Group equivariant neural networks

October 1, 2022

Benigno Ansanelli

Abstract

Group equivariant neural network are explored, strengths and shortcomings are investigated with tests on various datasets. A new a method for improving the performance of the G-CNN is tested and seems to work in some cases. All code is publicly available at https://github.com/itsbenigno/GrouPy_PyTorch_experiments.

1. Introduction

Convolutional neural networks have proven to be very powerful models of sensory data. A large amount of empirical evidence supports the notion that convolutional weight sharing is important for good predictive performance. Convolutional weight sharing is effective because there is a translation symmetry in most perception tasks. Although convolutions are equivariant to translation, they are not equivariant to reflection or rotation.
It may therefore be useful to encode a form of rotational symmetry in the architecture of a neural network. This could reduce the redundancy of learning to detect the same patterns in different orientations (freeing up model capacity) and the need for extensive data augmentation.
Cohen's paper shows how convolutional networks can be generalized to exploit larger groups of symmetries, including rotations and reflections.
The goal of this project is to reproduce (using Pytorch) the original experiments made by Cohen, highlighting the strengths and investigating the shortcomings of G-CNN, and also to propose a new approach that may help to cope with G-CNN weakness.

2. Related work

Presented at the same conference, (Dieleman et al., 2016) work is really similar to (Cohen & Welling, 2016), although it does not provide theoretically grounded formalism, so it may be harder to generalize the concept to more symmetries.
(Veeling et al., 2018) Exploits G-CNN for Pathology detection, since histopathology images are inherently symmetric under rotation and reflection.
(Lengyel & van Gemert, 2021) Investigate the filter parameters learned by GConvs and find certain conditions under which they become highly redundant.

3. Method

All the experiments were performed on my laptop (I was not able to install GrouPy on Google Colab). All the architectures used in the original experiments were implemented in Pytorch, with the same layers, optimization method, learning rate and so on. Only the number of epochs was reduced. Fixed seeds were used to guarantee reproducibility. The ResNet44 implementation was taken by PyTorch source code, because it was more close to the one described in the ResNet paper.
An early stopping mechanism for overfitting was used in every test with a patience of 5 epochs (if the model performance on validation set doesn't improve for more than 5 epochs, stop the training and pick the model with the best accuracy on validation set). I tried to do a comparison between Z2 (plain CNN), P4 and P4M in every test.
To test the claim that "convolution layers can be used as a drop-in replacement of standard convolutions that consistently improves the results" I used a simple model (LeNet) on MNIST dataset, and just substituted the convolutional layers. CNN and G-CNN models had the same number of parameters, obtained as described in (Cohen & Welling, 2016).
In some dataset, rotating a class in a certain orientation actually changes the class (e.g. in MNIST dataset, when we rotate the number 6 of 180 degrees it become a 9), so I checked if stopping the equivariance in the final layers could lead to an improvement.

Group	Param.	Test Error (%)
Z2	216k	1.25
P4	216k	1.1
P4M	216k	1.4

Table 1. Error rates of LeNet on MNIST

Network	Group	Param.	CIFAR10
ALLCNN	Z2	1.37M	20.34
	P4	1.37M	-
	P4M	1.22M	-
ResNet44	Z2	661k	21.64
	P4	660k	18.41
	P4M	660k	-

Table 2. Test error of original experiments

3.1. Implementation details

Cohen created GroupPy (an implementation of G-CNN in Chainer), in such a way that porting to other DL frameworks is relatively easy. All the indexing arithmetic and group functions are already implemented in a framework agnostic way. I used Adam Bielski's PyTorch implementation of GroupPy, since I checked its correctness, and has already been used in a peer reviewed paper.

4. Experimental results

Substituting the 2 layers of convolution in LeNet improved the performance using the P4 G-CNN, but P4M group actually did worse than standard CNN. (1)
I was not able to reproduce all of the original experiments, in particular the ALLCNN network with G-CNN was not feasible to train using my setup. A trend I noticed is that the more the group is big, the more the training is slow, indeed I was able to train ResNet on group P4, but not on P4M. The results on the ResNet are much worse than the original paper because of the reduced number of epochs, still, there is a clear improvement using the group P4. (2)
In (3) the CNN of group P4 had the worst performance, the plain convolution did a better job, and stopping the equivariance in the last layer, in both datasets and networks, was slightly better than the plain convolution.

Network	Param.	Dataset	Test error (%)
PlainCNN	21k	MNIST	0.81
P4CNN	24k	MNIST	1,78
earlyP4CNN	25k	MNIST	0.67
EPlainCNN	111k	EMNIST	6.93
EP4CNN	108k	EMNIST	13.11
EearlyP4CNN	111k	EMNIST	6.40

Table 3. Test on stopping early group equivariance

5. Discussion

Substituting a convolutional layer with a G-CNN one improves the performance of a model, but there is a catch, actually more than one. First of all, the use of P4 (resp. P4M) group increases the size of the feature maps 4 (resp. 8) fold. This means that to maintain the same number of parameters, we have to half (resp. divide by $\sqrt{8}$) the number of filters in each layer, this causes problems when the number of filters is already small. It also limits the deepness of the network (in favour of the width). Another collateral effect is that the training takes longer, even with the same number of parameters, so more epochs have to be done. This, together with the reduced number of filters, is the reason that caused the poor performance in (3). This also happened with P4M in (1).
Stopping earlier group equivariance (by substituting just the first layers of convolution with G-CNN) seems to work, the training is faster and the network can go deeper. The idea is that we don't want to be equivariant in certain datasets. In the results (3) I noticed a general improvement, not only in classes that change when rotated (like 6 and 9). All the tests confirmed that the models benefits from symmetries also at low level, so even if a dataset is not symmetric (in CIFAR10 there are no images of a rotated horse) the performances improve. Overall I think that with the proper tuning, and a solution for the width, e.g. (Lengyel & van Gemert, 2021) the G-CNN can improve performances without too much compromises.

6. Conclusion

The claim "G-convolutions increase the expressive capacity of the network without increasing the number of parameters" was tested, and true in most cases, but tests on small network or particular datasets show that it's not always like that, and most importantly tuning has to be done, because the G-CNN changes the network behaviour at training time, and its width. Stopping the rotation equivariance in the last layers apparently performs well in some datasets, it's more accurate than plain CNN and G-CNN. With symmetrical datasets the G-CNN really shines, as (Veeling et al., 2018) showed us. Cohen's work is important because highlights the importance of structured representations. I think that the idea of making the network itself rotation equivariant is good, but implemented in this way has the problem that if you want to make the network equivariant to all the possible rotations, you end up with 360 times more parameters. In case of P4 group, the network still has to learn to be invariant to rotations that are not multiples of 90 degrees.

References

Cohen, T. S. and Welling, M. Group equivariant convolutional networks. *CoRR*, abs/1602.07576, 2016. URL http://arxiv.org/abs/1602.07576.

Dieleman, S., Fauw, J. D., and Kavukcuoglu, K. Exploiting cyclic symmetry in convolutional neural networks. *CoRR*, abs/1602.02660, 2016. URL http://arxiv.org/abs/1602.02660.

Lengyel, A. and van Gemert, J. C. Exploiting learned symmetries in group equivariant convolutions. *CoRR*, abs/2106.04914, 2021. URL https://arxiv.org/abs/2106.04914.

Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., and Welling, M. Rotation equivariant cnns for digital pathology. *CoRR*, abs/1806.03962, 2018. URL http://arxiv.org/abs/1806.03962.