Scale-Equivariant Steerable Networks
Reproducibility Project – CS4240 Deep Learning

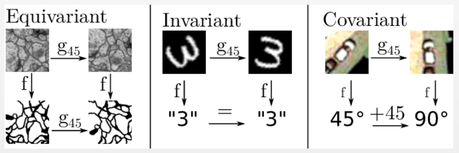
Mark Erik Lukacs, Stefan Petrescu

1 *Introduction*

This poster provides an overview of the reproducibility project for the Deep Learning Course. We reproduced the paper's results, running the experiments on our own, using the available code.
This poster provides an overview of the reproducibility project for the Deep Learning Course. We reproduced the paper's results, running the MNIST experiments on our own, using the available code.

The reason why CNNs excel in image processing is that convolutional layers are translation equivariant. This was achieved by excessive weight-sharing.
A special case of equivariance (which is valuable in image recognition tasks) is invariance. It is achieved by the feature-extractor pooling layers. [see image]

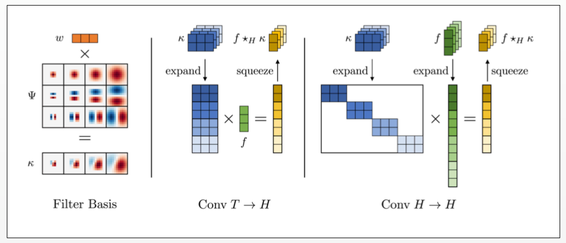
[ref g4] in their paper aimed to further extend the weight sharing via group convolutions. Building on this [ref sesn] defined scale-equivariant networks for weight sharing. Their network is invariant to both translation and scaling.



Scale-equivariance is derived from a mathematical concept: group-equivariance. A group equivariant transformation means that if the input of the layer is transformed by *g*, the output is also transformed by *g*. And *g* can be anything, like rotation, mirroring (g4 paper), scale (our paper), or some totally bullshit, thing like inversion.

2 *Method*

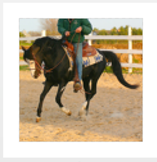
In our work we aimed to understand the background of the model
iny-tiny mathematical background
eq7
implementation
our implemented T-H layer from scratch

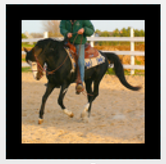


3 *Experiments*

For the experiments, two datasets were considered: MNIST [REF] and STL-10 [REF]. For both we had to process data, following the specific paper guidelines. For the MNIST, the data processing step consisted of rescaling the images, based on a randomly sampled uniform factor, between 0.3-1. Following this procedure, 6 different such realizations were generated. Each consisted of 60000 images, each split into 10000 for training, 2000 for evaluation, and 48000 for testing.
To realize this, we created some python scripts, which can be found on our project's GitHub repository.

For the STL-10 dataset, data augmentation was also applied. Thus, in this case the images were normalized, padded with a 12px border (2) and then randomly cropped (3) to their initial 96x96px size. Furthermore, random horizontal flips (4) with a probability of 50% were applied & cutout of 32px (5).


1


2


3


4


5

5 *Results*

Being able to replicate the MNIST experiments, we were able to reproduce the results. These are somewhat similar, with some small differences.
For the MNIST dataset, we ran the experiments on the following models:

- **mnist_ses_scalar_28**
- **mnist_ses_scalar_56**
- **mnist_ses_vector_28**
- **mnist_ses_vector_56**
- **mnist_ses_scalar_28p**
- **mnist_ses_scalar_56p**
- **mnist_ses_vector_28p**
- **mnist_ses_vector_56p**

For each of these, 2 experiments were conducted: one for which the scaling factor was set to 1 and one for which the scaling factor was set to 0.5 Therefore, for each model we obtained 12 results (6 for each realization). The results were stored in "results.yml" which was later processed using a python script. These can be found in the table below.

Method	28 x 28	28 x 28 +	56 x 56	56 x 56 +
SESN Scalar	1.95 ± 0.17	1.98 ± 0.14	1.68 ± 0.12	1.67 ± 0.18
SESN Vector	2.00 ± 0.2	2.00 ± 0.21	1.66 ± 0.17	1.59 ± 0.16

Acknowledgements & Links

We would like to thank Nergis Tömen and Tomasz Motyka for their advice and helpful insight.

Link to paper:
https://arxiv.org/abs/1910.11093

Link to our project's website:
https://spetrescu.github.io

Link to our project's GitHub repo:
https://github.com/vioSpark/reproduction-project-DL2021