A Quick Summary of Capsule Networks with an Application on FashionMNIST

Cristobal Silva
Email: crsilva@ing.uchile.cl

***Abstract*—This work is an application of a novel approach to how neuron outputs in neural networks are represented. These neurons are called capsules and they output vectors instead of scalars. Using these units with a dynamic routing mechanism between layers, a clothes dataset was trained and evaluated to verify the properties of this new approach. Results are encouraging, as the new architecture, while not achieving state of the art results on the dataset, is capable of encoding several visual properties in their vector representation. Additionally, the shallow architecture is a step forward towards more efficient neural network architectures.**

I. Introduction

Deep Learning has been the trending concept in Machine Learning for some years already. This started in 2012 when a convolutional neural network architecture won the ImageNet competition [4], vastly outperforming state of the art methods of that time. After that event, many resources have been injected into this research area to develop new architectures and learning methods to enhance their capabilities.

While convolutional neural networks have many advantages in computer vision tasks, many of their properties are unnatural from a neurological perspective [3]. One of the culprits behind this are the max-pool layers used to route the output of a convolutional layer to the next layer. While efficient in terms of results, they lack several desired effects. For instance, a good property for a routing mechanism is a good use of the underlying linear structure, which handles the large sources of variance between images.

To address these issues, the notion of a "capsule" was developed. Capsules were conceptualized as early as 1981 by Hinton et. al. [2], and its main purpose is to do coincidence filtering. A capsule represents the presence and the instantiation parameters of a multi-dimensional entity of the type that capsule detects. This capsule should output two things: the probability that the entity is present, and the generalized pose of the object (position, scale, deformation, etc.).

The general idea is that capsules in a layer receive prediction vectors from capsules in the layer below. If these predictions agree with the output of the current layer, then the probability for the entity represented by that capsule increases.

For capsules to communicate with each other between layers, an efficient routing mechanism better than max-pooling is needed. This mechanism will be in charge to iteratively check whether the next capsules should activate or not, enforcing an clean path from input to output.

II. Capsule Network components and architecture

A. Capsules

Capsules are neurons whose output is a vector instead of a scalar value. The idea behind the capsule is that its length indicates whether an entity exists or not. On the other hand, the values inside the vector represent instantiation parameters of this entity, such as orientation, scale, thickness, etc.

We want the norm of the output vector of a capsule to be between 0 and 1 so the length can represent whether the entity exists or not. This constraint is enforced via the following non-linear function:

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|} \tag{1}$$

where $\mathbf{v}_j$ is the vector output of capsule j and $\mathbf{s}_j$ is its total input. This input is a weighted sum over all "predictions" $\hat{\mathbf{u}}_{j|i}$ done by previous layer capsules:

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i} \tag{2}$$

where c_{ij} are weights representing coupling coefficients that indicate how much affinity there is between the previous and the current capsules. The coefficients between capsule i and all the capsules in the next layer sum to 1.

The prediction vector $\hat{\mathbf{u}}_{j|i}$ is obtained via a transformation matrix $\mathbf{W_{ij}}$ that translates the dimensions of i to the dimensions of j:

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}^T\mathbf{u}_i \tag{3}$$

B. Dynamic Routing Mechanism

We know that the total input of a capsule depends on the sum of all previous layer capsules, weighted by coupling coefficients. These coefficients are determined during an iterative process in which a softmax calculates the log-probability that capsule i should be coupled with capsule j:

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{4}$$

where b_{ij} are initial logits that will be adjusted during the iterative process. We typically set these to 0 at the start and then update them in terms of the agreement a_{ij} between the next layer output $\mathbf{v}_j$ and the current layer prediction of the next layer output $\hat{\mathbf{u}}_{j|i}$:

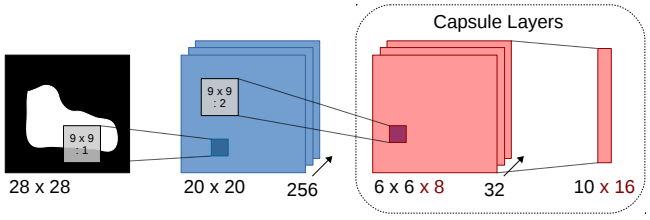

Fig. 1. CapsNet Architecture diagram. Blue layer represents a normal 2D convolutional layer. Red layers represent capsule layers, with the last dimension indicating the dimension of the activation vector.

$$a_{ij} = \mathbf{v}_j \cdot \hat{\mathbf{u}}_{j|i} \tag{5}$$

which is just a scalar product between the two vectors. The idea is that a_{ij} can be treated as a log-likelihood and can be added to the initial value of b_{ij}. The number of iterations for updating these values is a design parameter of the network.

The main idea is to check for fixed number of times how much the next capsule j is represented by the previous capsule i. While at first the previous capsule assigns equal probability to all following capsules, this is corrected each time the iteration happens. In the end, only a few capsules in the next layer will receive the output of the previous layer, effectively routing data only to the capsules that can make sense of it.

C. Loss Function

For evaluating the architecture, a margin loss for class existence is used. The idea is that the final output vectors of a capsule architecture represent each possible class. Because of these, we can directly use the length of such vectors to indicate whether a class is present or not. To allow multiple classes, a separate margin loss L_k is used for each class k:

$$\begin{aligned} L_k = T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \\ \lambda(1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2 \end{aligned} \tag{6}$$

where $T_k = 1$ if a digit of class k is present, $m^+ = 0.9$ and $m^- = 0.1$. $\lambda = 0.5$ is a down-weighting of the loss for absent digit classes. Total loss is calculated as the sum of the losses for all classes.

D. CapsNet Architecture

The vanilla architecture proposed by Sabour et. al. [1] consists of one convolutional layer, one capsule convolutional layer, and one capsule fully connected layer. Figure 1 shows a diagram of the architecture.

The initial convolutional layer outputs 256 channels using a size 9 kernel with a stride of 1. The purpose of this layer is to convert pixel intensities into low level features. These features are the input of the first capsules for them to start the inverse rendering process from simple entities.

The first capsule layer is constructed using a convolutional layer to generate 32 channels of 8D capsules. This is equivalent to generating 256 channels and then slicing them by 8 to generate 32 separate cubes. The idea behind a convolutional capsule is to preserve the good properties of convolutional layers (e.g., invariance to translations) while leveraging on the capsule representation benefits.

The second capsule layer is constructed using a fully connected layer to generate 10 nodes of 16D capsules. These capsules will represent each possible class and their existence will be given by the length of the activation vectors.

III. Experiments

Training was performed using the Fashion-MNIST dataset [5], as opposed to the original work using MNIST. The reason behind this change is to test the architecture against a more complex kind of data in terms of shape and texture.

Results can be seen in Table I for both 1 routing iteration and 3 routing iterations. Further routing iterations were not tested due to diminishing returns presented in the original work. We see that accuracy is competitive against other solutions[1], even though it is not the best performer.

TABLE I
Test Set results for the FashionMNIST dataset

Method	Test Accuracy
2 Conv Layers with max pooling	0.876
2 Conv Layers with 3 FC 1.8M parameters	0.932
WRN-28-10 + Random Erasing	**0.963**
CapsNet + 1 routing iteration	0.897
CapsNet + 3 routing iterations	0.927

Additional tests were performed to evaluate what do the output dimensions for each class represent. Results of these experiments can be found on Figure 2.

We see that most output dimensions represent some visual property of the class. For example, dimension 1 can represent the size of the shoe or whether it's a shoe or a boot. Dimension 13 represents the opposite, going towards a slipper shape. We also see that some dimensions truncate quicker than other if the same perturbations are done to them. This opens the possibility that some dimensions such as 11 and 14 can contain other transitions using lower perturbations. Finally, some dimensions can wrongly translate into other possible classes. For example, dimensions 4 and 8 quickly go from shoe to shirt after the perturbations.

IV. Discussion

For test set results, even though the score is not breaking the state of the art, the architecture is still an effective way to tackle visual classification problems. We also see better or equal performance compared to similar networks that use normal convolutions and max pooling. This indicates capsule dynamic routing is just as effective as max-pooling, while also retaining their good parameterization properties.

Regarding the number of iterations needed for dynamic routing, we see a consistent result in which more iterations

[1]Additional benchmarks can be found in the FashionMNIST respository: https://github.com/zalandoresearch/fashion-mnist

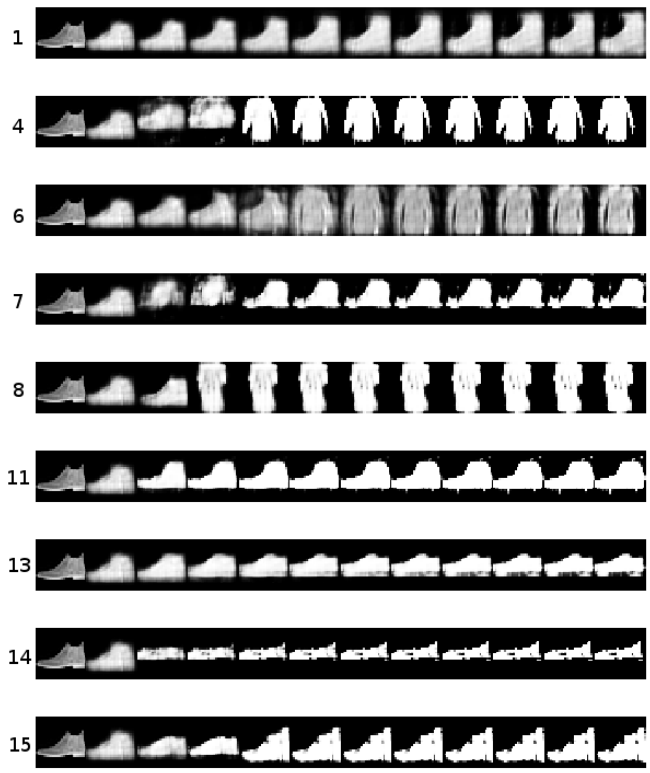

Fig. 2. Results of dimension perturbations for the shoe class. Each row shows the reconstruction when one of the 16 dimensions of the output layer are tweaked in intervals of 0.01 in the range $[0.0, 1.0]$. Not all output dimensions are shown in this image.

equals to better performance. This is because the more iterations, the more can the logits be updated to accurately route activations.

Finally, we confirm that output capsules can encode many properties of objects in the N-dimensional representation. While not perfect, the interpretation of the perturbations can vary between classes, as we see that some encode shape and size in different degrees.

V. Conclusions

This work presented an application of a novel view to neural network units using capsules with a dynamic routing mechanism. While this is just a re-implementation of the original work done by Sabour et. al., it serves both as a summary and proof of concept that the general idea and experiments can be extended to datasets other than MNIST.

In spite of not out-performing state of the art solutions, it has powerful representative properties that are encoded within the capsule dimensions. Further exploration of how perturbation can affect different classes should provide a more concrete view of how what each dimension learns to parameterize.

Future work may involve testing alternative architectures and different kinds of data. Particularly, it would be interesting to test this approach on bigger images and to explore generative properties of capsules themselves.

References

[1] S. Sabour and N. Frosst and G. E. Hinton, *Dynamic Routing Between Capsules*, Advances in Neural Information Processing Systems 30 (pre-proceedings), 2017

[2] G. E. Hinton, *A Parallel Computation that Assigns Canonical Object-Based Frames of Reference*, Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 1981

[3] G. E. Hinton and A. Krizhevsky and S. D. Wang, *Transforming Auto-encoders*, Artificial Neural Networks and Machine Learning, 2011

[4] A. Krizhevsky and I. Sutskever and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, Advances in Neural Information Processing Systems 25, 2012

[5] H. Xiao and K. Rasul and R. Vollgraf, *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*, arxiv pre-print, 2017