Reproduction blog of Scale-Equivariant Steerable Networks

Delft University of Technology
Wolfgang Bubberman 4704673 w.l.bubberman@student.tudelft.nl
Olaf Braakman 4695011 o.j.braakman@student.tudelft.nl

April 16, 2021

1 Introduction

This blog is part of the Deep Learning course at the Delft University of Technology and will walk you through the attempted reproduction of Scale-Equivariant Steerable Networks [6] by Sosnovik et al. The original paper introduces the concept and provides the building blocks need to construct scale-equivariant convolutional networks. They developed and implemented a state-of-the-art scale-convolution and also showcase the computational efficiency and numerical stability of the method. Furthermore, the authors compare their findings to other proposed models also attempting to introduce the same concept of scale-equivariance. All experiments of the paper are done on the MNIST-scale and the STL-10 datasets. Throughout this blog we hope to explain parts of the paper and reproduce some of the findings of table 2 and 3. We will also critique the paper in some ways, as there are some inconsistencies between their findings in the paper and their actual code.

2 Scale-equivariant mappings

Convolution Neural Networks (CNNs) are a cornerstone within the deep learning field. They have many application when it comes to image processing. At their foundation, CNNs are equivariant under translation as the convolution window slides across the image. However, there is no particular mechanism for dealing with changes in scale. Naturally objects in images can be nearby or far away, so one way of dealing with this is to train the CNN with augmented data. Attempts have already been made by Xu et al. [9] and by Kanazawa et al. [4]. However, they are far from optimal due to the time complexity which arises from their methods. Worrall and Welling do propose a viable version of a deep scale space which is equivariant over scale [8]. But this method can only handle integer downscales and therefore has its limitations. To improve, Sosnovik, Szmaja and Smeulders [6] propose the theory of scale-equivariant networks and demonstrate the concept of steerable filter parameterization. This method is, according to their theory, not limited by integer scale factors, but can instead handle an arbitrary scaling factor. To support their claims, Sosnovik et al. lay a complex theoretical foundation. They start their adventure by mathematically proving that a standard convolution is not scale-equivariant. Meaning it is not possible to perform the same translation trick CNNs rely on. Therefore, they turn towards symmetry groups.

The symmetry of an object is the set of transformations which leaves 'core' properties of an object intact [2]. An example of a symmetry group is the group $p4$ [1], which consists of all possible translations (CNN property) and all rotations by 90 degrees. A visual example of the $p4$ group is given in Figure 1

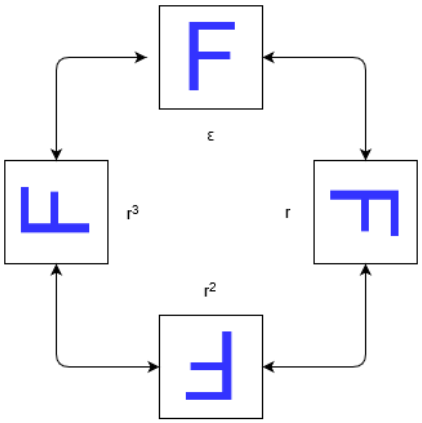

Figure 1: A visual example of the symmetry group $p4$

Instead of a symmetry group relying of rotation, Sosnovik et al. propose a new symmetry group H, the scale translation group. Based on this and some clever maths they build scale-equivariant mappings. Finally, they prove the equivariance of their newly constructed convolution. With this new model, the authors are able to group scale symmetries under a convolution. In the GitHub[1] of their work they showcase the following example where the scale-equivariant convolution is able to much better capture the details in a scaled down version of the same image compared to standard methods:

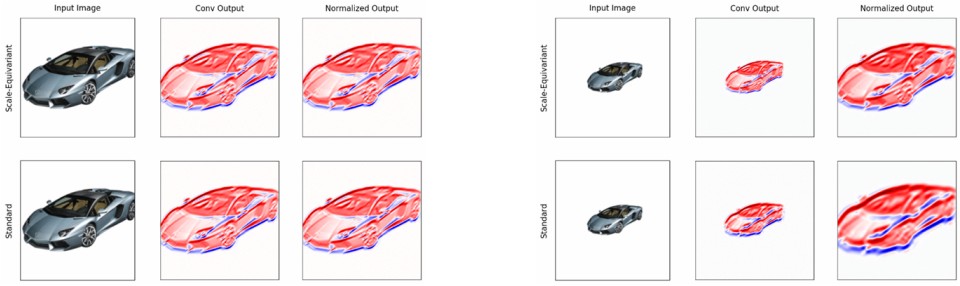

Figure 2: Comparison of scale-equivariant mappings to standard methods

If you are interested in reading more about symmetry groups as a foundation for the maths presented in the paper, Casper van Engelenburg has a great blog about it here.

[1] https://github.com/ISosnovik/sesn

3 Experiments of the paper

The experiments of the paper which we will study in more depth in this blog are the results showcased in Table 2 and Table 3.

Method	(28×28)	(28×28) +	(56×56)	(56×56) +	# Params
CNN	2.56 ± 0.04	1.96 ± 0.07	2.02 ± 0.07	1.60 ± 0.09	495 K
SiCNN	2.40 ± 0.03	1.86 ± 0.10	2.02 ± 0.14	1.59 ± 0.03	497 K
SI-ConvNet	2.40 ± 0.12	1.94 ± 0.07	1.82 ± 0.11	1.59 ± 0.10	495 K
SEVF Scalar	2.30 ± 0.06	1.96 ± 0.07	1.87 ± 0.09	1.62 ± 0.07	494 K
SEVF Vector	2.63 ± 0.09	2.23 ± 0.09	2.12 ± 0.13	1.81 ± 0.09	475 K
DSS Scalar	2.53 ± 0.10	2.04 ± 0.08	1.92 ± 0.08	1.57 ± 0.08	494 K
DSS Vector	2.58 ± 0.11	1.95 ± 0.07	1.97 ± 0.08	1.57 ± 0.09	494 K
SS-CNN	2.32 ± 0.15	2.10 ± 0.15	1.84 ± 0.10	1.76 ± 0.07	494 K
SESN Scalar	2.10 ± 0.10	1.79 ± 0.09	1.74 ± 0.09	1.50 ± 0.07	495 K
SESN Vector	$\mathbf{2.08 \pm 0.09}$	$\mathbf{1.76 \pm 0.08}$	$\mathbf{1.68 \pm 0.06}$	$\mathbf{1.42 \pm 0.07}$	495 K

Table 2: Classification error of different method on MNIST-scale dataset, lower is better. In the experiments an image resolution of 28x28 and 56x56 was used. Both the regime without data augmentation, and the regime with scaling data augmentation, denoted with "+", were tested. All results are reported as mean std over 6 different fixed realizations of the dataset. The best results are bold.

3.1 Table 2

Table 2 from the paper showcases the conducted experiments on the MNIST-scale dataset. The images of the MNIST dataset were rescaled from 0.3 to 1.0 of the original size and padded with zeros to not lose the initial resolution. This is all done uniformly and independent for every image. The resulting dataset is split into 10 000 for training and 50 000 for testing, 6 different realizations were done and fixed for all experiments to get a good median. The model from baseline CNN used is the one from [3] as it is described as the state-of-the-art result on the MNIST-scale dataset. For all the other models a scalar variant was made as well by adding an extra projection layer. This extra projection layer transforms the vector features in each spatial position of each channel into scalar features. The models were trained with the Adam optimizer for 60 epochs with a batch size of 128 and a learning rate of 0.01, which was divided by 10 after 20 and 40 epochs. In total four different regimes were used as well for each of the models. First an additional regime was used were the images were up-scaled to 56x56, and then the regimes were doubled by having ran the experiments without data augmentation and with scaling augmentation (those regimes denoted by the "+". Again, 6 different realizations were used. The paper also discloses that the values may differ somewhat as variations in the generated datasets and different training procedures may impact the results. These experiments of the paper find that SESN outperforms all other methods in all 4 regimes with the vector variant being the best.

3.2 Table 3

Table 3 from the paper showcases the conducted experiments on the STL-10 dataset. The dataset consists out of 8 000 for training and 5 000 for testing labeled images, with an addition of 100 000 extra unlabeled images. The images themselves have a resolution of 96x96 with RGB channels. The images belong to 10 different classes. The baseline chosen was the WideResNet (WRN) model. Then for the own models; SESN-A just vector features was used, for SESN-B maximum scalar projection several times in the intermediate layers was used and for SESN-C interscale interaction was used (performing the worst of the three). All the models were trained for 100 epochs with a batch size of 128 (64 for "bad" gpus as the code states), a learning rate of 0.1, which was divided by 5 after 300, 400, 600 and 800 epochs.

Method	Error, %	# Params
WRN	11.48	11.0 M
SiCNN	11.62	11.0 M
SI-ConvNet	12.48	11.0 M
DSS	11.28	11.0 M
SS-CNN	25.47	10.8 M
SESN-A	10.83	11.0 M
SESN-B	**8.51**	11.0 M
SESN-C	14.08	11.0 M
Harm WRN	9.55	11.0 M

Table 3: Classification error on STL-10. The best results are bold. Additionally reported the current best result achieved by Harm WRN from Ulicny et al. [7]

These experiments of the paper find that SS-CNN performs a lot worse than the baseline, whilst the other preexisting methods perform comparable. They also find that SESN-C performs significantly worse than SESN-A and SESN-B due to the high equivariance error introduced by interscale interaction. SESN-B improves the results of all other methods, supposedly due to projection between the scales according to the paper.

4 Reproduction

We chose to reproduce the results on the Google Cloud Platform (GCP), as this seemed like the best solution; we both have to use our machines for different things besides this reproduction as well. We used the code written and provided by the authors of the paper, Sosnovik et al. For the STL-10 dataset some tweaks to the hyperparameters were made by ourselves. All work done for the reproduction was done together, by the both of us. In the end we used up 5 different GCP coupons, all worth 50 euros, for the reproductions we managed to do, which is quite a lot and an expensive reproduction.

4.1 Reprodution Table 2

As can been seen from the comparison between reproduced Table 2 and the original Table 2 the general results seem to line up. In both tables SEVF Vector performs the worst in all situations and SESN performs the best. However, there are some slight differences: the results we seem to have reproduced are all better by a small margin and the standard deviation of everything is quite a bit higher. This most likely result of our replication rate of 3. We can also see that in our reproduction, SESN Scalar is actually better for the 28x28 and 28x28+ situation, compared to the SESN Vector, which is always better in the original paper. As we did not tweak the hyperparameters for the MNIST-scale dataset, it is hard to say where those differences came from, but they are definitely within a margin of error so we conclude that the results from Table 2 are indeed reproducible.

Method	(28×28)	(28×28)+	(56×56)	(56×56)+
CNN	2.26 ± 0.33	-	1.86 ± 0.22	-
DSS Scalar	2.28 ± 0.31	-	1.76 ± 0.20	-
DSS Vector	2.26 ± 0.33	-	1.77 ± 0.23	-
SI-ConvNet	2.22 ± 0.25	-	1.72 ± 0.16	-
SEVF Scalar	2.13 ± 0.17	-	1.73 ± 0.11	-
SEVF Vector	2.47 ± 0.24	-	1.93 ± 0.13	-
SS-CNN	2.18 ± 0.16	-	1.79 ± 0.16	-
SiCNN	2.16 ± 0.31	-	1.74 ± 0.22	-
SESN Scalar	1.96 ± 0.15	1.98 ± 0.16	1.65 ± 0.13	1.65 ± 0.16
SESN Vector	1.99 ± 0.21	1.99 ± 0.19	1.59 ± 0.13	1.64 ± 0.14

Reproduction of Table 2, lower is better.

Something of note is that the code that was provided by authors of the original paper did not seem to run the regimes with scaling data augmentation, denoted with "+", for the models not made by the authors. This is a slight point of concern, as we do not see a good reason for this, and there clearly are reported values in the original paper. As, reproducing the values so far took quite a bit of time, we did not try and find out how to run these regimes ourselves.

4.2 Reprodution Table 3

As can be seen from a quick glance, sadly we were not able to reproduce the majority of the values of Table 3. However, let us start of with the fact that the values that we were able to reproduce do seem in line with the original findings. The baseline model, only differs with 0.5% in our reproduction compared to the original and SS-CNN indeed performs quite bad compared to the baseline as the original paper states. However, the most interesting values of this table are missing, these values being the SESN-A, SESN-B and SESN-C error rates. This is as no matter what we tried we could not get them to run with the code provided, all efforts let to graphics card memory shortages. Even using the lower batch size suggested in a seperate method in the code, a batch size of 64 (and a learning rate of 0.05), let to a memory shortage. We even tried lowering the batch size further to 32 and lowering the learning rate to 0.025, but after having run this on the GCP for a day or two, only 51 of the 1 000 epochs had finished, which makes sense as the learning rate

was abhorently low to allow for such batch sizes. Something interesting we figured out after many trials and errors later, was that when using the default hyperparameters in the code, which are the hyperparameters the paper says they used, we ran out of graphics memory on the 16GB video card our GCP virtual machine instance had. This is quite odd, as the authors of the paper themselves claim to have used a GTX 1080TI, which only has 11GB of memory. Therefore, we speculate that the authors of the paper actually used a server or larger computing network for the experiments of Table 3, which allowed them to actually run the experiments instead of having them crash.

Method	Error %	# Params
WRN	11.98	11.0 M
SiCNN	-	11.0 M
SI-ConvNet	-	11.0 M
DSS	-	11.0 M
SS-CNN	26.86	11.0 M
SESN-A	-	11.0 M
SESN-B	-	11.0 M
SESN-C	-	11.0 M
Harm WRN	-	11.0 M

Reproduction of Table 3, lower is better.

This, as one might guess, costed a lot of our valuable time for this reproduction project, which was somewhat disappointing as we already did not have much time for the project. Sadly, we must hereby deem the Table 3 not easily reproducible and there is definitely something going on with the results of Table 3 that requires further investigation, which should be done in future reproduction work for this paper.

4.3 Additional reproduction for Table 2

As we were frustrated by the lack of results the STL-10 dataset gave us, we again looked at the MNIST-scale dataset to see if we could do some extra reproduction. Looking through the paper we found Table 4, describing the time performance of Table 2 for the 28x28 and 56x56 regimes. We decided to reproduce this as well, as this seemed like an interesting addition to the reproduction of Table 2. All values of that table we were able to reproduce, but with quite a few differences in the results. We also managed to create values of the time performance for the 28x28+ and 56x56 regimes for the SESN Scalar and SESN Vector models, which the original paper did not do. However, the differences are sometimes quite big, whilst the baseline CNN, SI-ConvNet and SEVF seem to match up pretty closely with the original time performance, the other models vary from a little to quite a lot. SESN and SiCNN, for example, seem to perform horrible for the 56x56 regime compared to the original findings. What we think happened here, is that GPUs used for the original findings and the reproduction done by us, must differ in such a way that these results vary. Therefore one cannot assume that the runtimes of a reproduction will be the same as the original, a valuable lesson to us as we expected the running of experiments to not take more than two days, the original total runtime mentioned by Sosnovik et al.

Method	(28x28)	(28x28)+	(56x56)	(56x56)+
CNN	3.9s	-	4.2s	-
DSS Scalar	7.9s	-	17.4s	-
DSS Vector	7.7s	-	17.0s	-
SI-ConvNet	13.0s	-	30.4s	-
SEVF Scalar	21.2s	-	52.3s	-
SEVF Vector	24.6s	-	53.9s	-
SS-CNN	9.8s	-	22.2s	-
SiCNN	29.6s	-	166.2s	-
SESN Scalar	6.3s	9s	28.1s	15.1
SESN Vector	7.4s	12.5s	81.3s	64.1

Reproduction of Table 4 time performance, see Appendix B of original paper. Lower is better.

5 Conclusion

To conclude, the paper introduces some very interesting ideas for dealing with scale-equivariant mappings and the usage of symmetry groups for this. It should be noted that the paper does this in a very complicated manner, overusing mathematics, which could be described as "mathiness", see [5]. However, not all results of the paper were easy to reproduce with the code provided. The main flaw of the reproduction of Table 2 can be contributed to the fact that, using the code provided, the regimes with scaling data augmentation could not be ran. Luckily, the reproduction results that could be done, were consistent with the findings in the paper for this table. Table 3, however, is another story. This table was not easy to reproduce and costed significantly more resources to run. We were only able to reproduce their baseline method (WRN) and the SS-CNN model, but all other models led to memory shortages, this was quite interesting as we used a GPU with more memory than the paper says they used (11GB vs 16GB). Therefore we think that the authors used a server or larger computing unit to produce the results presented. We also learnt that the time the paper projected it would cost to reproduce the experiments was not really accurate, and severely underestimated this for the STL-10 data-set. We plan to take more note of this in any future reproductions we might do. All in all, it was an interesting reproduction, and we hope you, the reader, might also read/view new papers with a more critical attitude than before.

A Time Performance

Method	28×28, s	56×56, s
CNN	3.8	3.8
SiCNN Scalar	13.5	18.9
SiCNN Vector	15.3	22.8
SI-ConvNet	18.4	33.1
SEVF Scalar	21.0	38.4
SEVF Vector	25.4	46.0
DSS Scalar	3.9	5.0
DSS Vector	3.9	4.8
SS-CNN	14.8	16.6
SESN Scalar	3.8	5.1
SESN Vector	3.8	6.8

Table 4: Average time per epoch during training on input data with resolution 28x28 and 56x56 [7]

References

[1] Taco S. Cohen and Max Welling. Group equivariant convolutional networks, 2016.

[2] Robert Gens and Pedro M Domingos. Deep symmetry networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[3] Rohan Ghosh and Anupam K Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1906.03861*, 2019.

[4] Angjoo Kanazawa, Abhishek Sharma, and David W. Jacobs. Locally scale-invariant convolutional neural networks. *CoRR*, abs/1412.5104, 2014.

[5] Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

[6] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. *arXiv preprint arXiv:1910.11093*, 2019.

[7] Matej Ulicny, Vladimir A Krylov, and Rozenn Dahyot. Harmonic networks with limited training samples. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2019.

[8] Daniel E. Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *CoRR*, abs/1905.11697, 2019.

[9] Yichong Xu, Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, and Zheng Zhang. Scale-invariant convolutional neural networks. *CoRR*, abs/1411.6369, 2014.