were potentially large on our average pooled $1x1xC$ tensor, they were being determined by a $1xk_i$ convolution where k_i was the size of the kernel. This meant that only the local channel features were being used in determining those offsets.

4.2. Non-local Interactions

We experimented with two types of ways to include non-local channel interactions while determining offsets for the deformable module. The first was by using a CxC matrix, where C was the average pooled channel vector, similar to [**?**]. We then chose the k highest correlated channels to determine offsets. Unfortunately, we ran these models on a different backbone network and therefore are not comparable with the rest of our results. We hope to continue exploring using the CxC matrix in the future.

The second was by using a channel shuffle [9] before each offset generation. In these experiments we shuffle the channels by groups as determined by the equation, where k is the kernel size:

$$2^{\lceil \log_2(k) \rceil}$$

We hope this shuffle operation will add non-local interactions between the channels by placing channels that are far apart closer together before the offset generation using 1D convolution. We saw that adding channel shuffle slightly improved our results, but more hyperparameter testing is required.

[1] https://paperswithcode.com/paper/resnet50-on-cifar-100-without-transfer

4.3. Weights

We also tried weighting the local and non-local parts of the channel attention by using a learnable parameter. Unfortunately, due to time constraints we did not have time to experiment with these changes as much.

4.4. Hyperparameter Optimization

To determine optimal hyperparameters for our module, we tested different types of schedulers. As our initial baseline, we started with the same hyperparameters as ECA used which were stochastic gradient descent with weight decay of 1e-4, momentum of .9, and mini batch size of 256. Unfortunately, these hyperparameters led to overfitting for our model, so we decided to experiment with different learning rate schedulers.

First, we reduced the batch size to 128 and set the initial learning rate at a lower value of .01. Then we tried cosine annealing scheduler as proposed in [6] and implemented in pytorch. This learning rate follows a cosine curve, increasing and decreasing from the initial learning rate to 0 and back to the initial learning rate over a defined period. We also tried cosine annealing with warmup restarts which is similar to cosine annealing except the period of the restart increases by a multiplier after each restart. Lastly, we tried reduce on plateau which reduces the learning rate once it starts to plateau for a given number of epochs. Due to time constraints we were unable to solve the over fitting problem that we were experiencing but our best model that we have trained so far reaches 70% top 1 accuracy. This was reached using the cosine annealing with warm up restarts with the number of epochs between restarts initially set to 50, and multiplied by 2 after each restart. We trained this model for 400 epochs. Our best results using each method is shown in Table 1. We hope to resolve the over fitting issue in the future.

5. Conclusion

In this paper, we apply 1 Dimensional deformable convolutions to determine channel wise attention weights. Our method experimented with multiple methods for offset generation and applies those offsets to the convolution on the channel wise average pooled values. The outputs from the convolution are multiplied onto the original feature map in a ResNet-50 network. While we were unable to optimize our hyperparameters to get optimal results, we saw that the deformable convolutions were able to achieve close to the performance by other state of the art channel attention mechanisms.

To further optimize our results we will try a few different experiments. First, we will look into different methods of determining offsets instead of a 1D convolution on the average pooled channel features. In addition, we will try methodical optimization of hyperparameters to better find optimal model to reduce the quick overfitting on the training data that we had observed. Lastly, once we have a optimal model, we would like to compare our channel activations with the other channel attention approaches to examine if there are any differences we can notice.

References

[1] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017. 2

[2] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[4] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CVPR 2018*, Jun 2018. 1, 3

[5] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 3

[6] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. 4

[7] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks, 2020. 1, 2, 3

[8] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. *CoRR*, abs/1807.06521, 2018. 2

[9] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017. 2, 3