ICLR 2019 Reproducibility Challenge / Dense Morphological Networks

Vincent Luczkow, Jonathan Maloney-Lebensold, & Yi Feng Yan
Team: Buffalo^3
Department of Computer Science
McGill University

{vincent.luczkow,jonathan.maloney-lebensold,yi.f.yan}@mail.mcgill.ca

Abstract

The 2019 ICLR Reproducibility Challenge aims to replicate papers to improve publication quality and introduce greater rigour in accepted papers. As part of this challenge, we replicated *Dense Morphological Networks: An Universal Function Approximator* Mondal et al. (2019). The paper introduces a novel network architecture and claims to achieve comparable accuracy to other simple network architectures with fewer parameters. We were able to replicate the claimed results in the paper, achieving comparable performance on most benchmarks in the original paper, although we did not attempt to replicate all results.

1 Introduction

The authors introduce a novel neural network architecture based on *dilation* and *erosion* neurons. In essence they introduce a new activation function (dilation-erosion) and reverse the usual order of operations in neural networks, applying the activation function first followed by a matrix multiplication.

They prove that this architecture is a universal function approximator in the same sense as a fully connected neural network. The authors argue that this architecture can learn more complex decision boundaries than a fully connected neural network, given a similar number of parameters.

They compare their architecture to fully connected networks with several common activation functions: ReLU (Nair & Hinton (2010)), tanh, and maxout (Goodfellow et al. (2013)). They benchmark these networks on several datasets: a toy concentric circles dataset, MNIST (Lecun et al. (1998)), Fashion-MNIST (Xiao et al. (2017)), CIFAR-10, and CIFAR-100 (Krizhevsky & Hinton (2009)).

They deepen their architecture in two ways. First, they apply multiple dilation-erosion layers in succession followed by a single matrix multiplication. They find that the first architecture takes a significantly longer time to train because only the gradient of a dilation-erosion layer is very sparse, and this problem is exacerbated by stacking multiple layers.

Then they try alternating between dilation-erosion and matrix multiplication layers. They find that the second architecture does not suffer from this issue, but does overfit the training set.

Finally, they suggest that future work should be focused on overcoming the overfitting problem in deeper networks, and in extending the dilation/erosion operation to a convolutional setting.

2 Technical Summary of Dense Morphological Networks

A dense morphological network is built on *dilation-erosion layers*, which consist of individual *dilations* and *erosions*.

A single dilation ($\oplus$) and a single erosion ($\ominus$) correspond to the following operations:

$$x \oplus y = \max_i (x_i + y_i)$$
$$x \ominus y = \max_i (x_i - y_i)$$

where $x, y \in \mathbb{R}^n$.

k dilations and l erosions are combined into a single dilation-erosion layer $f : \mathbb{R}^n \to \mathbb{R}^{k+l}$, which takes an n-dimensional vector and produces a $k + l$ dimensional vector. The j-th element of this output vector is defined as follows:

$$f(x)_j = \begin{cases} x \oplus W_j & \text{if} j \leq k \\ x \ominus W_j & \text{otherwise} \end{cases}$$

where W is a $(k + l) \times n$ matrix of weights, W_j is the j-th row of W, and x is the input sample.

The full architecture used in most of the experiments is a single dilation-erosion layer, followed by a matrix multiplication and a softmax layer to obtain the final class probabilities:

$$\hat{y} = \sigma(Mf(x))$$

where M is a $c \times (k + l)$ matrix of weights (c is the number of classes being predicted).

The deep architectures introduced at the end of the paper correspond to successive dilation-erosion layers followed by a matrix multiplication:

$$\hat{y} = \sigma(Mf_1(f_2(\ldots(f_n(x)))))$$

and alternating matrix multiplications and dilation-erosion layers:

$$\hat{y} = \sigma(M_1 f_1(M_2 f_2(\ldots(M_n f_n(x)))))$$

All architectures were trained using backpropagation with the Adam optimizer (seen in Kingma & Ba (2014)).

3 Implementation goals for Dense Morphological Networks

We attempted to replicate every result except CIFAR-100 and the results for the deeper networks. The former we decided not to replicate due to computational constraints, and the latter due to the lack of strong results.

The paper omitted analysis of the baseline networks on the MNIST dataset, and we decided to generate these results ourselves. This was partially because it would offer some confirmation of correctness of our baseline implementations, and partially because we wanted to have an additional comparison between DenMo and the baselines, especially since we did not attempt to replicate the CIFAR-100 results.

4 Methodology

We felt that most important experiment to reproduce was CIFAR-10. The CIFAR-10 experiment is the only one that compares DenMo directly to other architectures, and is thus the best for evaluating the claims of the paper. Unlike MNIST and Fashion-MNIST, CIFAR-10 is not as easily solved with typical feed-forward network architectures. We did not prioritize evaluating the toy dataset for two reasons: the dataset was not provided and beyond the visualization, it did not bolster the theoretical claims of the paper.

All architectures were implemented using the PyTorch framework (Paszke et al. (2017)). The DenMo and maxout (Golge (2018)) architectures were implemented from algebraic primitives, while the Tanh and ReLU architectures were built from existing implementations in PyTorch.

In order to produce consistent results, we developed a pipeline to produce standard reports across model architectures, hyperparameters and datasets.

5 Challenges

Overall, the authors did a good job providing hyperparameters and implementation details. Despite a clear explanation of their results, there were a few omissions. Running source code was not provided during the review process, however they were willing to review our PyTorch implementation. The authors also indicated that they will be releasing source code after the review process. The paper did not mention how the DenMo network was initialized and we only found out after presenting our initial results that they had elected to use Xavier uniform initialization (Glorot & Bengio (2010)) for the DenMo network.

The methods used for producing the circle dataset were omitted from the report.

6 Results

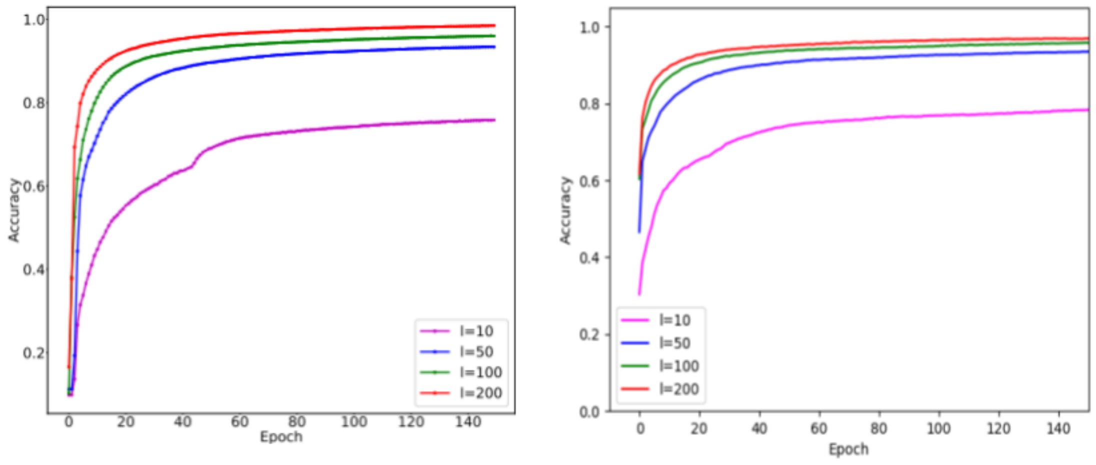

Figure 1: Paper results (left) and our results (right) for dense morphological networks of different widths.

In Figure 1, the authors performed a series of comparisons between various dense morphological network architectures trained on the MNIST dataset. They show that (unsurprisingly) the accuracy of the networks increases as the width of the networks increases. We were able to replicate these results almost exactly.

They then (Figure 2) assess the change in performance as the relative proportion of dilations to erosions changes. All perform quite well, and we were again able to replicate their results almost exactly.

Their main inter-model comparison is Figure 3, which compares the performance of their architecture to other architectures with a similar number of parameters. We were able to replicate the morph network performance, but got *worse* results for the baselines.

In Figure 4 they perform another comparison of different dilation/erosion distributions, this time on CIFAR-10. We were able to replicate these results as well, although our models did not learn quite as fast as theirs.

See the full set of results in Table 1.

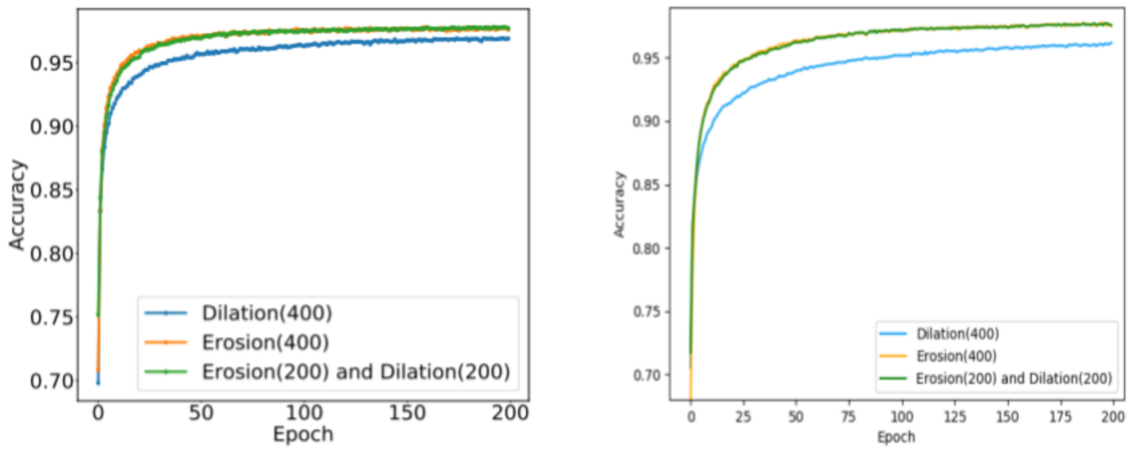

Figure 2: Paper results (left) and our results (right) for different relative numbers of dilations and erosions.

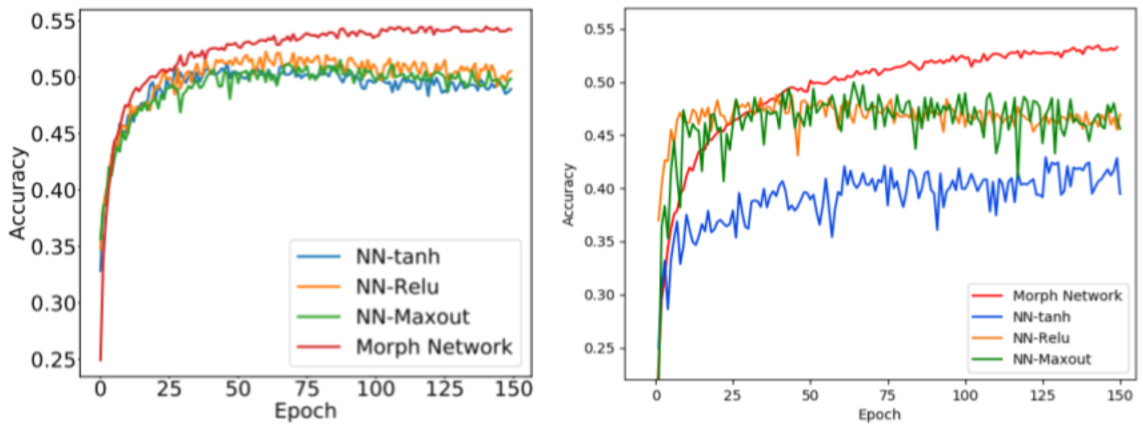

Figure 3: Paper results (left) and our results (right) the CIFAR-10 dataset.

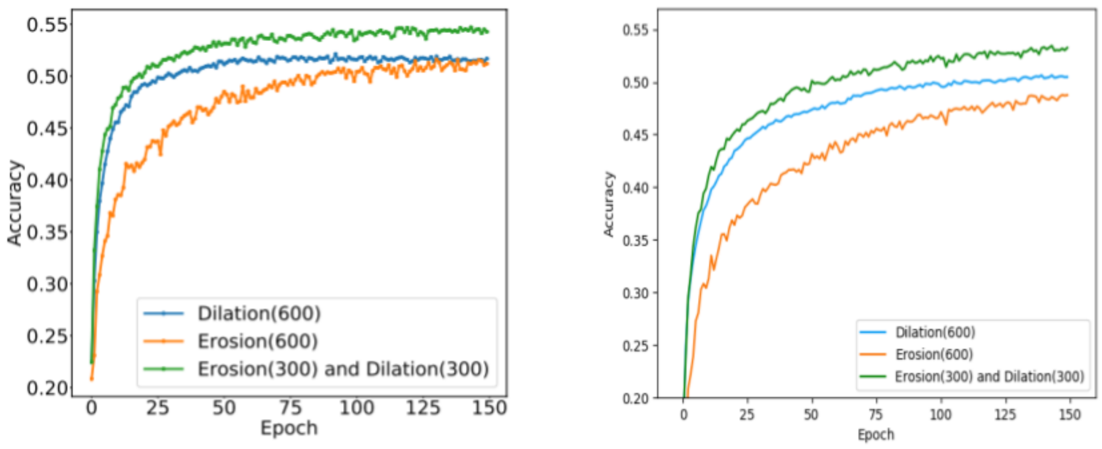

Figure 4: Paper results (left) and our results (right) for different DenMo nets on CIFAR-10.

Table 1: Empirical results from the paper compared to our own.

	ARCH. / DATA	EPOCHS	PARAMETERS	THEIRS	OURS
1	DenMo MNIST	400	D:5, E:5	76.35	80.62
2	DenMo MNIST	400	D:25, E:25	93.38	94.66
3	DenMo MNIST	400	D:50, E:50	95.51	96.32
4	DenMo MNIST	400	D:100, E:100	96.85	93.94
5	DenMo MNIST	400	D:400, E:0	~95	97.45
6	DenMo MNIST	400	D:0, E:400	~95	96.45
7	DenMo MNIST	300	D:200, E:200	98.43	97.56
8	DenMo Fashion_MNIST	300	D:400, E:400	89.87	98.19
9	DenMo CIFAR-10	150	D:100, E:100	51.84	50.61
10	Tanh CIFAR-10	150	h_size:200	48.88	50.25
11	ReLU CIFAR-10	150	h_size:200	49.28	46.04
12	Maxout CIFAR-10	150	h_size:200	49.51	51.26
13	DenMo CIFAR-10	150	D:200, E:200	53.41	52.23
14	Tanh CIFAR-10	150	h_size:400	49.39	51.88
15	ReLU CIFAR-10	150	h_size:400	50.43	47.95
16	Maxout CIFAR-10	150	h_size:400	50.10	50.64
17	DenMo CIFAR-10	150	D:300, E:300	54.49	53.26
18	DenMo CIFAR-10	150	D:0, E:600	~50	48.75
19	DenMo CIFAR-10	150	D:600, E:0	~50	50.48
20	Tanh CIFAR-10	150	h_size:600	51.24	53.01
21	ReLU CIFAR-10	150	h_size:600	52.25	39.72
22	Maxout CIFAR-10	150	h_size:600	51.51	51.43
23	Tanh MNIST	400	h_size: 200	N/A	98.00
24	ReLU MNIST	400	h_size: 200	N/A	88.36
25	Maxout MNIST	400	h_size: 200	N/A	98.10

7 Possible Sources of Error

In our correspondence with the authors, they indicated that their DenMo implementation was done using Keras. Depending on the back-end framework (e.g. Tensorflow, Theano), there may be implementation details which would affect performance. Given that our results were very similar to those reported, we could not identify a specific source of possible error.

Table 2: Reproducibility Checklist

	CRITERIA	EVALUATION
	Algorithms	
1	A clear description of the algorithm.	✓
2	An analysis of the complexity (time, space, sample size) of the algorithm.	N/A
3	A link to downloadable source code, including all dependencies.	✗
	Theoretical Claims	
4	A statement of the result.	✓
5	A clear explanation of any assumptions.	✓
6	A complete proof of the claim.	✓
	Figures and Tables	
7	A complete description of the data collection process, including sample size.	✓[1]
8	A link to downloadable version of the dataset or simulation environment.	✗
9	An explanation of how samples were allocated for training / validation / testing.	✓
10	An explanation of any data that was excluded.	N/A
11	The range of hyperparameters considered, method to select the best hyperparameter configuration, and specification of all hyperparameters used to generate results.	✗[2]
12	The exact number of evaluation runs.	✓
13	A description of how experiments were run.	✗
14	A clear definition of the specific measure or statistics used to report results.	✓
15	Clearly defined error bars.	✗
16	A description of results including central tendency and variation.	✗
17	The computing infrastructure used.	✗

1. The toy dataset was not provided
2. Hyperparameters are provided, but no reasoning or method is mentioned

8 Conclusion

We were able to replicate all the morphological network results for common datasets (MNIST, Fashion-MNIST, CIFAR-10). While our results for the baseline implementation (ReLU, Tanh, maxout) did not match exactly, we do not consider this problematic. See Pineau (2018) for a reproducibility checklist presented at NeurIPS 2018. We felt it would be a worthwhile exercise to evaluate this paper with respect to this checklist in Table 2.

9 Appendix

9.1 Environment Details

Work was performed with 2 machines running NVIDIA hardware: a GeForce GTX 1060 and a GeForce GTX 970. Our programming environment was Python 3.6.

9.2 Hyperparameters

Normalization was the the pre-processing operation we applied. We were able to find almost all of the hyperparameters for the models and datasets in the original paper and the OpenReview comments.

- All models were optimized with the Adam optimizer, with $\mathrm{learningrate} = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$
- The DenMo networks were initialized with Glorot initializationGlorot & Bengio (2010)
- The loss function is cross entropy.
- The output of each network is passed through softmax to produce class probabilities.

- The number of training epochs varies between datasets, and is recorded with each individual results.

References

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `http://proceedings.mlr.press/v9/glorot10a.html`.

Eren Golge. Maxout layer 805, 2018. URL `https://github.com/pytorch/pytorch/issues/805`.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28, 3 of *Proceedings of Machine Learning Research*, pp. 1319–1327, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `http://proceedings.mlr.press/v28/goodfellow13.html`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

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

Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Ranjan Mondal, Sanchayan Santra, and Bhabatosh Chanda. Dense morphological network: An universal function approximator, 2019. URL `https://openreview.net/forum?id=SyxknjC9KQ`.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 807–814, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. URL `http://dl.acm.org/citation.cfm?id=3104322.3104425`.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Joelle Pineau. Neurips: Reproducible, reusable, and robust reinforcement learning, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.