CAPSULE NETWORK: A MORE RIGOROUS APPROACH FOR IMAGE RECOGNITION

Mohd Azfar Nazim
Electrical and Computer Engineering Dept.
University of Waterloo
Waterloo, Canada
manazim@uwaterloo.ca

Md Omor Faruk
Electrical and Computer Engineering Dept.
University of Waterloo
Waterloo, Canada
mofaruk@uwaterloo.ca

***Abstract*— To classify an image appropriately from a dataset is being considered as a central dilemma since the birth of machine learning. Throughout the timeline of machine intelligence, researchers have presented many solutions in which Convolutional Neural Network (CNN) is considered the most feasible accomplishment for image recognition. Afterwards, CNNs have been tuned in different manners progressively to achieve better result in different image datasets. However, CNNs encounter problems when they deal with images with spatial relationship between the sub-objects and eventually trigger false positive for images with the sub-objects placed in incorrect order. Researchers initially resolved the problem by putting more similar images into training set taken from different viewpoints. Recently, capsule network has been introduced which can preserve more information of the sub-objects of an image. A capsule is a group of neurons whose outcomes present various properties of a similar element. It supports equivariance. By applying multi-layer capsule framework, high accuracy on MNIST can be accomplished compared to a convolutional net at perceiving very overlapping digits. Additionally, to accomplish these results an iterative routing agreement is utilized. In this paper, we have investigated the architecture of CNNs as well as Capsule Networks to build a hypothesis whether Capsule Network is an evolvement of CNNs or not.**

Keywords—Convolutional Neural Network, Capsule, Capsule Network, Equivariance, Routing by agreement

I. INTRODUCTION

Convolutional Neural Networks (CNNs) have been hugely successful in the field of deep learning and they are the primary reason why deep learning is so popular right now. 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 [1]. They have been very successful, but they have drawbacks in their basic architecture, causing them to not work very well for some tasks. CNN's detect features in images and learn how to recognize objects with this information. There is no spatial information that is used anywhere in a CNN and the pooling function that is used to connect layers, is really inefficient [2].

II. BACKGROUND

A. *Convolutional Neural Ntwork (CNN)*

Convolutional Neural Networks (CNN) are as often as possible solution approach in computer vision applications on account of their victories on object detection and classification tasks. CNNs are made out of numerous neurons stacked together. Figuring convolutions crosswise over neurons require a great deal of calculation, so pooling procedures are frequently used to lessen the size of system layers. Convolutional methodologies make it conceivable to learn numerous mind boggling highlights of our information with basic calculations. By performing numerous network augmentations and summations on our information, we can reach at a solution to our inquiry [2].

A neural network that has one or various convolutional layers is called Convolutional Neural Network (CNN) [3]. We should consider a case of a deep convolutional neural network for image classification shown in fig. 1where the input picture size is 28 x 28 x 1. In the primary layer, we apply the convolution activity with 32 channels of 5 x 5 so our output will wind up 24 x 24 x 32. At that point, we will apply pooling with 2 x 2 channel to lessen the size to 12 x 12 x 32. In the subsequent layer, we will apply the convolution activity with 64 channels of size 5 x 5. The output measurements will wind up 8 x 8 x 64 on which we will apply pooling layer with 2 x 2 channel and the size will lessen to 4 x 4 x 64. At long last, we will go it through two completely associated layers to change over our image matrix into an classification matrix.

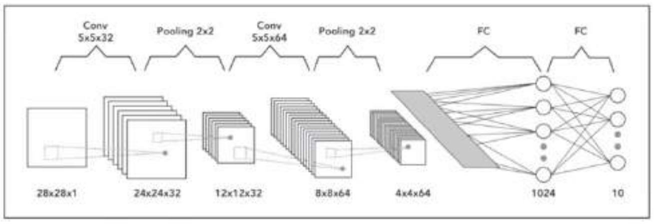


Fig.1: Operation of Convolutional Neural Network

B. *CNN Limitations*

It is a fact that CNNs have shown great success in solving object recognition and classification problems. However, they aren't perfect. One of the big challenges with convolving is that our image will continuously shrink if we perform convolutional operations in multiple layers. Let's say if we have 100 hidden layers in our deep neural network and we perform convolution operation in every layer than our image size will shrink a little bit after each convolutional layer [4][5]. The second downside is that the pixels from the corner of the image will be used in few outputs only whereas the middle region pixels contribute more so we lose data from the corners of our original image. However, these issues have been addressed in progressively tuned CNNs lately. But If a CNN is shown an object in an orientation it's unfamiliar with or where objects appear in places that it's not used to, it's prediction task will likely fail (fig. 2).

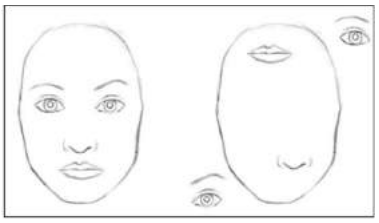


Fig.2: Mouth, nose have switched places, but CNN still classifies it as face

For instance, if we turn a face upside down, the network will no longer be able to recognize eyes, a nose, a mouth, and the spatial relationship between the two. Similarly, if we alter specific regions of the face (i.e. switch the positions of the eyes and nose), the network will be able to recognize the face— but it is no longer a real face.

As we can see from Fig. 1, pooling actually downsampled the feature size to half of the input feature size. Generally Max Pooling (or any kind of Pooling) is used to downsample the feature size to a manageable level. Besides reducing the size of the feature vector, it has some other ideas behind it. By only considering the max, we essentially are only interested if a feature is present in a certain window, but we do not really care about the exact location. If we have a convolution filter, which detects edges, an edge will give a high response to this filter and the Max Pooling only keeps the information if an edge is present and throws away "unnecessary" information; which also include the location and the spatial relationship between certain features. Since Max Pooling does not care about the spatial relationships (it throws the information away), the mere presence of certain features can be a good indicator of the presence of the object. However, because it does not care about the spatial relationship it would say, that the image on the right in Fig. 2 is also a face.

In a nutshell, CNNs learn statistical patterns in images, but they don't learn fundamental concepts about what makes something [4].

III. Improvements to CNNs

As described earlier, CNN had few problems of itself. To counter those and make the model more robust, several improvements have been made to CNN.

A. *AlexNet [1][10]*

Krizhevsky introduced better non-linearity in the network with the ReLU activation, whose derivative is 0 if the feature is below 0 and 1 for positive values. This proved to be efficient for gradient propagation. He further introduced the concept of dropout as regularization. From a representation point of view, forcing the network to forget things at random, so that it can see he next input data from a better perspective. He also introduced data augmentation. When fed to the network, images are shown with random translation, rotation, crop. That way, it forces the network to be more aware of the attributes of the images, rather than the images themselves. Overall the network was deeper as can be seen in Fig.3. They stacked more convolutional layers before pooling operations. The representation captures consequently finer features that reveal to be useful for classification.

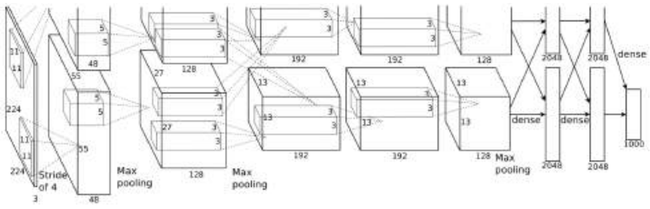


Fig.3: Basic structure of AlexNet

B. *GoogleNet [8][10]*

GoogleNet comprised of convolutions with different filter sizes which are processed on the same input, and then concatenated together. This allows the model to take advantage of multi-level feature extraction at each step. For example, general features can be extracted by the 5x5 filters at the same time that more local features are captured by the 3x3 convolutions. Basic architecture of GoogleNet can be shown in Fig.4.

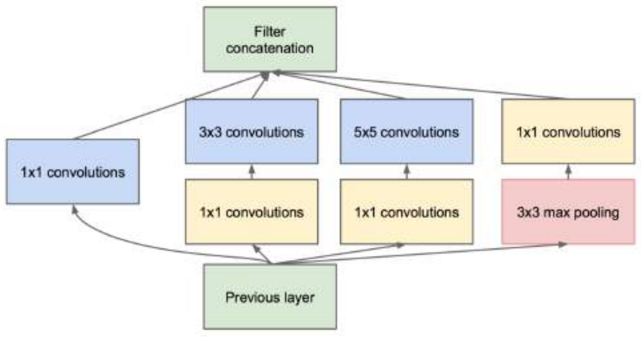


Fig.4: Basic structure of GoogleNet

C. *ResNet [10]*

At some stage, it was realized that stacking more layers does not lead to better performance. In fact, the exact opposite occurs because of the Gradient. In every two layers of ResNet, there is an identity mapping via an element-wise addition (Fig.5). This proved to be very helpful for gradient propagation, as the error can be backpropagated through multiple paths. This helps to combine different levels of features at each step of the network.

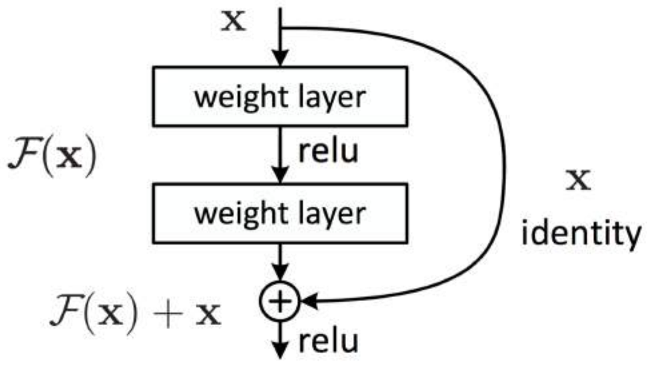

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig.5: Basic architecture of ResNet

IV. Capsules

A capsule is an abstract idea of having a group of neurons with an activity vector that contains more information about the object. According to Hinton et al, capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part [6].

Firstly, a capsule receives an input vector which represents the different information of a sub- object from an object. Secondly, affine transformation is applied to encode spatial relationships between sub-object of the object [6][7]. For instance, to determine a an image as a dog's face, one matrix could represent the information that the right eye is in the right top part of the dog face, the dog face should be about 10 times bigger, and the angle of the face should be the same as the angle of the eye. This matrix aims to transform the input vector into the position of the predicted output vector representing the next level — the face (higher level feature). After this matrix multiplication, we get a predicted position of the face.

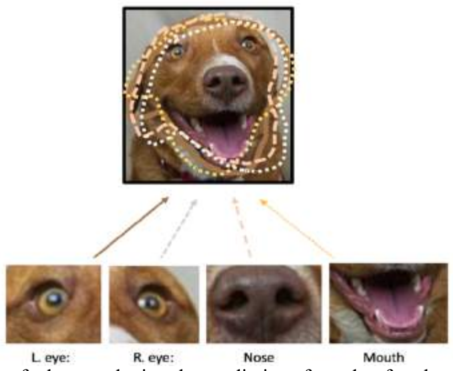


Fig.6: Example of what overlaying the predictions for a dog face based on each input

This happens from each vector applying weighted sum by the weights, learned by the routing algorithm "squashing" it to 0–1 with the nonlinear "squashing" activation function to get new vector, ready to be sent onwards.

A. *Equivariance*

One key feature of Capsule Networks is that they preserve detailed information about the object's location and its pose, throughout the network [9]. For example, as seen in Fig.7, if we rotate the image slightly, we notice that the activation vectors also change slightly.

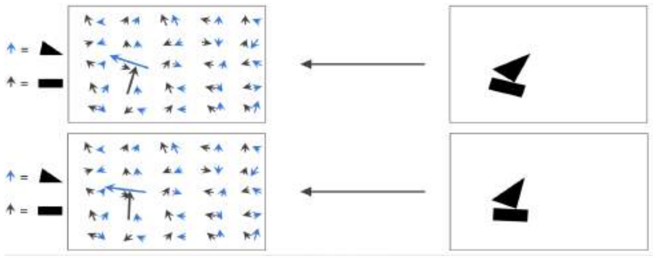


Fig.7: Equivariance in capsules [9]

If we compare the top right picture of the boat with the bottom right one, we can see that they are slightly rotated. Now, in the corresponding capsule network we can find that the corresponding capsules also rotated accordingly to preserve the pose of the picture. This is called equivariance. In a regular convolutional neural net, there are generally several pooling layers, and unfortunately these pooling layers tend to lose information, such as the precise location and pose of the objects. It's really not a big deal if we just want to classify the whole image, but it makes it challenging to perform accurate image segmentation or object detection which require precise location and pose. The fact that capsules are equivariant makes them very promising for these applications.

B. *Hierarchy of parts*

Capsule networks can handle objects that are composed of a hierarchy of parts. For example, let us consider a boat centered at position x=22 and y=28, and rotated by 16° (Fig.8). This boat is composed of parts. In this case one rectangle and one triangle. So, this is how it would be rendered. Now we want to do the reverse, we want inverse graphics, so we want to go from the image to this whole hierarchy of parts with their instantiation parameters.

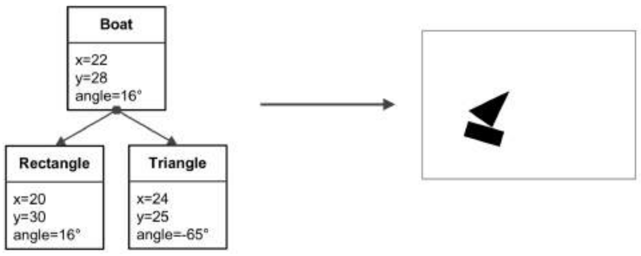


Fig.8: Hierarchy of parts for Boat [9]

Similarly, we could also draw a house, using the same parts, a rectangle and a triangle, but this time organized in a different way. Similar to the boat, the house is also centered at position x=22 and y=28, but the rotation is now -5° (Fig.9).

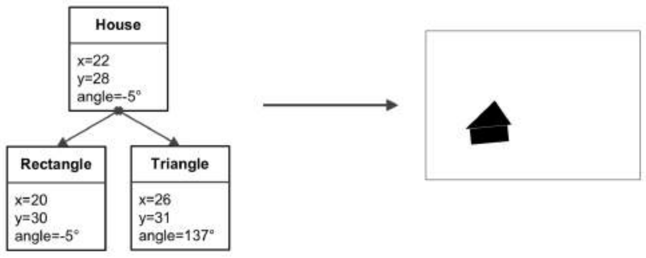


Fig.9: Hierarchy of parts for House [9]

The way to find out that the capsules preserve hierarchy of parts can be proved by the technique routing by agreement which we will discuss in later part of this paper.

V. Input and Output

Two aspects (length and orientation) of the output vector of the capsule are very important. The length represents the probability that the entity represented by the capsule is present in the current input. The orientation represents the object's estimated pose (rotation, thickness, translation etc.) parameters [9]. In practice, a good way to implement this is to first apply a couple of convolutional layers, just like in a regular convolutional neural net. This will output an array containing a bunch of feature maps. We can then reshape this array to get a set of vectors for each location. For example, suppose the convolutional layers output an array containing, say, 18 feature maps (2 times 9), we can easily reshape this array to get 2 vectors of 9 dimensions each, for every location. We can also get 3 vectors of 6 dimensions each, and so on.

The last step is to ensure that no vector is longer than 1, since the vector's length is meant to represent a probability, it cannot be greater than 1. We therefore use a non-linear "squashing" function mentioned below. It preserves the vector's orientation, but it squashes it to ensure that its length is between 0 and 1.

$$V_j = \frac{\|S_j\|^2}{1+\|S_j\|^2}\frac{S_j}{\|S_j\|} \dots\dots\dots\dots\dots (1)$$

where V_j is the vector output of capsule j and S_j is its total input [6].

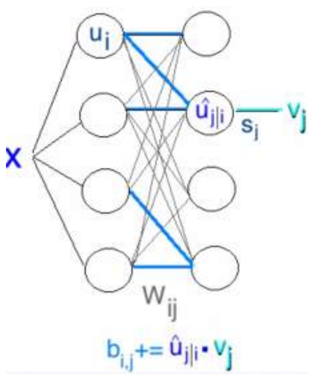


Fig.10: Dynamic routing of the data [11].

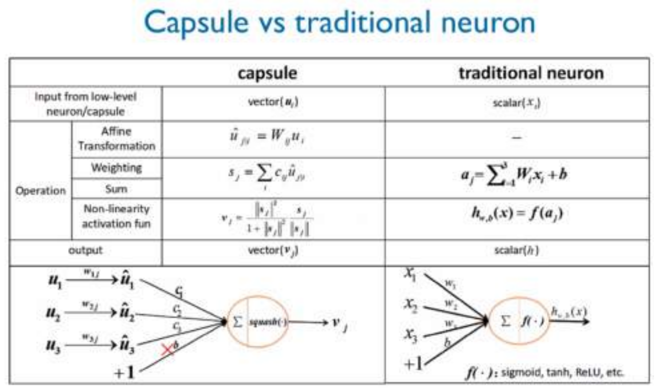
Let's assume the network in Fig.10. We have the input X which goes to the first layer called the primary capsules. When data comes in, the primary capsules are activated we get we get a u and we then compute the weight of the routes. For all the u we compute their mean weighted prediction with W matrix. The network will learn this W matrix through gradient descent. We start with a W and we get a mean value for j given all the u we get the mean value.

$$\widehat{u_{ji}} = W_{ij} u_i \ldots\ldots\ldots\ldots\ldots (2)$$

Next, we compute $b_{i,j}$ which iteratively is just an updated equation between u_{ij} and the output V_j. S_j is the weighted sum. Initially we guess b_{ij} and use it to find the c_{ij} which is the softmax function. Our initial W goes into the u bar and we get S_j.

$$s_j = \sum_i c_{ij} \widehat{u_{ji}} \ldots\ldots\ldots\ldots\ldots (3)$$

We then apply the squashing function mentioned above and we get a V which is our output. We then do this again and again and we update the b.


Capsule vs traditional neuron

		capsule	traditional neuron
Input from low-level neuron/capsule		vector(u_i)	scalar(x_i)
Operation	Affine Transformation	$\hat{u}_{j\|i} = W_{ij} u_i$	–
	Weighting	$s_j = \sum_i c_{ij} \hat{u}_{j\|i}$	$a_j = \sum_{i=1}^{3} W_i x_i + b$
	Sum		
	Non-linearity activation fun	$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2} \frac{s_j}{\|s_j\|}$	$h_{w,b}(x) = f(a_j)$
output		vector(v_j)	scalar(h)

u_1
w_{1j}
$\hat{u}_1$
u_2
w_{2j}
$\hat{u}_2$
u_3
w_{3j}
$\hat{u}_3$
+1
c_1
c_2
c_3
b
Σ squash(·)
v_j
x_1
x_2
x_3
+1
w_1
w_2
w_3
b
Σ f(·)
$h_{w,b}(x)$
f(·): sigmoid, tanh, ReLU, etc.

Fig. 11: Difference in architecture between a capsule and traditional neuron [11].

If we compare a capsule with traditional neuron as in Fig.11, we can figure out the main difference. In a traditional network we have a X in the input which is a scalar but the input in a capsule is a vector. We then apply an affine transformation which we don't have in traditional neurons.

We then apply a weighting and we say that S_{ji} is a weighted sum of the average u or affine transformation. This is just like weight updating of traditional neuron W_iX_i. The only difference is that in capsule we don't deal with the bias.

Then we apply a non-linearity. In a traditional neuron this is usually a sigmoid maybe it's a tan H or a RELU. In a capsule we apply a squashing function and the squashing function is the type of vector non-linearity and it applies to vectors not the scalars. The output of our capsule is now a vector whereas the output of a traditional neuron is a scalar.

VI. Routing by Agreement

Now let's see if the network can identify a boat given the input image of a boat. As discussed earlier, we run a couple of convolutional layers, we reshape the output to get vectors, and we squash them. This gives us the output of the primary capsules. We've got the first layer already. The next step is where most of the magic and complexity of capsule networks takes place. Every capsule in the first layer tries to predict the output of every capsule in the next layer. For example, let's consider the capsule that detected the rectangle (fig.12). We'll call it the rectangle-capsule.

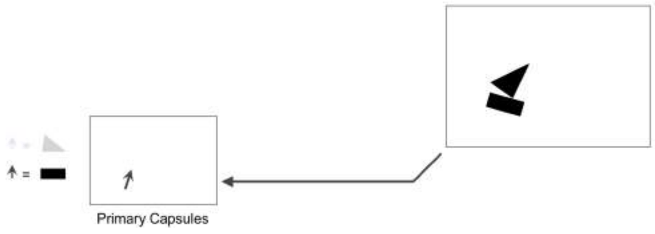


Fig.12: Rectangle-capsule [9]

Let's suppose that there are just two capsules in the next layer, the house-capsule and the boat-capsule (Fig.13). Since the rectangle-capsule detected a rectangle rotated by 16°, it predicts that the house-capsule will detect a house rotated by 16° and the boat-capsule will detect a boat rotated by 16° as well. That's what would be consistent with the orientation of the rectangle.

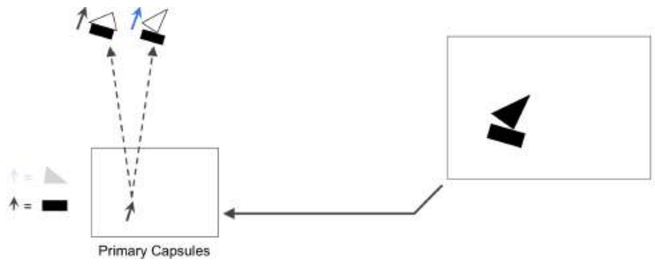


Fig.13: Prediction for the next layer (rectangle) [9]

To make this prediction, what the rectangle-capsule does is it simply computes the dot product of a transformation matrix W_{ij} with its own activation vector u_i according to equation 2. During training, the network will gradually learn a transformation matrix for each pair of capsules in the first and second layer. In other words, it will learn all the part-whole relationships, for example the angle between the wall and the roof of a house, and so on.

Now, the triangle capsule predicts that the house-capsule will detect an upside-down house, and that the boat-capsule will detect a boat rotated by 16° (Fig. 14). These are the positions that would be consistent with the rotation angle of the triangle.

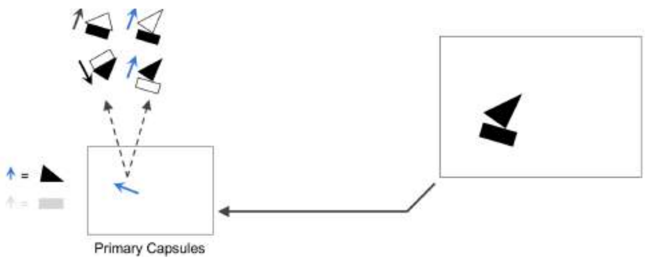


Fig. 14: Prediction for the next layer (triangle) [9]

We can see, the rectangle-capsule and the triangle-capsule strongly agree on what the boat-capsule will output (Fig.15). In other words, they agree that a boat positioned in this way would explain their own positions and rotations. And they totally disagree on what the house-capsule will output. Therefore, it makes sense to assume that the rectangle and triangle are part of a boat, not a house. Now that we know that the rectangle and triangle are part of a boat, the outputs of the rectangle capsule and the triangle capsule really concern only the boat capsule, there's no need to send these outputs to any other capsule, this would just add noise. They should be sent only to the boat capsule. This is called routing by agreement.

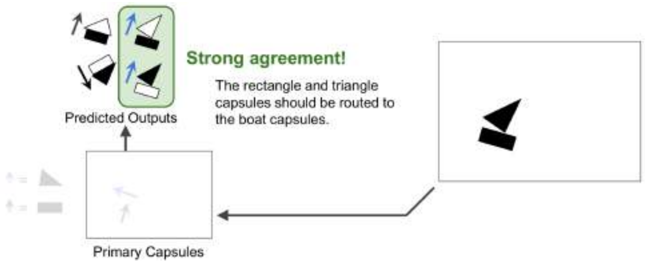

Strong agreement!
The rectangle and triangle capsules should be routed to the boat capsules.
Predicted Outputs
Primary Capsules

Fig.15: Agreement and disagreement between capsules [9]

There are several benefits of this technique: first, since capsule outputs are only routed to the appropriate capsule in the next layer, these capsules will get a cleaner input signal and will more accurately determine the pose of the object. Second, by looking at the paths of the activations, you can easily navigate the hierarchy of parts, and know exactly which part belongs to which object (like, the rectangle belongs to the boat, or the triangle belongs to the boat, and so on). Lastly, routing by agreement helps parse crowded scenes with overlapping objects.
Let's look at how routing by agreement is implemented in Capsule Networks. Here, in Fig. 16, we have represented the various poses of the boat, as predicted by the lower-level capsules. For example, one of these circles may represent what the rectangle-capsule thinks about the most likely pose of the boat, and another circle may represent what the triangle-capsule thinks, and if we suppose that there are many other low-level capsules, then we might get a cloud of prediction vectors, for the boat capsule, like this. In this example, there
are two pose parameters: one represents the rotation angle, and the other represents the size of the boat. As we mentioned earlier, pose parameters may capture many different kinds of visual features, like skew, thickness, and so on. Or precise location. So, the first thing we do, is we compute the mean of all these predictions.

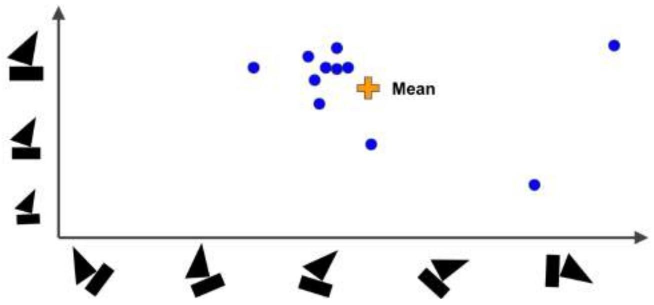


Fig.16: Clusters of Agreement

The next step is to measure the distance between each predicted vector and the mean vector. Basically, we want to measure how much each predicted vector agrees with the mean predicted vector. Using this agreement measure, we can update the weight of every predicted vector accordingly.

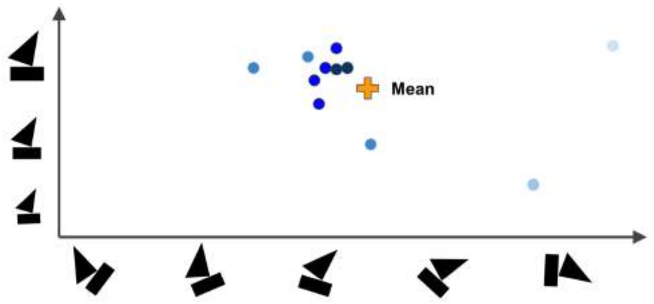


Fig.17: Clusters of Agreement

In Fig.17, we note that the predicted vectors that are far from the mean now have a very small weight and are represented in faded color, and the ones closest to the mean have a much stronger weight. We've represented them in black.
Now we can just compute the mean once again, and we can notice that it moves slightly towards the cluster, towards the center of the cluster (Fig.18). So next, we can once again update the weights. And now most of the vectors within the cluster have turned black. And again, we can update the mean.

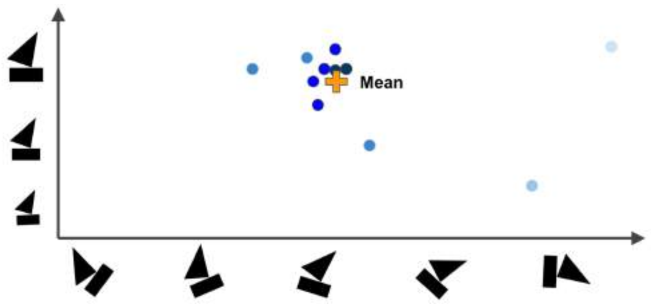


Fig.18: Updated weights

We can repeat this process a few times. In practice 3 to 5 iterations are generally sufficient. This is how we find clusters of agreement.

VII. Handling crowded scenes

Routing by agreement is really great to handle crowded scenes, such as the one represented in Fig.19. One way to interpret this image, as can see a house upside down in the middle. However, if this was the case, then there would be no explanation for the bottom rectangle or the top triangle, no reason for them to be where they are.

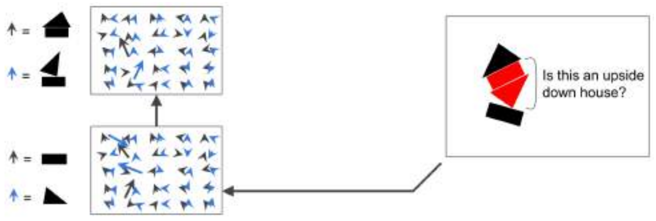


Fig.19: Handling crowded scenes [9]

The best way to interpret the image is that there is a house at the top and a boat at the bottom (Fig.20). And routing by agreement will tend to choose this solution, since it makes all the capsules perfectly happy, each of them making perfect predictions for the capsules in the next layer.

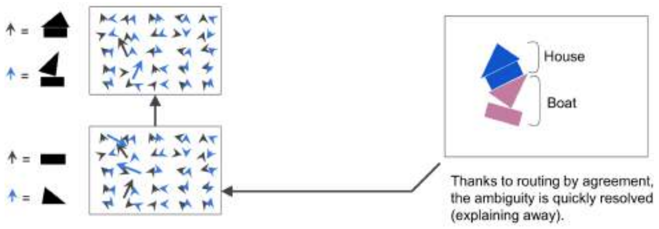

House
Boat
Thanks to routing by agreement, the ambiguity is quickly resolved (explaining away).

Fig.20: Handling crowded scenes [9]

VIII. Applications

A. Image Classification

We can create a nice image classifier with capsule networks just by having one capsule per class in the top layer. All we need to add is a layer that computes the length of the top-layer activation vectors, and this gives us the estimated class probabilities. We could then just train the network by minimizing the cross-entropy loss, as in a regular classification neural network. However, in the paper they use a margin loss (Fig.21) that makes it possible to detect multiple classes in the image. So, without going into too much details, this margin loss is such that if an object of class k is present in the image, then the corresponding top-level capsule should output a vector whose length is at least 0.9. It should be long. Conversely, if an object of class k is not present in the image, then the capsule should output a short vector, one whose length is shorter than 0.1. So, the total loss is the sum of losses for all classes.

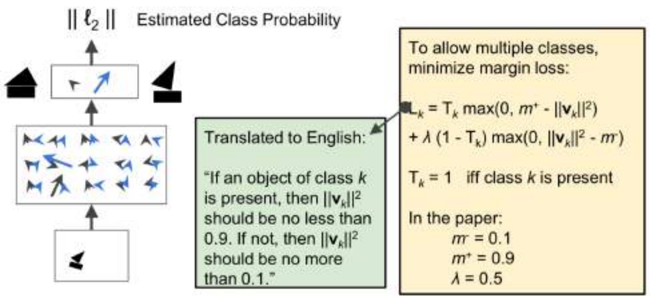

$\|\ell_2\|$ Estimated Class Probability
Translated to English:
"If an object of class k is present, then $\|\mathbf{v}_k\|^2$ should be no less than 0.9. If not, then $\|\mathbf{v}_k\|^2$ should be no more than 0.1."
To allow multiple classes, minimize margin loss:
$L_k = T_k \max(0, m^+ - \|\mathbf{v}_k\|^2) + \lambda (1 - T_k) \max(0, \|\mathbf{v}_k\|^2 - m^-)$
$T_k = 1$ iff class k is present
In the paper:
$m^- = 0.1$
$m^+ = 0.9$
$\lambda = 0.5$

Fig.21: Image classification technique of capsule

B. Image Reconstruction

In the paper, the authors also added a decoder network on top of the capsule network (Fig.22). It's just 3 fully connected layers with a sigmoid activation function in the output layer. It learns to reconstruct the input image by minimizing the squared difference between the reconstructed image and the input image. The full loss is the margin loss we discussed earlier, plus the reconstruction loss (scaled down considerably so as to ensure that the margin loss dominates training). The benefit of applying this reconstruction loss is that it forces the network to preserve all the information required to reconstruct the image, up to the top layer of the capsule network, its output layer. It reduces the risk of overfitting and helps generalize to new examples.

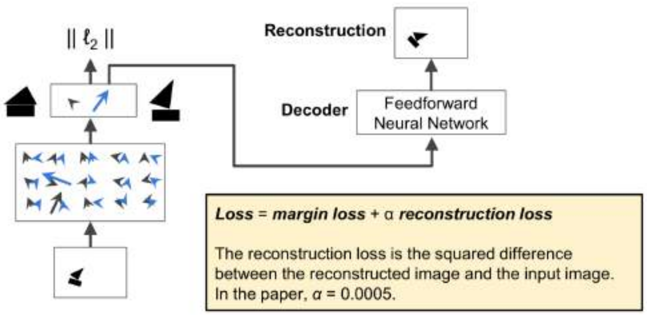

$\|\ell_2\|$
Reconstruction
Decoder
Feedforward Neural Network
Loss = margin loss + α reconstruction loss
The reconstruction loss is the squared difference between the reconstructed image and the input image.
In the paper, α = 0.0005.

Fig.22: Image reconstruction [9]

So, in a nutshell, Fig.23 shows a full capsule network for MNIST. We can see the first two regular convolutional layers, whose output is reshaped and squashed to get the activation vectors of the primary capsules. And these primary capsules are organized in a 6 by 6 grid, with 32 primary capsules in each cell of this grid, and each primary capsule outputs an 8-dimensional vector. So, this first layer of capsules is fully connected to the 10 output capsules, which output 16 dimensional vectors. The length of these vectors is used to compute the margin loss, as explained earlier.

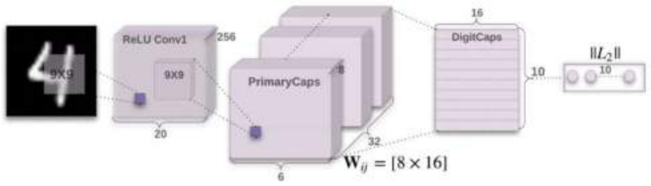


Fig.23: Capsule network for MINST

Now, the figure 2 from the main paper (Fig. 24), shows the decoder sitting on top of the capsnet. It is composed of 2 fully connected ReLU layers plus a fully connected sigmoid layer which outputs 784 numbers that correspond to the pixel intensities of the reconstructed image (which is a 28 by 28-pixel image). The squared difference between this reconstructed image and the input image gives the reconstruction loss.

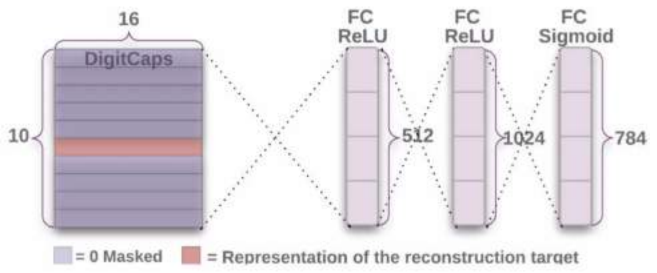


Fig.24: Capsule network for MINST

According to figure 4 from the paper, one nice thing about capsule networks is that the activation vectors are often interpretable. For example, this image (Fig. 25) shows the reconstructions that we get when we gradually modify one of the 16 dimensions of the top layer capsules' output. We can see that the first dimension seems to represent scale and thickness. The fourth dimension represents a localized skew.

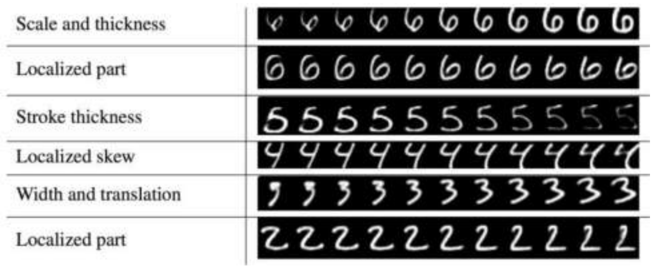


Fig.25: Interpretable Activation Vectors [12]

The fifth represents the width of the digit plus a slight translation to get the exact position.

IX. Advantages of Capsule Networks

A. *Viewpoint invariance*

The utilization of parameter vectors, or posture frameworks [7], permits Capsule Networks to perceive questions paying little heed to the viewpoint from which they are seen. Moreover, Capsule Networks are respectably robust to affine transformations of the information [6].

B. *Less Training Information*

Through unsupervised learning and the dynamic routing method, Capsule Networks meet in less epochs than CNNs. Moreover, CNNs need exponentially all the more training information to comprehend relative changes [6]

C. *Better generalization to new perspectives*

CNNs retain, that an object can be seen from various perspectives. This requires the system to "see" every single diverse change conceivable. Capsule Networks anyway generalize better to new perspectives, since parameter data of a case can catch these perspectives as insignificant direct changes [7]. Along these lines CapsNets are not as inclined to misclassification of concealed information.

D. *Few Parameters*

The associations between layers require less parameters, since just neuron gatherings are completely associated, not simply the neurons. The CNN prepared for MultiMNIST comprised of 24.56M parameters, while the CapsNet just required 11.36M parameters [6]. This is near half the same number of as in the past. Network cases with EM-routing required even less [7]. This likewise implies the model can sum up better.

E. *Validatable*

An issue for industry use of CNNs is their discovery conduct. It is neither unsurprising how a CNN will perform on new information, nor can its exhibition be appropriately broke down and comprehended. Since Capsule Networks expand upon the idea of backwards illustrations, the system's thinking can be clarified extensively superior to CNNs. Shahroudnejad et. al [18] proposed a logic strategy building naturally upon containers and their structure. This proposes, Capsule Networks are better than CNNs in validatability.

F. *Defense against white-box adversarial attacks*

Normal assaults on CNNs utilize the Fast Gradient Sign Method. It assesses the inclination of every pixel against the loss of the system. The pixels are then changed insignificantly to boost the loss without twisting the first picture. This strategy can drop the precision of CNNs to beneath 20%. Case Networks anyway keep up a precision over 70%

X. Pitfalls of Capsule Networks

A. *Scalability to complex data*

Hinton et. al [6] assessed the system tentatively on the CIFAR10 dataset, failing to perform as acceptable as current CNNs. The outcomes were practically identical to the first CNNs handling the test. Lattice cases and other examined approaches in area 3.9 attempt to handle this issue however are still a long way from performing on the ImageNet challenge.

B. *Unoptimized implementation*

The first, however not sole, execution of the Capsule Network can be found at [15]. It appears, that Capsule Networks present various difficulties for current profound learning structures. Neural Networks are frequently spoken to as a graph. Consequently, the quantity of directing emphases must be characterized exactly [6], considering the for loop to be unfolded beforehand and bound **r** times together in the diagram. Another issue for preparing neural network is that the directing calculation is dynamic and difficult to parallelize, counteracting GPUs from utilizing their full computation power. In any case almost certainly, these profound learning structures will adjust with time.

C. *Capsules require to model everything*

As depicted in [6], capsules share this issue with generative models. The system attempts to represent everything in the image. This likewise implies it performs better, in the event that it can show the messiness like foundation clamor, rather than having an extra "not-classifiable" classification. LaLonde et. al [16] attempt to comprehend this issue by reproducing not the entire image, yet just the division. This expels the requirement for the system to display the foundation and enables it to concentrate just on the dynamic class. For testing challenge of dividing therapeutic images, this methodology shows promising outcomes.

D. Loss Function

Since the network produces vector or matrix output, existing loss function can't be basically reused. In any case, they can frequently be adjusted and some of the time leverage the extra information, as can be found in the recreation loss. All things considered, utilizing the CapsNet on another dataset will frequently require another loss function as well.

E. Crowding

Human Vision experiences the "crowding" issue [17]. We can't recognize objects, when they are near one another. This can likewise be seen in Capsule Networks [6], since this idea was utilized to display the case in any case [6]. Capsules depend on the thought, that in every area in the picture is all things considered one example of the kind of substance that the case speaks to [6]. While this empowers cases to productively encode the portrayal of the substance [6], this could likewise introduce itself as an issue for explicit use cases.

XI. Conclusion and Future Work

It appears that Convolutional Neural Networks seek to clarify how neural networks should function to appropriately model intelligence. The proposed various levelled approach for learning in Convolutional Neural Networks, however not new in different fields of study, is an intriguing thought for neural network since current state-of-art models, for example, CNN are not instinctive in the manner they learn. If Capsules will be the way to go from now on, is hard to say, because we still need experiments on large datasets to get to know the real capacity of them. One challenge task could to be to achieve state of the art on ImageNet by using a fraction of the input images, since the goal of including geometrical relationships is to use way less data to learn. Though Hinton himself assembles a fascinating contention for his work, yet he additionally concedes that there is still a ton of work that should be done so as to improve the learning technique of neural network [19]. Overall, the new design holds numerous advantages over the regular CNN. Recent research demonstrates that, with certain modifications, Capsule Networks can perform even in complex situations [14]. Nonetheless, Capsule Networks must be grown further to outflank, or even supplant CNNs in real world situations, particularly when information isn't an issue.

Acknowledgment

Of the many who provided us with constructive comments, we are especially grateful to Prof. Dr. Haitham Amar and Mr. Mahmoud Mohamed Mahmoud Nasr Mohamed for their kind feedback and clear direction. We would like to pay our sincere gratitude to all the authors of the main paper on which we based our work.

References

[1] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017.

[2] M. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks", *Computer Vision – ECCV 2014*, pp. 818-833, 2014.

[3] "Convolutional Neural Network", *Encyclopedia of Social Network Analysis and Mining*, pp. 418-418, 2018.

[4] F. Jurišić, I. Filković and Z. Kalafatić, "Evaluating the Effects of Convolutional Neural Network Committees", *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.

[5] "Deep Learning Convolutional Neural Network", *Deep Learning Neural Networks*, pp. 41-55, 2016.

[6] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.

[7] S. Sabour, N. Frosst, and G. E. Hinton, "Matrix capsules with EM routing," in *ICLR 2011 Conference*.

[8] C. Szegedy, W. Liu, P, Sermanet, S. Reed, D, Anguelov, . Erhan, V. Vanhoucke, A. Rabinovich, "Going Deeper with Convolutions", in *CVPR*, 2015

[9] "Capsule Networks (CapsNets)" – Tutorial by Aurélien Géron, Nov 21, 2017 – video tutorial

[10] "Capsule Networks: An Improvement to Convolutional Networks" - Tutorial by Siraj Raval, Oct 31, 2017 – video tutorial

[11] "Understanding Hinton's Capsule Networks" by Max Pechyonkin, Nov 3, 2017 – video tutorial

[12] "Capsule Networks "by Charles Martin, Dec 23, 2017 – video tutorial

[13] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". *arXiv preprint arXiv:1603.04467*, 2016.

[14] P. Afshar, A. Mohammadi and K. Plataniotis, "Brain Tumor Type Classification via Capsule Networks", *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018. Available: 10.1109/icip.2018.8451379.

[15] Soskek. Capsnets tensorflow implementation. https://github.com/soskek/dynamic_routing_ between_capsules

[16] R. LaLonde and U. Bagci, "Capsules for Object Segmentation." ArXiv e-prints, Apr. 2018.

[17] D. G. Pelli, "Crowding: A cortical constraint on object recognition. Current opinion in neurobiology". 18(4):445–451, 2008.

[18] A. Shahroudnejad, P. Afshar, K. Plataniotis and A. Mohammadi, "IMPROVED EXPLAINABILITY OF CAPSULE NETWORKS: RELEVANCE PATH BY AGREEMENT", *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2018. Available: 10.1109/globalsip.2018.8646474

[19] G. E. Hinton. What is wrong with convolutional neural nets? Talk recorded on youtube, https://youtu.be/rTawFwUvnLE; last accessed on 2018/06/14.