What is a CapsNet or Capsule Network?

What is a Capsule Network? What is a Capsule? Is CapsNet better than a Convolutional Neural Network (CNN)? In this article I will talk about all the above questions about CapsNet or Capsule Network released by Hinton.

Note: This article is not about pharmaceutical capsules. It is about Capsules in Neural Networks or Machine Learning world.

There is an expectation from you as a reader. You need to be aware of CNNs. If not, I would like you to go through this article on Hackernoon. Next I will run through a small recap of relevant points of CNN. That way you can easily grab on to the comparison done below. So without further ado lets dive in.

CNN are essentially a system where we stack a lot of neurons together. These networks have been proven to be exceptionally great at handling image classification problems. It would be hard to have a neural network map out all the pixels of an image since it's computationally really expensive. So convolutional is a method which helps you simplify the computation to a great extent without losing the essence of the data. Convolution is basically a lot of matrix multiplication and summation of those results.

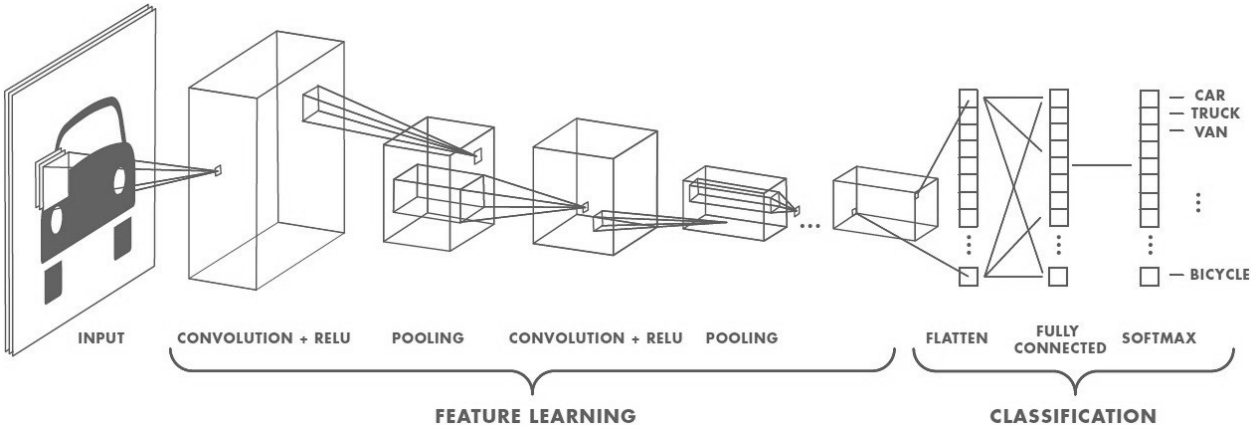

image 1.0: Convolutional Neural Network

After an image is fed to the network, a set of kernels or filters scan it and perform the convolution operation. This leads to creation of feature maps inside the network. These features next pass via activation layer and pooling layers in succession and then based on the number of layers in the network this continues. Activation layers are required to induce a sense of non linearityin the network (eg: ReLU). Pooling (eg: max pooling) helps in reducing the training time. The idea of pooling is that it creates "summaries" of each sub-region. It also gives you a little bit of positional and translational invariance in object detection. At the end of the network it will pass via a classifier like softmax classifier which will give us a class. Training happens based on back propagation of error matched against some labelled data. Non linearity also helps in solving the vanishing gradient in this step.

What is the problem with CNNs?

CNNs perform exceptionally great when they are classifying images which are very close to the data set. If the images have rotation, tilt or any other different orientation then CNNs have poor performance. This problem was solved by adding different variations of the same image during training. In CNN each layer understands an image at a much more granular level. Lets understand this with an example. If you are trying to classify ships and horses. The innermost layer or the 1st layer understands the small curves and edges. The 2nd layer might understand the straight lines or the smaller shapes, like the mast of a ship or the curvature of the entire tail. Higher up layers start understanding more complex shapes like the entire tail or the ship hull. Final layers try to see a more holistic picture like the entire ship or the entire horse. We use pooling after each layer to make it compute in reasonable time frames. But in essence it also loses out the positional data.

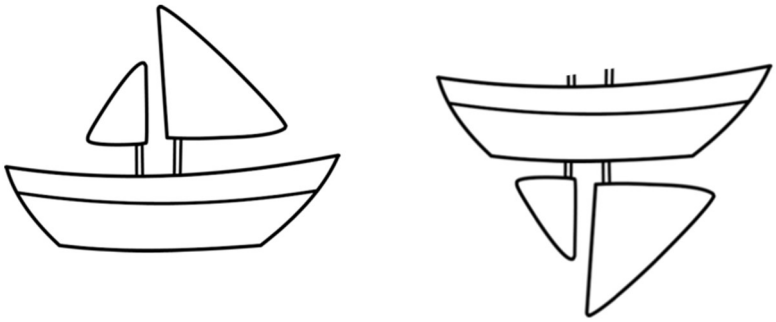

image 2.0: Disfiguration transformation

Pooling helps in creating the positional invariance. Otherwise CNNs would fit only for images or data which are very close to the training set. This invariance also leads to triggering false positive for images which have the components of a ship but not in the correct order. So the system can trigger the right to match with the left in the above image. You as an observer clearly see the difference. The pooling layer also adds this sort of invariance.

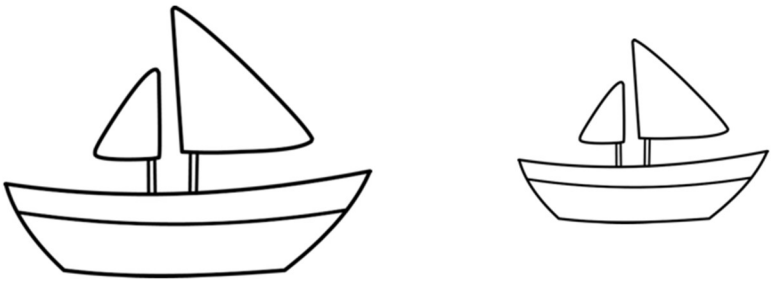

image 2.1 Proportional transformation

This was never the intention of pooling layer. What the pooling was supposed to do is to introduce positional, orientational, proportional invariances. But the method we use to get this uses is very crude. In reality it adds all sorts of positional invariance. Thus leading to the dilemma of detecting right ship in

image 2.0 as a correct ship. What we needed was not invariance but equivariance. Invariance makes a CNN tolerant to small changes in the viewpoint. **Equivariance** makes a CNN understand the rotation or proportion change and adapt itself accordingly so that the spatial positioning inside an image is not lost. A ship will still be a smaller ship but the CNN will reduce its size to detect that. This leads us to the recent advancement of Capsule Networks.

What is a Capsule Network?

Every few days there is an advancement in the field of Neural Networks. Some brilliant minds are working on this field. You can pretty much assume every paper on this topic is almost ground breaking or path changing. Sara Sabour, Nicholas Frost and Geoffrey Hinton released a paper titled ***"Dynamic Routing Between Capsules"*** 4 days back. Now when one of the Godfathers of Deep Learning "Geoffrey Hinton" is releasing a paper it is bound to be ground breaking. The entire Deep Learning community is going crazy on this paper as you read this article. So this paper talks about Capsules, CapsNet and a run on MNIST. MNIST is a database of tagged handwritten digit images. Results are showing a significant increase in performance in case of overlapped digits. The paper compares to the current state-of-the-art CNNs. In this paper the authors project that human brain have modules called "capsules". These capsules are particularly good at handling different types of visual stimulus and encoding things like pose (position, size, orientation), deformation, velocity, albedo, hue, texture etc. The brain must have a mechanism for "routing" low level visual information to what it believes is the best capsule for handling it.

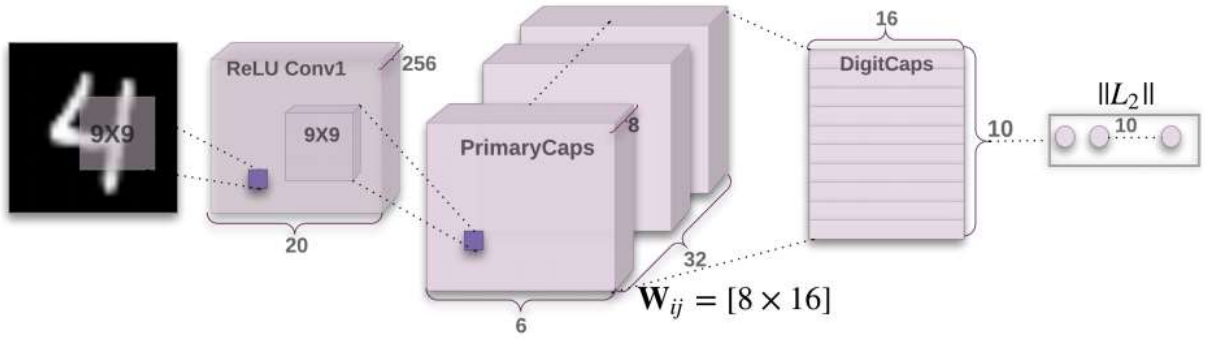

image 3.0: CapsNet Architecture

Capsule is a nested set of neural layers. So in a regular neural network you keep on adding more layers. In CapsNet you would add more layers inside a single layer. Or in other words nest a neural layer inside another. The state of the neurons inside a capsule capture the above properties of one entity inside an image. A capsule outputs a vector to represent the existence of the entity. The orientation of the vector represents the properties of the entity. The vector is sent to all possible parents in the neural network. For each possible parent a capsule can find a prediction vector. Prediction vector is calculated based on multiplying it's own weight and a weight matrix. Whichever parent has the largest scalar prediction vector product, increases the capsule bond. Rest of the parents decrease their bond. This **routing by agreement** method is superior than the current mechanism like max-pooling. Max pooling routes based on the strongest feature detected in the lower layer. Apart from dynamic routing, CapsNet talks about adding squashing to a capsule. Squashing is a non-linearity. So instead of adding squashing to each layer

like how you do in CNN, you add the squashing to a nested set of layers. So the squashing function gets applied to the vector output of each capsule.

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

image 3.1: Novel Squashing Function

The paper introduces a new squashing function. You can see it in image 3.1. ReLU or similar non linearity functions work well with single neurons. But the paper found that this squashing function works best with capsules. This tries to squash the length of output vector of a capsule. It squashes to 0 if it is a small vector and tries to limit the output vector to 1 if the vector is long. The dynamic routing adds some extra computation cost. But it definitely gives added advantage.

Now we need to realise that this paper is almost brand new and the concept of capsules is not throughly tested. It works on MNIST data but it still needs to be proven against much larger dataset across a variety of classes. There are already (within 4 days) updates on this paper who raise the following concerns:

1. It uses the length of the pose vector to represent the probability that the entity represented by a capsule is present. To keep the length less than 1 requires an unprincipled non-linearity that prevents there from being any sensible objective function that is minimized by the iterative routing procedure.

2. It uses the cosine of the angle between two pose vectors to measure their agreement for routing. Unlike the log variance of a Gaussian cluster, the cosine is not good at distinguishing between quite good agreement and very good agreement.

3. It uses a vector of length n rather than a matrix with n elements to represent a pose, so its transformation matrices have n 2 parameters rather than just n.

The current implementation of capsules has scope for improvement. But we should also keep in mind that the Hinton paper in the first place only says:

The aim of this paper is not to explore this whole space but to simply show that one fairly straightforward implementation works well and that dynamic routing helps.

So that's a lot of theory. Lets have some fun and build a CapsNet. I will take you through some code to setup a basic CapsNet for MNIST data. I will comment inside the code so you can follow through line by line and get an understanding of how it works. I will take you through two important pieces in the code.