Inspecting Matrix Capsule Network Configurations

Henrique Augusto Richter
Department of Informatics
Federal University of Paraná
Curitiba, Brazil. December, 2018
henrique.a.richter@gmail.com

***Abstract*—In this paper different configurations of Matrix Capsule Network were tested to see its convergence and capabilities in the Cifar-10 database. So far, the results indicate that Capsule Networks have potential, but there is still work to be done.**

***Index Terms*—Capsule Network, Machine Learning, Image Classification, Cifar-10**

I. Introduction

One of the main challenges in computer vision tasks is to correctly find and classify objects in an image. Since Krizhevsky et al. [1] showed that Convolutional Neural Networks (CNNs) can be used to achieve great performance in this kind of task, CNNs have become a popular research topic.

Even though CNNs have been improved over the years, they still have some flaws like losing information about the spatial relationship between entities in the image and lack of rotational invariance [2]–[5].

To address these flaws, Hinton et al. [6] showed the concept of capsules and how they could provide a way to recognize part-whole relationships and be viewpoint-invariant.

Using the Hinton et al. [6] idea of capsules, Sabour et al. [4] and Hinton et al. [5], presented two novel types of artificial neural networks called Capsule Networks (CapsNet). This new architecture achieved state-of-the-art performance on MNIST and smallNORB databases [7], [8].

Both versions of CapsNet use a routing-by-agreement mechanism, where a lower and a higher level capsule must agree with each other on the presence of an entity.

In the first version of CapsNet the outputs are vectors where the length of the vector represents the existence of an entity and each position of the vector represents its properties.

Whereas the second type of CapsNet presented by Hinton et al. [5] uses Matrix Capsules with the Expectation-Maximization algorithm (EM Routing), where the outputs are a scalar value that represents the presence of an entity and a 4x4 matrix which could learn the properties of the image and the relationships between entities.

Since this is a new architecture of Neural Network its behavior in different databases and with different parameters is yet to be known.

To further test how CapsNet with Matrix Capsules perform on more complex databases, this work aims to check its capabilities and convergence in the Cifar-10 database [9], with various hyperparameters and amount of layers.

II. Overview of Matrix Capsules

Hinton et al. [6] explains how CNNs tend to lose information about the pose of the objects in an image, and are not effective to learn the precise spatial relationships between entities, like orientation, perspective and size. For example, Fig. 1 shows two faces that could be wrongly classified by CNNs as human faces [10].

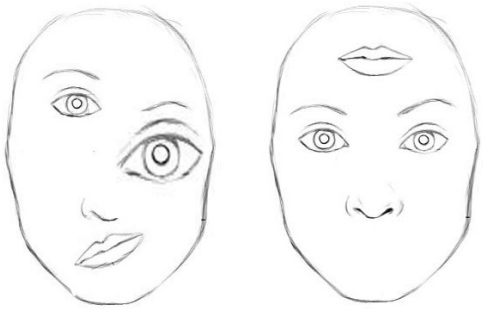

Fig. 1. Are these humans faces?

Another example of how CNNs can misclassify images, is what is called adversarial attack, a known technique where the CNN makes a wrong prediction when a small perturbation is added to the image and is imperceptible for humans but, as shown in Fig 2, it could make CNNs misclassify the image [11].

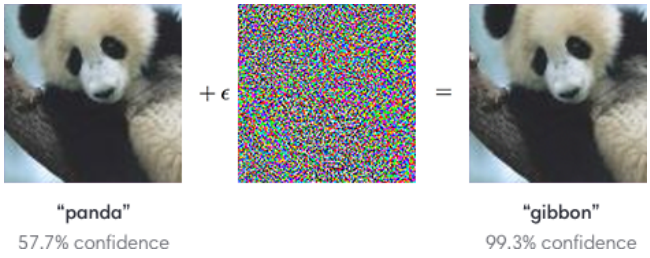

Fig. 2. Adversarial attack fooling CNNs.

CNNs perform well extracting features but are not very efficient at relating these features among each other. In Fig. 1 for example, a CNN could correctly extract the features of eyes, mouth, nose, but without information about size and position regarding to other entities, it could incorrectly activate the neurons related to human face detection, resulting in an incorrect classification.

To solve that Hinton et al. [6] presented the idea of "capsules" and how Artificial Neural Networks could use them instead of neurons.

Capsules are able to recognize wholes by recognizing their parts and learn not only if an object is present in the image but also could learn information like pose, size and lightning.

Fig. 3 shows the general architecture of a Capsule Network with Matrix Capsules presented by Hinton et al. [5].

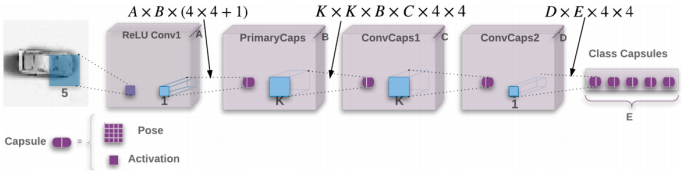

Fig. 3. Capsule Network.

The model in the general architecture (Fig. 3) starts with Relu Conv1 which is a 5x5 convolutional layer with 32 filters (A=32), with stride of 2 and ReLU as activation function. Followed by the PrimaryCaps layer that produces 32 capsules (B=32) with 4x4 pose matrix and an activation. The activation goes through a sigmoid activation function and represents if the capsule is activated or not.

ConvCaps1 is a convolutional capsule layer with filters of size 3x3 (K=3) and stride of 2.

ConvCaps2 is the same of ConvCaps1 but with stride of 1.

Both ConvCaps1 and ConvCaps2 have 32 capsules each (C=D=32).

The Class Capsules layer is one capsule per output class (E), Cifar-10 for example have 10 classes, so E=10.

This type of Capsule Network uses the Expectation-Maximization algorithm to do a dynamic routing between a lower and a higher level capsule, and it is applied in between ConvCaps1, ConvCaps2 and Class Capsules. In other words, each capsule in the lower layer corresponds to a data point, and each capsule in a the higher layer corresponds to a Gaussian [5] . Using Fig. 1 as example, a lower and a higher level capsule would have to agree that eyes, mouth and nose have the proper dimensions, orientation, size and location for the image to be correctly classified as a human face.

III. Database

The Cifar-10 database consists of 10 classes, airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck, with 6000 colored samples of each class and is organized in 50000 images for training and 10000 images for testing, with size of 32x32 pixels, some samples can be seen on Fig 4 [9]. This database was chosen because it's widely used to test computer vision algorithms and is suitable to test how Matrix Capsule Network perform on images more complex than the images on MNIST.

IV. Tests

For the experiments, the Cifar-10 database was used in its default configuration and no preprocessing or data augmentation techniques were used in the images.

The batch size was set to 8 images in the tests with 32 capsules in each layer (A=B=C=D=32), 6 images in the tests with 42 capsules in each layer (A=B=C=D=42) and 4 images in the tests with 64 capsules in each layer (A=B=C=D=64).

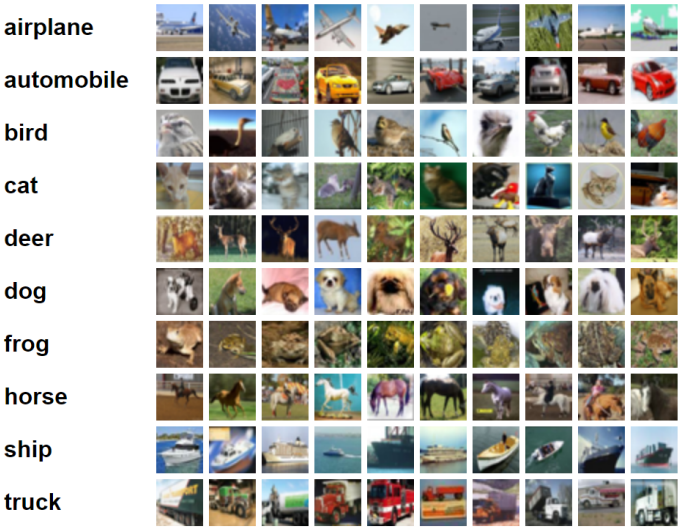

Fig. 4. Cifar-10 samples.

The different batch sizes were mainly due to limits of the GPUs memory. The same network configuration with different batch sizes was also tested, to check how much the batch size affect the results.

As a starting point, the first tests were made using the default configuration presented by Hinton et al. [5] (Fig. 3), which is, the input and hidden layers with 32 capsules (A=B=C=D=32) and the output layers with 10 class capsules (E=10).

The network was also tested with more layers (Y and Z), also with 32, 42 or 64 capsules in each layer.

Different learning rates were also tested, 0.01 and 0.001. Learning rate decay was not used in any of the tests.

The optimizer used for all the experiments was Adam, except on the optimizer tests, where Adam, RMSprop and SGD were tested.

V. Results and Discussion

In this section the results are presented and, in the end, some limitations encountered while doing this project are discussed.

For the first test, 2 and 3 iterations in the EM routing function were compared and the results are shown in Fig. 5.

With 2 iterations the top accuracy was 62.95% on epoch 15 on the test set and the network converged faster than with 3 iterations.

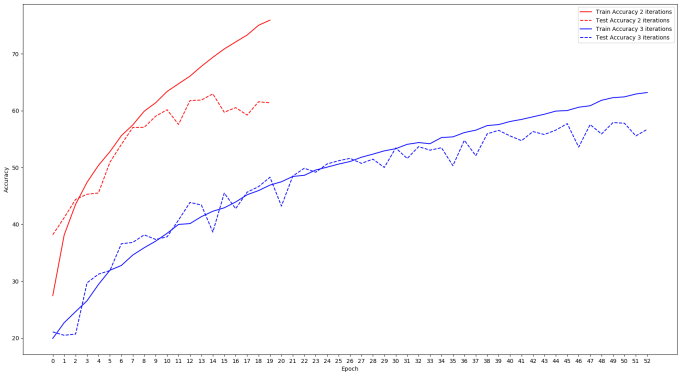

Fig. 5. Convergence with 2 and 3 iterations.

Fig. 6 shows the results for the tests with Adam, RMSprop and SGD optimizers, all of them with learning rate of 0.01.

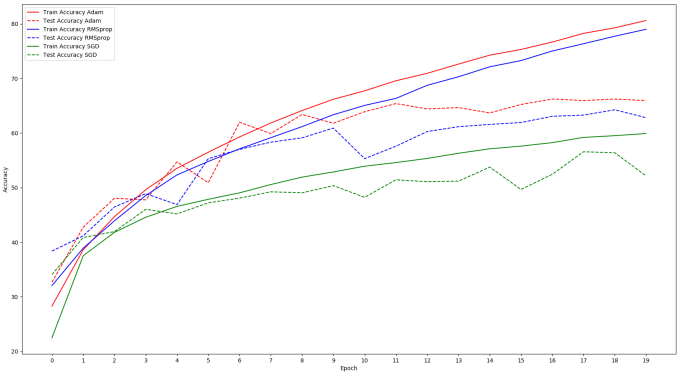

Fig. 6. Optimizers.

Adam optimizer had the best results with accuracy of 66.23% on epoch 17 while SGD had the worst convergence. Knowing that, all the other test were made with Adam optimizer.

The tests with different learning rates and how they impact on the network convergence are in Fig. 7.

Learning rate of 0.01 had best results on the test set and achieved 66.23% of accuracy on epoch 17.

Fig. 8 show the tests with 3, 4, 5 and 6 layers and 2 iterations.

With 3 layers the network had worse results. 4, 5 and 6 layers were similar, with 6 layers achieving 64.25% of accuracy on epoch 19.

The tests comparing 42 and 32 capsules in each layer are shown in Fig. 9.

With less capsules the convergence was better at the beginning. The top accuracy was 62.95%

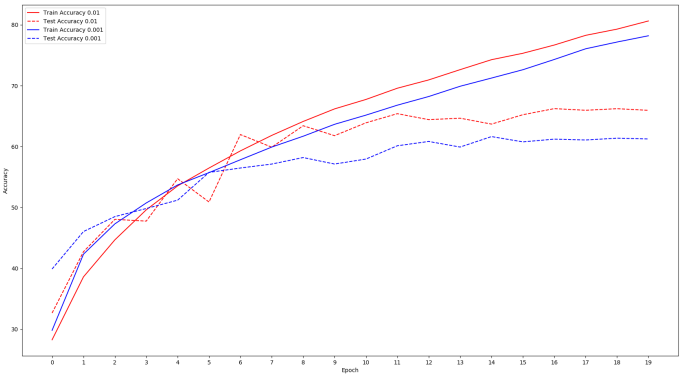

Fig. 7. Convergence with different learning rates.

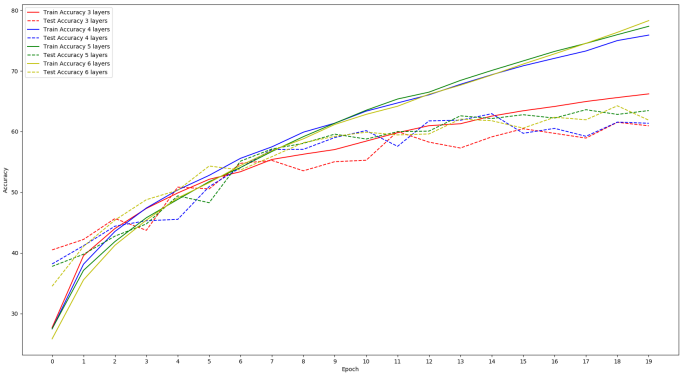

Fig. 8. Comparing 3, 4, 5 and 6 layers.

achieved by the network with 32 capsules on epoch 15.

To check if the number of images in a batch can affect the result, the network was also tested with multiple batch sizes with the same amount of capsules. Fig. 10 show the results.

The best accuracy, 62.95% on epoch 15, was

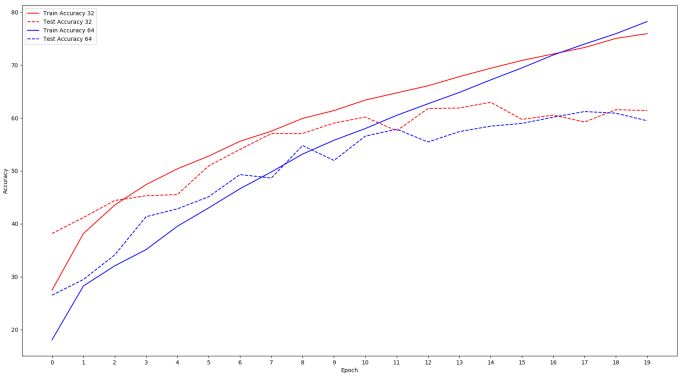

Fig. 9. Different number of capsules in each layer.

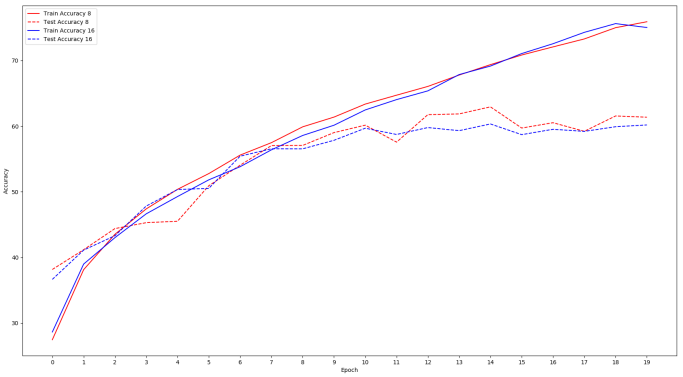

Fig. 10. Different batch sizes.

achieved with a batch size of 8, but on average, both batch sizes had similar results. That doesn't mean bigger batch sizes would have the same results, as Ian et al. [12] states, larger batches could provide a more precise estimate of the gradient, resulting in less error and better convergence.

All the test realized for this work can be seen in the Table I.

TABLE I
ALL THE TESTS.

Configuration					Test accuracy on each epoch (%)		
Layers	Iterations	Learning Rate / Optimizer	Batch Size	Top Accuracy (%) / epoch	1	10	20
A=B=C=D=32	2	0.01, Adam	8	62.95 / 15	38.17	59.02	61.37
			16	60.34 / 15	36.66	57.85	60.19
	3		8	57.9 / 50	21.1	37.35	48.29
		0.001, Adam		56.53 / 28	10.13	47.07	51.94
A=256, B=C=D=32	2	0.01, Adam		66.23 / 17	32.67	61.79	65.95
		0.001, Adam		61.64 / 15	39.9	57.13	61.25
		0.01, RMSprop		64.26 / 19	38.37	60.91	62.81
		0.01, SGD		56.57 / 18	34.06	50.36	52.14
A=B=C=D=64		0.01, Adam	4	61.2 / 18	26.52	51.96	59.45
A=B=C=32			8	61.53 / 19	40.49	55.04	60.95
A=B=C=D=Y=32	3			63.61 / 19	37.77	59.58	63.47
				58.59 / 169	16.38	19.98	26.67
		0.001, Adam		57.95 / 38	10.11	43.64	54.00
	2	0.1, Adam		57.46 / 30	22.18	45.28	54.21
		0.001, Adam		58.83 / 15	35.15	57.14	56.91
A=256, B=C=D=Y=32		0.01, Adam	6	63.32 / 23	32.73	56.16	61.5
			16	63.89 / 19	31.4	58.67	62.28
			8	66.14 / 25	35.15	59.42	64.36
A=B=C=D=Y=42			6	63.32 / 23	33.23	56.99	62.82
		0.001, Adam		59.82 / 13	35.07	57.09	59.25
A=B=C=D=Y=64		0.01, Adam	4	63.98 / 20	31.73	60.61	63.98
		0.001, Adam		60.76 / 16	28.92	56.38	59.11
A=B=C=D=Y=Z=32		0.01, Adam	8	64.25 / 19	34.5	59.21	61.88

What could be noticed is that normally with 2 iterations in the EM routing function the network converges faster than with 3 iterations. When using a learning rate of 0.001 instead of 0.01 the convergence is faster for 3 iterations, but for 2 iterations the results are worse.

The best accuracy overall was achieved by the network configuration with 256 capsules in the first layer (A=256) and 32 capsules in the following layers (B=C=D=32), EM routing with 2 iterations, optimizer Adam with learning rate of 0.01, and batch size of 8, reaching 66.23% on epoch 17.

The accuracy reported by the Matrix Capsules paper on Cifar-10 database was 88.1% [5], which is much lower than state-of-the-art results, that can go up to 98.52% of accuracy [13].

In the other hand, for MNIST and smallNORB databases, Matrix Capsules achieved state-of-the-art results with accuracy of 99.56% and 98.6% respectively [5].

One of the reasons for the difference in accuracy is that MNIST and smallNORB images are in grayscale and have no background, therefore are less complex than Cifar-10.

Increasing the number of capsules or layers could help to increase the accuracy, but to be able to do that, the GPU needs to have enough memory, or the algorithm used in this work needs to be optimized.

Since the GPU used for this work has only 11GB of memory, the size of the network could not be increased further without reducing the number of images in the batch, which may be the reason for the lower accuracy presented by this paper when compared to the result reported by Hinton et al. [5].

Despite of the low accuracy achieved in this work, Capsule Networks seems to have a great potential but more work still needs to be done regarding its performance and capabilities to recognize more complex images and in higher resolutions.

REFERENCES

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

[2] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2016. https://arxiv.org/pdf/1506.01497.pdf.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollr, Ross Girshick. Mask R-CNN. 2018. https://arxiv.org/pdf/1703.06870.pdf.

[4] Sara Sabour, Nicholas Frosst, Geoffrey E. Hinton. Dynamic Routing Between Capsules. 2017. https://arxiv.org/abs/1710.09829.

[5] Geoffrey Hinton, Sara Sabour, Nicholas Frosst. Matrix Capsules With EM Routing. 2018. https://openreview.net/pdf?id=HJWLfGWRb.

[6] G. E. Hinton, A. Krizhevsky, S. D. Wang. Transforming Auto-encoders. 2011. http://www.cs.toronto.edu/ fritz/absps/transauto6.pdf.

[7] Yann LeCun, Corinna Cortes, Christopher J.C. Burges. The mnist database of handwritten digits. 1998. http://yann.lecun.com/exdb/mnist/.

[8] Fu Jie Huang, Yann LeCun. The Small Norb Dataset. 2005. https://cs.nyu.edu/ ylclab/data/norb-v1.0-small/.

[9] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset. 2009. https://www.cs.toronto.edu/ kriz/cifar.html.

[10] Jonathan Hui. Understanding Matrix capsules with EM Routing (Based on Hinton's Capsule Networks). https://jhui.github.io/2017/11/14/Matrix-Capsules-with-EM-routing-Capsule-Network/.

[11] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, Pieter Abbeel, Jack Clark. Attacking Machine Learning with Adversarial Examples. 2017. https://blog.openai.com/adversarial-example-research/.

[12] Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning. 2016. http://www.deeplearningbook.org/.

[13] Ekin D. Cubuk, Barret Zoph, Dandelion Man, Vijay Vasudevan, Quoc V. Le. AutoAugment: Learning Augmentation Policies from Data. 2018. https://arxiv.org/pdf/1805.09501.pdf.