Dynamic Routing Between Capsules Review

Pavan Ravishankar(CS17S026) | github.com/prsettigunte

1 Summary

1.1 What is new?

This paper introduces a new algorithm called **Dynamic Routing** on **Capsules**[3]. It is inspired from inverse graphics where objects are represented relative to other objects and object matching occurs by deconstructing hierarchy of world in brain. Hierarchy of world is independent of **viewpoint**. Further there is relationship between parts to whole.

1.2 Why is it required?

Traditional neuron is capable of detecting variations but does not capture amount of variation nor does it incorporate hierarchical structure in its framework. Convolutional Neural Networks(CNN) captures translational variance but fails to detect other affine transformations. CNN does not capture hierarchical relationship thereby falling trap to adversaries. Max Pooling loses information which might be useful in few cases(overlapping digit detection). If it has to capture them, it requires lot of training examples which is certainly not the approach taken due to memorization and for the fact that brain does not operate that way.

1.3 What is capsule?

Capsules are **equivariant** as they not only detect but also capture amount of variation and **viewpoint invariant** as probability of detection is same when viewpoint changes. Each capsule captures parameters(like likelihood,width,angle,size etc.) of feature in a vector $\mathbf{u}$ whose magnitude represents probability of occurance of that feature.

1.4 How to compute in CapsNet?

Each capsule $\mathbf{u}_i$ in layer L is transformed by $\mathbf{W}_{ij}$ to get $\mathbf{u}_{ji}$=$\mathbf{W}_{ij}.\mathbf{u}_i$ that represents what parent node j in layer L+1 should be according to child node i. $\mathbf{u}_{ji}$ is multiplied with scalar c_{ij} depending upon how similar parent node is to what i^{th} node believes it to be. If similarity(calculated by dot product) is more c_{ij} is more, else less. Parent $\mathbf{v}_j$ is squashed output($\mathbf{v}_j$=$\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}.\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$) of weighted sum of what children nodes thinks it is($\mathbf{s}_j$=$\sum_i \mathbf{u}_{ji}.c_{ij}$). Squashing is required to ensure $\mathbf{v}_j \approx \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$(unit vector) for large $\|\mathbf{s}_j\|$ and $\approx \|\mathbf{s}_j\|.\mathbf{s}_j$ for small $\|\mathbf{s}_j\|$.

1.5 How to learn the parameters?

$\mathbf{W}_{ij}$ $\forall i \forall j$ across layers is learnt by backpropagation. Since $\mathbf{v}_j$ is dependent on c_{ij} and c_{ij} is dependent on $\mathbf{v}_j$, iterative algorithm is used to determine both of them by assuming initial value of c_{ij}.

1.6 Routing Agreement Algorithm

For each child node i, since $\sum_j c_{ij}$=1 and $c_{ij} \geq 0$, c_{ij} can be interpreted as probability distribution which is realised by softmax with $c_{ij} = \frac{e^{b_{ij}}}{\sum_k e^{b_{ik}}}$.

Steps
(1)Assume b_{ij}=0 $\forall i \in L, \forall j \in L+1$ layer.
(2)Calculate c_{ij} $\forall i \in L, \forall j \in L+1$
(3)Calculate $s_j = \sum_j c_{ij}.\mathbf{u}_{ji} \forall j \in L+1$
(4)$\mathbf{v}_j = \frac{\|\mathbf{S}_j\|^2}{1+\|\mathbf{S}_j\|^2} \cdot \frac{\mathbf{S}_j}{\|\mathbf{S}_j\|} \forall j \in L+1$
(5)Update $b_{ij} = b_{ij} + \mathbf{u}_{ji}.\mathbf{v}_j$ based on similarity of $\mathbf{u}_{ji}$ and $\mathbf{v}_j$ $\forall i \in L, \forall j \in L+1$
(6)Repeat (2) to (5) r times and return $\mathbf{v}_j$ $\forall j \in L+1$

Initially b values assumed to be 0 as lower level node is uncertain about higher level nodes, thereby giving equal weights to all of them. One interesting thing to observe is that, probability assigned to each capsule is independent of probability of other capsule(useful for overlapping digits detection).

1.7 What is Loss function used?

Experiment conducted is multi-digit classification. If a digit is present and probability of prediction for that digit $\geq m^+$, account for no error. If a digit is not present and probability of prediction for that digit $\leq m^-$, account for no error. Additional factor of λ is specified that controls importance one gives for wrong classification(not present but predicted with high probability). If λ is low, false digit capsules will stop learning.

$L_k(k^{th}$ digit capsule loss$) = T_k.\max(0, m^+ - ||\mathbf{v}_k||^2) + (1 - T_k).\max(0, ||\mathbf{v}_k||^2 - m^-)$

1.8 CapsNet Architecture

$28 \times 28 \times 1$ input image is fed into ReLU that use 256, 9×9 kernel with stride size 1 to get $20 \times 20 \times 256$ output. This is fed to PrimaryCapsules that uses 9×9 kernel with stride size 2 to get $6 \times 6 \times 256$ output. $6 \times 6 \times 256$ is divided into $6 \times 6 \times 32$ capsules of 8D. DigiCaps(CapsNet) applies $\mathbf{W}$ on each 8D capsule to get 16D vector as higher layers require more parameters to represent features. 16D vector is mapped to 10 16D vectors each of which represents a digit.

1.9 Regularization

To avoid overfitting during training, CapsNet is shown true output and corresponding capsule is reconstructed using Fully Connected ReLU(512 nodes), Fully Connected ReLU(1024 nodes) and Sigmoid(784 nodes). Squared Error in pixels of reconstructed image and original image is used to change the weights of entire network.

2 Drawbacks

2.1 Handling Poses

Dynamic routing only captures viewpoint invariance between child node and parent node with the help of weight vector but different poses are still modelled by different capsules. In Matrix capsules paper[2], pose matrix is introduced to explicitely handle poses.

2.2 Agreement across nodes

Dynamic routing links child to parent nodes depending on similarity of what the child thinks of parent and true value of parent node. It does not take into account relationship between lower level nodes. This is important to capture parts to whole relationship correctly. E.g. Right eye capsule and Left eye capsule might fit face well but when fitted together does not represent face at all. Matrix capsules paper[2] proposes clustering technique that takes this into account.

2.3 Modelling Background

CapsNet tries to fit background into child parent relationship. This influences weights and performance of network on the whole as mentioned in Sec 7[1] on CIFAR10 dataset whose background varies a lot.

2.4 Multiple Occurances of Same Digit

Current architecture of CapsNet, identifies multiple digits, segments multiple digits and reconstructs them but cannot handle same digit present mutiple times.

3 Future Research

3.1 Reinforcement Learning Perspective

Instead of having deep convolutional network as a function approximator of Q, we can replace it with Capsnet which has fewer parameters to learn, provides greater accuracy and performs well in situations where closely related pixels can mean completely different things.

3.2 Adversarial Attack

Algorithm proposed in paper is prone to Adversarial attack as experimented in [7]. Matrix Capsules paper [2] mentions that capsules are resistant to adversarial attack. By making it less prone to adversarial attacks one can leverage benefits of CapsNet completely. This can be a direction of research.

References

[1] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. *Dynamic Routing Between Capsules*. 31st Conference NIPS, Long Beach, CA, 2017.

[2] Anonymous. *Matrix capsules with EM routing*. 6th Conference, ICLR, Vancouver, Canada, 2018(Blind Review)

[3] G. E Hinton, A. Krizhevsky and S. D. Wang. *Transforming Auto-encoders*. Springer, International Conference on Artificial Neural Networks(44-51), Berlin, 2011

[4] Jonathan Hui Blog, *Understanding Dynamic Routing between Capsules*. Link

[5] Jonathan Hui Blog, *Understanding Matrix Capsules with EM routing*. Link

[6] Max Pechyonkin Blog, *Capsule Network Intuition, Working and Dynamic Routing*. Link

[7] Jaesik Yoon *Adversarial Attack to Capsule Networks*. Link