

Orthogonalizing Weights in Capsule Network Architecture



Shubhranil Kundu and B. Gagana

Abstract Scalar neural network algorithms are limited in their ability to understand scale, rotational, or affine transformations within images and resort to average or max-pooling techniques which result in translational invariance. In an attempt to overcome these limitations, Hinton et al. introduced vectorized capsule network frameworks which support equivariance while capturing spatial relationships between data points, thus enhancing predictive capabilities of networks. However, experimenting with activation functions, hyperparameters, and optimizers have proven faster convergence and orthogonalizing weights within the layers of capsules enhance performance by slashing associated average error rates.

Keywords Weights · Orthogonality · Capsules

1 Introduction

1.1 Capsule Network Architecture

The vector based encoder–decoder neural architecture [1] mainly comprises of six layers namely convolutional 2D layer, primary capsule layer, digit capsule layer along with three fully connected layers while handling a total of 8,238,608 parameters during the process of image reconstruction.

The detailed architecture is as demonstrated (see Fig. 1).

A detailed breakdown of the layers of the capsule network is as follows.

Encoder. The encoder accepts the input image and predicts the set of structure variables corresponding to transformation matrices while dealing with margin as

S. Kundu (✉)
Madhav Institute of Technology, Gwalior, India
e-mail: neel.sk37@gmail.com

B. Gagana
PES University, Bangalore, India
e-mail: gaganab20496@gmail.com

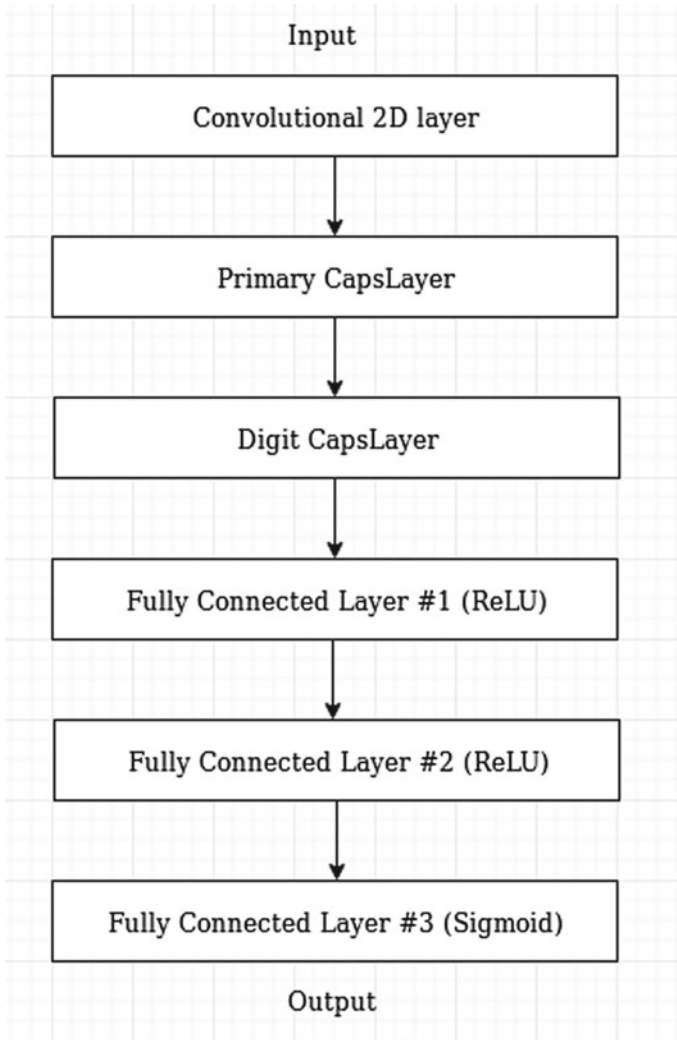


Fig. 1 Layer wise Capsule Network architecture

well as reconstruction loss. The functionality of each of the layers is as described below.

Convolutional 2D Layer. The layer with $9 \times 9 \times 1 \ 2^8$ kernels with stride 1, accepts the 28×28 input image and outputs a $20 \times 20 \times 256$ tensor and has 20,992 trainable parameters in all.

Primary Caps Layer. This layer which is composed of 32 primary capsule units, accepts the $20 \times 20 \times 256$ tensor from the previous layer, applies $9 \times 9 \times 256$ convolutions to the input volume and produces combinations of the already detected

features. The output of this layer is a $6 \times 6 \times 8$ tensor and has 5,308,672 trainable parameters.

Digit Caps Layer. The 10 digit capsules layer accepts the $6 \times 6 \times 328$ dimensional tensor and maps the input to a 16D capsule output space. These coefficients are then used in routing [2]. The layer has 149,760 trainable parameters per capsule.

The loss of the system is defined by

$$\text{loss} = \text{margin_loss} + \text{reconstruction_loss} \quad (1)$$

where the reconstruction loss function is defined by a regularization factor which doesn't dominate over the marginal loss. The margin loss function for each category c is defined by a zero loss when correct prediction occurs with probability greater than 0.9 or when an incorrect prediction occurs having a probability less than 0.1 (non zero otherwise in either case). The same can be mathematically represented as follows:

$$L_c = T_c \max(0, m^+ - \|v_c\|)^2 + \lambda (1 - T_c) \max(0, \|v_c\| - m^-)^2 \quad (2)$$

where λ is the numerical stability constant, and for a matching training label, there is only one correct DigitCaps capsule while the rest nine remain incorrect. The equation then retracts all values accordingly.

Decoder. The decoder part of the capsule system accepts the image format previously processed by the encoder and produces the final reconstructed output image after passing the input through three fully connected layers which also perform denoising thus, supporting routing by agreement. Each of the layers of the decoder accept processed input from the previous layers and pass the input to the next layer in a feed-forward manner while calculating number of parameters based on bias. The structure of each of the layers is as described below.

Fully Connected Layer #1. The layer with the ReLU activation function accepts the 16×10 input from the Digit CapsLayer and produces a 512 vector output while dealing with 82,432 parameters.

Fully Connected Layer #2. This ReLU activated layer accepts the 512 vector output from the previous fully connected layer and produces a 1024 vector output while dealing with 525,312 parameters.

Fully Connected Layer #3. The Sigmoid activated layer accepts the 2^{10} vector output from the previous fully connected layer and produces a 784 vector which is also the 28×28 reconstructed output while dealing with 803,600 parameters.

1.2 Orthogonality in Weights

As the training of neural networks is laborious due to vanishing or exploding gradients [3], proliferation or fluctuation in saddle points [4], and/or feature statistic shifts [5], introducing orthogonality in weights ensures faster and more stable convergence during training and is enforced through variants of Frobenius norm regularizer, mutual coherence or restricted isometric properties [6]. Orthogonality implies energy preservation and hence, stabilizes the distribution of activations over the concerned layer. The various methods for enforcement of orthogonality are as described below.

Double soft orthogonality regularizer. The soft orthogonality regularizer defined by:

$$\lambda \|W^T W - I\|_F^2 \quad (3)$$

is expanded to compensate for under and over complete cases using

$$\lambda \left(\|W^T W - I\|_F^2 + \|W W^T - I\|_F^2 \right) \quad (4)$$

where $W^T W = W W^T = I$ for an orthogonal W .

Mutual coherence regularizer. The mutual coherence regularization can be defined by:

$$\lambda \left(\|W^T W - I\|_\infty \right) \quad (5)$$

where the gradient can be solved with smoothing techniques applied to the l_∞ norm [7].

Spectral restricted isometry property regularizer. This regularization is defined by the following equation:

$$\lambda \sigma \left(W^T W - I \right) \quad (6)$$

where W is well-conditioned and regularizes computation cost from $O(n^3)$ to $O(mn^2)$

2 Related Work

2.1 Capsule Network Architecture

Fundamental research. While capsule networks are still in a nascent stage of research, experiments have been carried out around implementing additional inception blocks [8], and enabling sparsity in activation value distribution of capsules in primary capsule layer [9, 10].

Application research. Capsules have, in recent times, been employed in multitude of tasks including audio processing [11], mobility-on-demand network coordination [12], gait recognition [13], traffic classification for smart cities [14, 15], natural language tasks like sentiment analysis [16] and healthcare applications [17, 18].

2.2 Weight Orthogonality

Past research along the lines of weight orthogonality have been specifically focused toward reducing internal covariate shifts [19, 20], avoiding gradient explosions, exploring soft and hard constraints [21], using nonlinear dynamics to stabilize weight layer-wise activation distribution [20].

3 Methodology

The following section describes the architectural frameworks and the detailed experimental setup used to obtain the results described in Sect. 4.

A pytorch implementation of Capsule Networks was run on a tesla K40c GPU environment while implementing the same on the FashionMNIST [21] dataset. Since, under simplified assumptions [22], it's concluded that random initialisations produce similar convergence rates as in case of unsupervised pretraining, initial orthogonality wouldn't necessarily sustain throughout training and could breakdown if improperly regularized. With proper regularization, experimentation proves that adding orthogonality regularizations can impact accuracy as well as empirical convergence. However, the following regularizations and optimizations ensure faster convergence, and hence, have been executed for 20 epochs as the results are known to stabilize thereafter.

The modifications incorporated into the original architecture is as detailed below.

Activation or Transfer function. Activation functions are nonlinear complex functional mappings between incoming data and response variables. Capsule Networks by default employs the ReLU [23] activation function which can be mathematically represented as:

$$A(x) = \max(0, x) \tag{7}$$

However, our past experimentation [24] has shown that newer nonlinear transformations like Swish [22] outperforms standard ReLU. Ramachandran et al. proposed function can hereby be defined in terms of the sigmoid as:

$$f(x) = x.\text{sigmoid}(\beta x) \tag{8}$$

Table 1 Comparison between the original and the modified Capsule Network architecture in terms of accuracy

Architecture	Minimum accuracy value (%)	Maximum accuracy value (%)	Average accuracy value (%)
Original	48.7	93.8	82.8655
Modified	56.4	96.3	86.3420

Softmax optimization. In order to enhance the object recognition performance, additional softmax layers [25] are augmented into the decoder of the Capsule architecture and tend to have the same number of nodes as that of the previous layer [26]. The softmax layer computes the probability distributions for all involved classes and is mathematically:

$$P(y = j|x) = e_{w_j T_x + b_j} / \sum_{k \in K} e^{w_k T_x} + b_k \quad (9)$$

Optimizer. As optimizers like Adam [27] fail to stably converge at extreme learning rates, newer techniques like Adabound [28] tend to work better where the lower bound is η_l and upper bound is η_u with gradient clipping that enhances adaptive moment estimation with dynamic bounds.

Weight orthogonality. The capsnet system has two types of weights: (a) the weights between the primary and digit caps and (b) the dynamic weight connections. Applying the orthogonality equations to both of these ensures coherent and faster convergence.

4 Results and Discussions

The results of the experiments are as follows. The baseline is the original capsule network architecture which achieves maximum accuracy of 93.8% while the architecture with the above-suggested modifications achieves maximum accuracy of 96.3%. Hence, a relative increase of 3.4765% can be observed in the average accuracy values. The same are tabulated (see Table 1).

A visual representation of the same is as follows (see Fig. 2).

5 Conclusion and Future Work

It is evident from the above experimentation that implementation of weight orthogonality along with other optimizations and activation functions, ensure faster convergence and hence, lesser training time while enhancing reconstruction performance. Future work could entail aforementioned optimizations applied to more complex models or further enhancements to the architecture like experimenting with newer

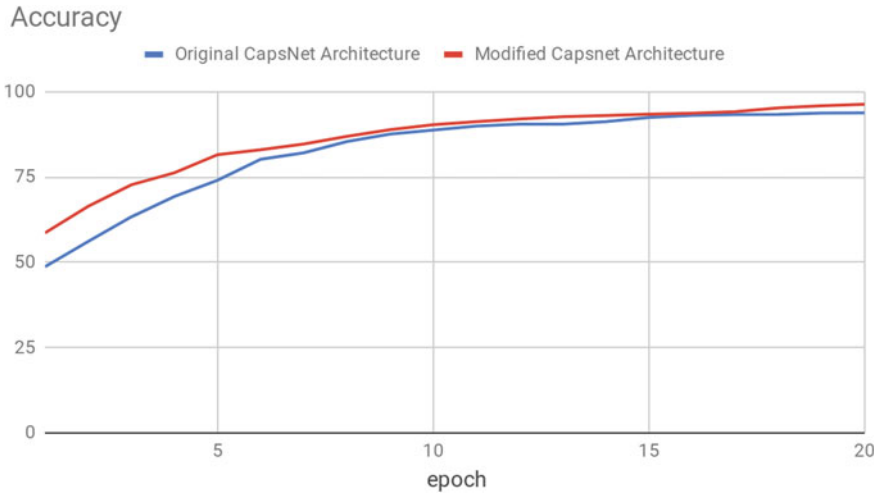


Fig. 2 Comparison between accuracy of original and modified Capsule Network architecture with reference to discussed dataset

techniques and methodologies. While the scientific community continues to explore capsule networks and other upcoming systems, the future could spur out challenging avenues to explore current and newer problems in interdisciplinary spaces as in case of nuclear variability in galaxies [29].

References

1. Sabour S, Frosst N, Hinton G (2017) Dynamic routing between capsules. In: Thirty first conference on neural information processing systems. [arXiv:1710.09829](https://arxiv.org/abs/1710.09829)
2. Hinton G, Sabour S, Frosst N (2018) Matrix capsules with EM routing. In: International conference on learning representations
3. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: International conference on machine learning, pp 1310–1318
4. Dauphin Y et al (2014) Identifying and attacking the saddle point problem in high dimensional non convex optimization. In: Advances in neural information processing systems, pp 2933–2941
5. Ioffe S et al (2015) Batch normalization: accelerating deep network training by reducing internal covariance shift. In: International conference on machine learning, pp 448–456
6. Bansal N, et al (2018) Can we gain more from orthogonality regularizations in training Deep CNN's? In: Advances in neural information processing systems. [arXiv:1810.09102](https://arxiv.org/abs/1810.09102)
7. Balestrierio R et al (2018) Mad max: affine spline insights into deep learning. [arXiv:1805.06576](https://arxiv.org/abs/1805.06576)
8. Zhu Z et al (2019) A convolutional neural network based on a capsule network with strong generalisation for bearing fault diagnosis. *Neurocomputing* 323: 62–75
9. Lin Z, Lu C, Li H (2015) Optimized projections for compressed sensing via direct mutual coherence minimization. [arXiv:1508.03117](https://arxiv.org/abs/1508.03117)
10. Zhong C (2019) Residual capsule networks. In: 11th annual undergraduate research symposium. Poster session. University of Wisconsin, Milwaukee

11. Jain R (2019) Improving performance and inference on audio classification tasks using capsule networks. [arXiv:1902.05069](https://arxiv.org/abs/1902.05069)
12. He S, Shin K (2019) Spatio-temporal capsule based reinforcement learning for mobility-on-demand network coordination. In: Association for computing machinery. ISBN: 978-1-4503-6674-8/19/05
13. Zhaopeng Xu, Wei Lu, Zhang Qqui, Yeung Yuileong, Chen Xin (2019) Gait recognition based on capsule network. *J Vis Commun Image Represent* 59:159–167
14. Yao H et al (2019) Capsule network assisted IoT traffic classification mechanism for smart cities. *IEEE Internet Things J*
15. Kim Y, Wang P, Zhu Y, Mihaylova L (2018) A capsule network for traffic speed prediction in complex road networks. In: Sensor data fusion: trends, solutions, applications. IEEE, Bonn, Germany. <https://doi.org/10.1109/sdf.2018.8547068>
16. Du Y, Zhao X, He M, Guo W (2019) A novel capsule based hybrid Neural network for sentiment classification. *IEEE Access* 7: 39321–39328. <https://doi.org/10.1109/access.2019.2906398>
17. Qiao K, Zhang C, Wang L, Yan B, Chen J, Zeng L, Tong L (2018) Accurate reconstruction of image stimuli from human fMRI based on the decoding model with capsule network architecture. [arXiv:1801.00602](https://arxiv.org/abs/1801.00602)
18. Mobiny A, Nguyer H (2018) Fast Capsnet for lung cancer screening. In: Frangi A, Schnabel J, Davatzikos C, Alberola Lopez C, Fichtinger G (eds) *Medical image computing and computer assisted intervention (MICCAI)*. Lecture notes in Computer Science, vol 11071. Springer, Cham
19. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feed forward neural networks. In: *Proceedings of thirteenth international conference on artificial intelligence and statistics*, pp 249–256
20. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on image classification. In: *Proceedings of IEEE international conference on Computer Vision*, pp 1026–1034
21. Vorontsov E, Trabelsi C, Kadoury S, Pal C (2017) On orthogonality and learning recurrent networks with long term dependencies. [arXiv:1702.00071](https://arxiv.org/abs/1702.00071)
22. Saxe A et al (2013) Exact solutions to the non linear dynamics of learning in deep linear neural networks. [arXiv:1312.6120](https://arxiv.org/abs/1312.6120)
23. Rodriguez P, et al (2016) Regularizing CNN’s with locally constrained decorrelations. [arXiv:1611.01967](https://arxiv.org/abs/1611.01967)
24. Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. [arXiv:1708.07747](https://arxiv.org/abs/1708.07747)
25. Mass AL et al (2013) Rectified non linearities improve neural network acoustic models. In: *International conference on machine learning*, vol 30
26. Gagana B, Natarajan S (2019) Hyperparameter optimizations for capsule networks. In: *EAI endorsed transactions on cloud systems*
27. Gagana B, Athri U, Natarajan S (2018) Activation function optimizations for capsule networks. In: *IEEE international conference on advances in computing, communications and informatics*. ISBN: 978-1-5386-5314-2
28. Le QV et al (2017) Swish: a self gated activation function. In: *Neural and evolutionary computing, computer vision and pattern recognition*. [arXiv:1710.05941](https://arxiv.org/abs/1710.05941)
29. Liang M et al (2015) Recurrent convolutional neural network for object recognition. In: *Conference on computer vision and pattern recognition*
30. Kingma D, Ba J (2015) Adam: a method for stochastic optimisation. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
31. Luo L, Xiong Y, Liu Y, Sun X (2019) Adaptive gradient methods with dynamic bound of learning rate. In: *International conference on learning representations*
32. Katebi R (2019) Examining extreme nuclear variability in the galaxies that host Active Galactic Nuclei. *American Astronomical Society*