PATH CAPSULE NETWORKS

A PREPRINT

Mohammed Amer[*]
School of Computer Science
University of Nottingham
Semenyih, Malaysia
hcxma1@nottingham.edu.my

Tomás Maul
School of Computer Science
University of Nottingham
Semenyih, Malaysia
tomas.maul@nottingham.edu.my

February 12, 2019

ABSTRACT

Capsule network (CapsNet) was introduced as an enhancement over convolutional neural networks, supplementing the latter's invariance properties with equivariance through pose estimation. CapsNet achieved a very decent performance with a shallow architecture and a significant reduction in parameters count. However, the width of the first layer in CapsNet is still contributing to a significant number of its parameters and the shallowness may be limiting the representational power of the capsules. To address these limitations, we introduce Path Capsule Network (PathCapsNet), a deep parallel multi-path version of CapsNet. We show that a judicious coordination of depth, max-pooling, regularization by DropCircuit and a new fan-in routing by agreement technique can achieve better or comparable results to CapsNet, while further reducing the parameter count significantly.

1 Introduction

Convolutional Neural Networks (CNNs) [Fukushima and Miyake, 1980, LeCun et al., 1995] have remained state-of-the-art in image processing and computer vision tasks since their successful large scale training by Krizhevsky et al. [2012]. CNNs were biologically inspired by the visual cortex [Hubel and Wiesel, 1968] and were built on the principle of translation invariance, achieved through local receptive fields, weight sharing and pooling operations. Despite their success, CNNs suffer from inherent limitations, most significantly the fact that translation invariance by definition causes loss of location information. This limitation has stimulated a lot of research in the direction of augmenting learning with location data [Wang and Veksler, 2018, Tang et al., 2015, Ghafoorian et al., 2017].

Sabour et al. [2017] argued that the main limitation of CNNs is the focus on achieving translation invariance, and that equivariance should also be targeted. Hence, the authors proposed CapsNet as a step towards achieving equivariance. The philosophy of CapsNet is that a single activation/feature should be replaced by a pose vector, named capsule, representing the different properties of an object's viewpoint. CapsNet has two main components, which are PrimaryCapsule and DigitCaps layers. PrimaryCapsules represent the different parts of the underlying objects, which are then multiplied by translation matrices to get prediction vectors, representing the votes of each PrimaryCapsule with respect to each DigitCaps, which are then routed using routing by agreement to compute DigitCaps activations, which can then be used to signify the presence of an object. The philosophy is that with changing the viewpoint of an object, the change in pose matrices should be coordinated, such that the voting agreement is maintained. We consider using another form of routing by agreement, fan-in routing in contrast to fan-out routing used by Sabour et al. [2017], which we show can have better performance under some conditions.

CapsNet was shown to achieve very good results with a shallow architecture and decent parameter savings, compared to deep CNNs. However, the lack of depth can be limiting to the expressiveness of the network. Moreover, the first convolutional layer in CapsNet is large and contributes to increasing the number of CapsNet parameters significantly. We believe that a coordinated inclusion of depth and multiple pathways can help increase the network performance and simultaneously help save more parameters.

---

[*]Corresponding author

We consider a multipath architecture for including more depth into CapsNet. Multiple paths in neural networks are biologically plausible and biological neural networks have been shown to exhibit multipath parallel processing [Gollisch and Meister, 2010, Otsuna et al., 2014]. Aside from biological inspiration, we think that using different paths for generating PrimaryCapsules can be exploited to enhance performance while saving parameters significantly. A PrimaryCapsule generated by a deep path can be considered a deep version of the original CapsNet capsule, which we believe can exhibit more expressiveness and more abstraction, similar to other deep structures in the deep learning paradigm.

The universal approximation theorem by Hornik et al. [1990] showed that any Borel measurable function can be approximated by a sufficiently wide single layer multilayer perceptron (MLP). Empirically, however, this is infeasible due to optimization limitations, and is rarely desirable due to the problem of overfitting. On the other hand, making use of depth is statistically motivated by composition of functions and empirically can lead to better generalization. Moreover, as we show, depth can be added judiciously to save parameters without sacrificing performance.

Our contributions in this paper are:

1. We propose PathCapsNet, a multipath deep version of CapsNet.
2. We enrich the routing by agreement methodology by a new variant, fan-in routing.
3. By carefully adding depth and max-pooling, along with a multi-path structure, fan-in routing and DropCircuit, we achieved comparable results to CapsNet with significant parameter savings.
4. We open the possibility of leveraging significant model parallelism in the context a capsule networks.

In the next section we discuss the previous work done around capsule networks and multipath architectures, and how we enhance by building on these concepts.

2   Related Work

CapsNet [Sabour et al., 2017] was introduced as an architecture that builds up on the conventional CNN [Fukushima and Miyake, 1980, LeCun et al., 1989] trying to overcome its limitations. The main motivation behind CapsNet is achieving equivariance, in addition to the invariance properties already implemented by CNN. CapsNet could achieve a good generalization using relatively fewer parameters than deep CNNs (only 8.2M parameters for the MNIST model with reconstruction). Different variants have been introduced since the original CapsNet. Phaye et al. [2018] introduced DCNet as a dense version of capsule networks and DCNet++ by stacking multiple DCNets. In DC-Net++, each DCNet in the stack produces its version of the PrimaryCapsule layer, which is then fed to the next DCNet in the stack. The final output is calculated based on both the output of each subnetwork and their concatenation. They also made some modifications to the decoder (reconstruction) subnetwork. DCNet++ achieved good generalization in relatively few epochs at the cost of using more parameters (13.4M).

Another variant is MS-CapsNet [Xiang et al., 2018]. MS-CapsNet is composed of three successive modules. The first module is the feature extractor and it has two convolutional paths of depths 1 and 2 and a third path which is just a skip connection. Each path produces a PrimaryCapsule of different dimension. The second module is a capsule encoding and it is responsible for projecting the PrimaryCapsules to a common dimension and concatenating them. The third module, capsule dropout, is applied before routing and it is responsible for dropping random capsules as a way of regularization in a manner similar to dropout [Srivastava et al., 2014] and other similar techniques. Capsule dropout showed enhancement in performance relative to the non-dropout condition. MS-CapsNet could achieve better performance than the original CapsNet on FashionMNIST and CIFAR-10 with fewer parameters ($\sim$11M).

SECaps [He et al., 2018] is an adaptation of CapsNet to sequential tasks, specifically Natural Language Processing (NLP). The word embeddings of single words are treated as PrimaryCapsules. Since the dynamic routing is not sequential in nature and doesn't respect order, the seq-caps layer is introduced. This layer is basically composed of a long short-term memory (LSTM) layer that is applied to a given sequence of the data as a series encoding, and then the output is dynamically routed in the conventional way to produce the output of the next layer. Multiple seq-caps layers can be stacked. Another module, the attention module, transforms the word embeddings, which are then concatenated with the seq-layer output. The final output is produced by an MLP subnetwork. SECaps was evaluated on multiple charge prediction datasets, achieving better performance than the state-of-the-art.

Siamese capsule network (SCN) [Neill, 2018] is the capsule version of the conventional siamese network. Neill [2018] introduced SCN as a face verification approach similar to DeepFace [Taigman et al., 2014]. SCN is very similar in architecture to the original CapsNet. It has a convolutional layer, followed by the PrimaryCapsules layer and then a layer called Face Capsule layer, which is essentially similar to the DigitCaps layer. The final output is produced by a

fully connected layer on top of the Face Capsule layer. SCN achieved good performance on different datasets with a smaller model, little preprocessing and less data.

Matrix capsules network was proposed by Hinton et al. [2018] as a generalization of the original CapsNet for more efficient pose estimation. Each capsule is represented by a matrix and a sigmoid unit that controls the probability of activating the capsule. Every pose matrix is multiplied by a transformation matrix to get the votes which will be used for routing to the next layer. Routing is done using expectation maximization (EM) that takes as input the votes and activation probabilities of the previous layer. Matrix capsules network achieved a very good accuracy improvement on the smallNORB dataset, a dataset that is highly viewpoint variant, but it seems that it doesn't have the same advantage on MNIST.

The ideas of branching, parallel computation and multiple paths are well established in the deep learning literature and have their supporting biological plausibility [Gollisch and Meister, 2010, Otsuna et al., 2014]. In [Cireşan et al., 2012], each path in a multi-path CNN is trained on a different preprocessing/distortion of the input image and the columns outputs are averaged to produce the final output. A similar approach is used in [Wang, 2015], but with different types of inputs which are the source image and a bilateral filtered version of it, and the outputs of the paths are integrated using fully connected layers. Szegedy et al. [2015a] proposed the Inception-v1 model , which was responsible for winning ILSVRC-14, and is composed of a highly branched multipath architecture. Szegedy et al. [2015b] further improved the design of Inception-v1 to produce Inception-v2&3 which exploit large scale branching and multiple paths even more. The Xception architecture [Chollet, 2016] is a further extension to the Inception family, that uses more branching based on separable convolutions. ResNetXt [Xie et al., 2016] and Residual Inception [Zhang et al., 2018] are extensions of ResNet [He et al., 2016] where the modular block is multipath instead of single path.

FractalNet [Larsson et al., 2016] is another type of architecture that has a recursive self-similar, highly branched structure. Parallel circuit networks, introduced by Phan et al. [2016], adopt an extensively multipath architecture, and have demonstrated generalization improvements using a dropping technique called DropCircuit [Phan et al., 2018]. Related to the DropCircuit technique is the path dropout used by Bender et al. [2018] to regularize the training of a one-shot model, which is an implicit form of a multipath network, where a whole space of possible branches is trained simultaneously.

We build on previous work by:

1. Adding representational power to PrimaryCapsules by generating each capsule using a deep path.
2. Enriching dynamic routing by agreement with a new fan-in variant.
3. Combining depth, a multipath architecture, DropCircuit, max-pooling and fan-in routing to obtain a level of performance congruent with the original CapsNet, with significant parameter savings.
4. Showing that max-pooling is not inherently contradictory with the CapsNet philosophy, and that it can be used to save parameters significantly without sacrificing nether performance nor pose awareness.

In the next section, we explain the general PathCapsNet architecture and the different pieces that contribute to its performance.

3   Methods

The original CapsNet [Sabour et al., 2017] has two main capsule types, namely the PrimaryCapsules and the Digit-Caps. PrimaryCapsules are formed by applying an initial convolution layer to produce 256 channels, then another set of convolutions, which are then rearranged into 32 8D PrimaryCapsules. PrimaryCapsules are then routed to the next DigitCaps layer using dynamic routing by agreement. In one variant of CapsNet, namely CapsNet with reconstruction, a reconstruction layer is learned on top of the DigitCaps layer to facilitate the learning of instantiation (or transformation) parameters and therefore enhance generalization.

PathCapsNet fig. 1 shares the upper part of CapsNet, starting from the PrimaryCapsules layer, through the DigitCaps layer and ending with a reconstruction layer if needed. However, PathCapsNet is fundamentally different in how the PrimaryCapsules are constructed. In PathCapsNet, each PrimaryCapsule is formed by a deep CNN, named a path. So, the input is fed into different CNNs (paths) and the output of each path comprises one PrimaryCapsule.

The experiments done by Phan et al. [2018] demonstrate enhanced generalization in multipath MLPs, named parallel circuits in their work, using a drop technique called DropCircuit. DropCircuit is an adaptation of dropout to multipath architectures, where different paths are dropped during training, using a pre-specified probability. This is believed to enhance generalization by promoting independence between paths, hence allowing for problem decomposition and learning more useful representations, similar to dropout [Srivastava et al., 2014] and related techniques.
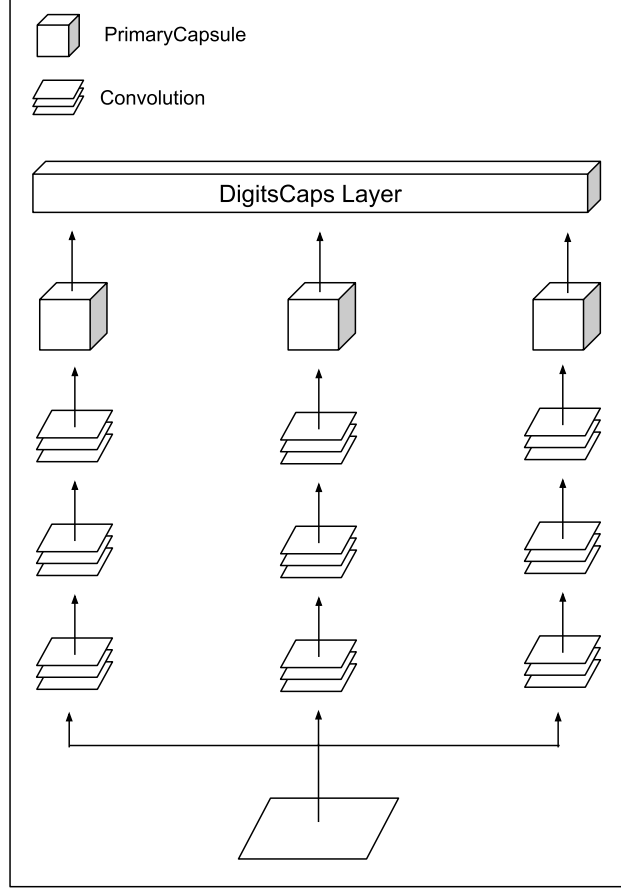
Figure 1: PathCapsNet architecture

Dynamic routing is the mechanism by which PrimaryCapsules are routed to DigitCaps capsules, such that similar votes from PrimaryCapsules contribute more strongly to the target DigitCaps. The dynamic routing by agreement algorithm used in [Sabour et al., 2017] updates the contribution of votes based on the similarity between the output DigitCaps and the prediction vector, representing the vote, using dot product as a measure of similarity. So, given the prediction vectors (votes) from the previous layer of capsules (PrimaryCapsule layer) $\hat{\mathbf{u}}_{j|i}$, where j is the index of the DigitCaps capsule and i is the index of a single capsule in the PrimaryCapsule layer, the output vector (DigitCaps) is calculated as,

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i} \tag{1}$$

where c_{ij} are the coupling coefficients weighting the contributions of different prediction vectors,

$$c_{ij}^{(fout)} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{2}$$

and b_{ij} is the log probability (logits) that the ith PrimaryCapsule should be coupled to the jth DigitCaps capsule. We call this fan-out (fout) routing, since the weights of the contributions of the ith PrimaryCapsule to each DigitCaps capsule in the next layer are normalized probabilities that sum to 1.0. For PathCapsNet, we used a different form of dynamic routing by agreement, named fan-in (fin) routing, where logits are normalized such that the weights of the contributions to the jth DigitCaps capsule from all the PrimaryCapsules are normalized probabilities that sum to 1.0. Coupling coefficients for fan-in routing are calculated as,

$$c_{ij}^{(fin)} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ki})} \tag{3}$$

The accuracy and reconstruction losses are calculated the same way as [Sabour et al., 2017], using margin loss and sum of squared errors loss, respectively.

In the next section, we present the details of our experimental design and the results we obtained.

4 Results

4.1 PathCapsNet Architecture

For all of our experiments, each path had the same architecture table 1. The number of paths in each experiment, however, varied and will be clarified for each set of experimental results. We will use the notation PathCapsNet-[num], where [num] is replaced by the number of paths, so PathCapsNet-5 is PathCapsNet with 5 paths. All PrimaryCapsules were 8D with spatial dimensions 7x7. As each path produces one PrimaryCapsule, the number of PrimaryCapsules is equal to the number of paths. The DigitCaps layer was exactly the same as Sabour et al. [2017]. We used 3 routing iterations in all the experiments and whenever we used fan-in routing, we initialized the transformation matrices of the DigitCaps layer randomly from a standard normal distribution. The Adam optimizer was used in all of the experiments using the default parameters and learning rate. When DropCircuit was used, the probability of path dropping was 0.5. Our benchmark was the original CapsNet [Sabour et al., 2017] with and without reconstruction and using 3 routing iterations. The benchmark was implemented using the same architecture as reported in the original paper without any modifications, unless otherwise specified. All reported results are based on an average of three trials, with 300 epochs of training each. This number of epochs is relatively small compared to the number used by Sabour et al. [2017], which seems to be more than 1000 epochs.

Layer	Type	Kernel	Padding	Stride	Output Channels
1	Conv	9	4	1	16
2	Conv	9	4	1	16
3	Maxpool	2	0	2	16
4	Conv	9	4	1	16
5	Conv	9	4	1	8
6	Maxpool	2	0	2	8

Table 1: Single path architecture

4.2 Experiments on MNIST

Our experiments were conducted on the MNIST dataset. We trained on 90% of the training dataset and left 10% for calculating validation performance, used for selecting the best model. The test performance is reported on the full test set. Following Sabour et al. [2017], the only augmentation used during training was padding by 2 and random cropping using a 28x28 patch. Our performance results on MNIST are summarized in table 2.

For the no-reconstruction setting, our fan-in routing improved CapsNet test error from 0.48% to 0.42%. A similar improvement was observed for PathCapsNet-5, where test error improved from 0.54% to 0.47%. With DropCircuit, we observed no improvement for a small number of paths, i.e PathCapsNet-5, while a significant improvement was observed for PathCapsNet-10, where the error improved from 0.52% to 0.42%, which is better than the standard CapsNet with only 21% of the parameters. A regularization effect can be observed from the validation curves fig. 2. For the reconstruction setting, CapsNet had the best validation error of 0.35%, however, we could achieve a very near performance of 0.38% with PathCapsNet-16 and DropCircuit with only 44% of the parameters.

In the next section we discuss our interpretations and hypotheses explaining the different techniques that contributed to these results and why we think that they enhance the current methodologies.

| No.	Architecture	Routing	Paths	DropCircuit	Parameters count	Parameters (%)	Test error (%)
No Reconstruction							
1	CapsNet	Fan-out	N/A	N/A	6.8M	100%	0.48 ± 0.02
2	CapsNet	Fan-in	N/A	N/A	6.8M	100%	**0.42**±0.03
3	PathCapsNet	Fan-out	5	Yes	683K	10%	0.54 ± 0.05
4	PathCapsNet	Fan-in	5	No	683K	10%	0.48 ± 0.07
5	PathCapsNet	Fan-in	5	Yes	683K	10%	0.47 ± 0.04
6	PathCapsNet	Fan-in	10	No	1.4M	21%	0.52 ± 0.03
7	PathCapsNet	Fan-in	10	Yes	1.4M	21%	**0.42**±0.05
Reconstruction							
1	CapsNet	Fan-out	N/A	N/A	8.2M	100%	**0.35**±0.04
2	CapsNet	Fan-in	N/A	N/A	8.2M	100%	0.47 ± 0.03
3	PathCapsNet	Fan-out	10	No	2.8M	34%	0.44 ± 0.06
4	PathCapsNet	Fan-in	10	No	2.8M	34%	0.47 ± 0.02
5	PathCapsNet	Fan-out	10	Yes	2.8M	34%	0.49 ± 0.02
6	PathCapsNet	Fan-in	10	Yes	2.8M	34%	0.42 ± 0.05
7	PathCapsNet	Fan-in	16	Yes	3.6M	44%	0.38 ± 0.02

Table 2: MNIST results
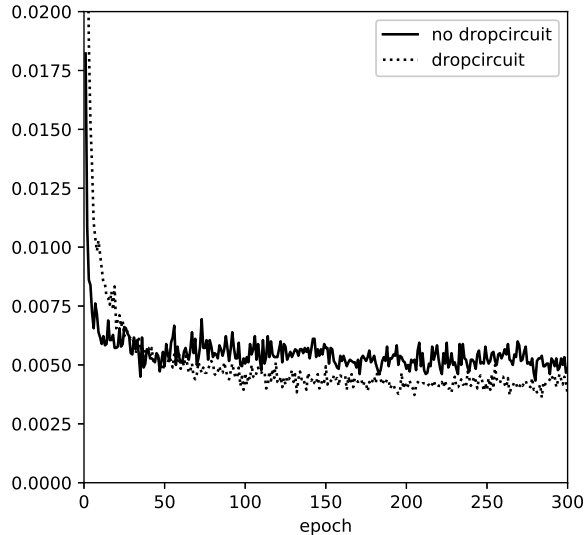
Figure 2: Average validation curves for PathCapsNet-10 with fan-in routing

5 Discussion

Three main components, we believe, contributed to the performance of PathCapsNet, namely deep PrimaryCapsules, fan-in routing and DropCircuit. Deep paths, even without fan-in routing and no DropCircuit, could achieve a decent test performance of 0.44% with only 34% parameters of the corresponding CapsNet (reconstruction condition 3 in table 2). We attribute this to the increased representational power of each PrimaryCapsule.

The main rationale behind fan-out routing was that each detected part of an object should contribute more strongly to a single object category rather than to multiple object categories. Fan-in routing, on the other hand, is expressing the idea that for a given object, different detected parts should contribute differently. While both philosophies can be seen to have different pros and cons, making each one optimal for a different set of contexts, empirically, we observed enhancement of generalization when using fan-in routing with DropCircuit on MNIST. One explanatory hypothesis may be that MNIST digits share most of the parts, which means that the problem is not about assigning a part to different digits, but how strongly the different parts contribute to a target digit.
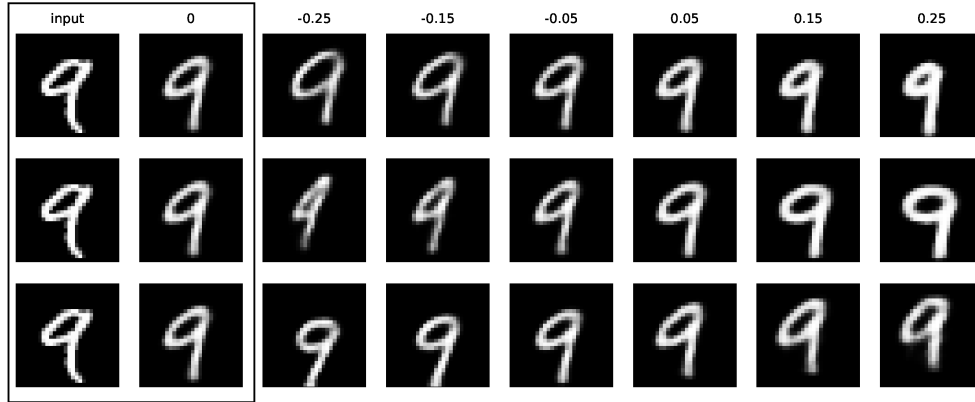
6

Figure 3: Perturbing different dimensions of PathCapsNet-10 (DropCircuit and fan-in). The images in the box are, from the left, the input and the unperturbed reconstruction, respectively.

DropCircuit, coupled with fan-in, showed remarkable performance enhancement for conditions with large numbers of paths, and no significant effect for small numbers of paths. We believe DropCircuit, being a form of regularization, needs a sufficiently large number of paths to show a positive effect. With a small number of paths, dropping becomes too destructive, specially with a high dropping rate, to show any significant improvement. We believe DropCircuit, like other drop techniques, is introducing independence between paths and promoting the extraction of more useful PrimaryCapsule representations.

During experimentation, we noticed that fan-in routing is usually more robust to using DropCircuit, resulting in more enhancement, specially in the reconstruction setting where using DropCircuit worsened the performance of fan-out routing. We think that this is essentially due to the difference in the softmax direction between fan-out and fan-in routing. DropCircuit effectively means that on average only a fraction of PrimaryCaps capsules, and hence prediction vectors contributing to a given DigitCaps capsule, exist at any given iteration. This introduces stochasticity in the fan-out prediction vector sum since the softmax is across DigitCaps capsules. This in effect will make the output of the DigtCaps layer noisy. This noise may not have a great effect on the norm of the capsules, which determines the object identity, but it makes it difficult for the reconstruction layer to function properly. On the other hand, since fan-in routing softmax is local to each DigitCaps capsule, this means that even with missing prediction vectors, on average it can converge to a more stable prediction vector sum and, hence, output.

We could achieve a performance comparable to CapsNet with significant parameter savings. This was possible thanks to a careful coordination of depth, multi-path structure and regularization. Essentially, we substituted the wide convolutional layer of CapsNet with deeper narrower paths regularized by DropCircuit. Fan-in routing enabled the effective utilization of DropCircuit regularization, since our experiments show that fan-out routing is less tolerant to DropCircuit (reconstruction cases 3 and 5 in table 2). Another component that contributed to reducing parameter counts was max-pooling which may be considered incompatible with CapsNet and its equivariance aim. However, we showed that it is possible to use max-pooling layers in PathCapsNet, which allowed further parameter savings without sacrificing performance.

Moreover, experiments perturbing different dimensions in the DigitCaps layer confirmed that, even when using max-pooling, different pose parameters can be successfully learned fig. 3. For example, perturbing the first three dimensions of the DigitCaps layer of one of the models fig. 3 suggested that the first dimension in the model was controlling multiple pose parameters, like the elongation of the circular and linear regions and stroke thickness. The second dimension seemed to affect the circle axes orientations and also stroke thickness, while the third dimension resulted in a combination of vertical translation, vertical axis inclination and elongation of the circular part.

6 Conclusion

We have introduced PathCapsNet, a multipath capsule network that can achieve better or comparable performance to CapsNet with significant parameter savings. In order to achieve this, we used regularization by DropCircuit along with a new variant of dynamic routing by agreement, fan-in routing. The careful coordination of depth, max-pooling,

fan-in routing and DropCircuit allowed for maintaining CapsNet performance, while cutting down parameter counts considerably. Reconstructions with perturbations showed that the use of max-pooling is not necessarily in conflict with retaining location information and pose estimation. The independence of paths renders the model suitable for model parallelism, a property which we didn't investigate in detail and we leave for future work. We think there is still more space for enhancing PathCapsNet, specially in the reconstruction setting where we believe there is a complex interaction between routing, DropCircuit and reconstruction.

References

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and Simplifying One-Shot Architecture Search, jul 2018. ISSN 1938-7228. URL http://proceedings.mlr.press/v80/bender18a.

François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. oct 2016. URL http://arxiv.org/abs/1610.02357.

Dan Cireşan, Ueli Meier, and Juergen Schmidhuber. Multi-column Deep Neural Networks for Image Classification. feb 2012. URL http://arxiv.org/abs/1202.2745.

Kunihiko Fukushima and Sei Miyake. Neocognitron: Self-organizing network capable of position-invariant recognition of patterns. In *Proc. 5th Int. Conf. Pattern Recognition*, volume 1, pages 459–461, 1980.

Mohsen Ghafoorian, Nico Karssemeijer, Tom Heskes, Inge W. M. van Uden, Clara I. Sanchez, Geert Litjens, Frank-Erik de Leeuw, Bram van Ginneken, Elena Marchiori, and Bram Platel. Location Sensitive Deep Convolutional Neural Networks for Segmentation of White Matter Hyperintensities. *Scientific Reports*, 7(1):5110, dec 2017. ISSN 2045-2322. doi: 10.1038/s41598-017-05300-5. URL http://www.nature.com/articles/s41598-017-05300-5.

Tim Gollisch and Markus Meister. Eye Smarter than Scientists Believed: Neural Computations in Circuits of the Retina, 2010. ISSN 08966273.

Congqing He, Li Peng, Yuquan Le, and Jiawei He. SECaps: A Sequence Enhanced Capsule Model for Charge Prediction. oct 2018. URL http://arxiv.org/abs/1810.04465.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with {EM} routing. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJWLfGWRb.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, jan 1990. ISSN 0893-6080. doi: 10.1016/0893-6080(90)90005-6. URL https://www.sciencedirect.com/science/article/pii/0893608090900056.

DH Hubel and TN Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 1968. URL http://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1968.sp008455/abstract?rss=1.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks, 2012. URL https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. FractalNet: Ultra-Deep Neural Networks without Residuals. may 2016. URL http://arxiv.org/abs/1605.07648.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, dec 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL http://www.mitpressjournals.org/doi/10.1162/neco.1989.1.4.541.

Yann LeCun, Yoshua Bengio, and Others. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

James O' Neill. Siamese Capsule Networks. may 2018. URL http://arxiv.org/abs/1805.07242.

Hideo Otsuna, Kazunori Shinomiya, and Kei Ito. Parallel neural pathways in higher visual centers of the Drosophila brain that mediate wavelength-specific behavior. *Frontiers in neural circuits*, 8:8, 2014. ISSN 1662-5110. doi: 10.3389/fncir.2014.00008. URL http://www.ncbi.nlm.nih.gov/pubmed/24574974http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=

Kien Tuong Phan, Tomas Henrique Maul, Tuong Thuy Vu, and Lai Weng Kin. Improving Neural Network Generalization by Combining Parallel Circuits with Dropout. dec 2016. doi: 10.1007/978-3-319-46675-0_63. URL http://arxiv.org/abs/1612.04970http://dx.doi.org/10.1007/978-3-319-46675-0{_}63.

Kien Tuong Phan, Tomas Henrique Maul, Tuong Thuy Vu, and Weng Kin Lai. DropCircuit : A Modular Regularizer for Parallel Circuit Networks. *Neural Processing Letters*, 47(3):841–858, jun 2018. ISSN 1370-4621. doi: 10.1007/s11063-017-9677-4. URL http://link.springer.com/10.1007/s11063-017-9677-4.

Sai Samarth R Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti Bathula. Dense and Diverse Capsule Networks: Making the Capsules Learn Better. may 2018. URL http://arxiv.org/abs/1805.04001.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic Routing Between Capsules. oct 2017. URL http://arxiv.org/abs/1710.09829.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. ISSN 15337928. doi: 10.1214/12-AOS1000.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 1–9, 2015a. ISBN 9781467369640. doi: 10.1109/CVPR.2015.7298594.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. dec 2015b. URL http://arxiv.org/abs/1512.00567.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, 2014. URL https://www.cv-foundation.org/openaccess/content{_}cvpr{_}2014/html/Taigman{_}DeepFace{_}Closing{_}th

Kevin Tang, Manohar Paluri, Li Fei-Fei, Rob Fergus, and Lubomir Bourdev. Improving Image Classification with Location Context. may 2015. URL https://arxiv.org/abs/1505.03873.

Mingming Wang. Multi-path Convolutional Neural Networks for Complex Image Classification. jun 2015.

Zhenyi Wang and Olga Veksler. Location Augmentation for CNN. jul 2018. URL https://arxiv.org/abs/1807.07044.

Canqun Xiang, Lu Zhang, Yi Tang, Wenbin Zou, and Chen Xu. MS-CapsNet: A Novel Multi-Scale Capsule Network. *IEEE Signal Processing Letters*, 25(12):1850–1854, dec 2018. ISSN 1070-9908. doi: 10.1109/LSP.2018.2873892. URL https://ieeexplore.ieee.org/document/8481393/.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

Xingpeng Zhang, Sheng Huang, Xiaohong Zhang, Wei Wang, Qiuli Wang, and Dan Yang. Residual Inception: A New Module Combining Modified Residual with Inception to Improve Network Performance. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3039–3043. IEEE, oct 2018. ISBN 978-1-4799-7061-2. doi: 10.1109/ICIP.2018.8451515. URL https://ieeexplore.ieee.org/document/8451515/.