Capsule Networks

Beyond Convolutional Neural Networks

Presentation by Matteo Tiezzi

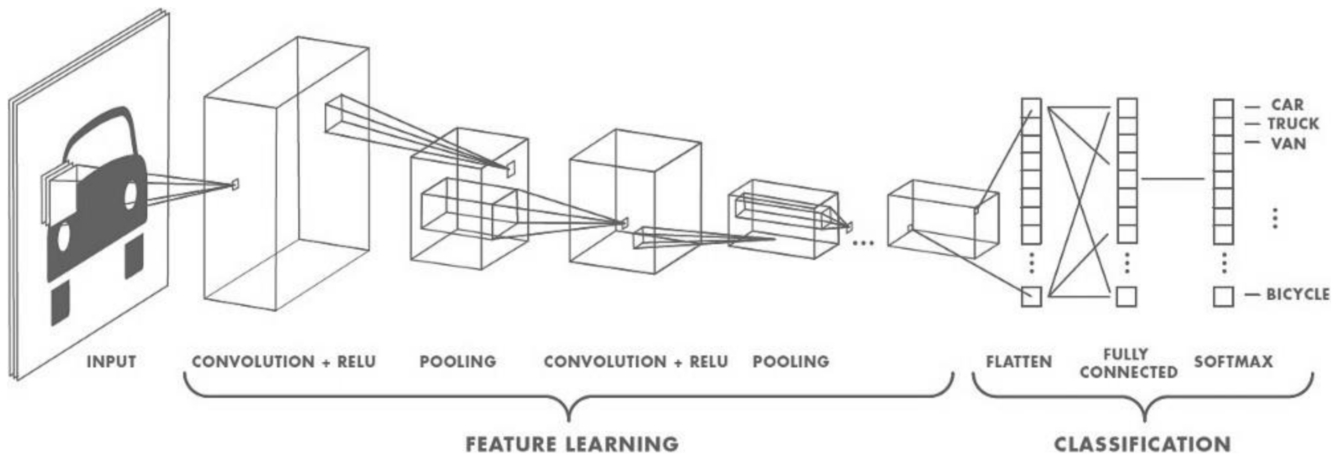
A work by Geoffrey Hinton

Presentation based on the following papers:

- **Transforming auto-encoders ,** Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. In International Conference on Artificial Neural Networks, pages 44–51. Springer, **2011**.
- **Dynamic Routing Between Capsules,** Sara Sabour, Nicholas Frosst, Geoffrey E Hinton, *Nov* **2017**
- **(Matrix capsules with EM routing,** Anonymous authors, Paper under double-blind review)

An evolving theory

Convolutional Neural Networks



Stack of layers with Convolution, Subsampling and Nonlinearity Operations

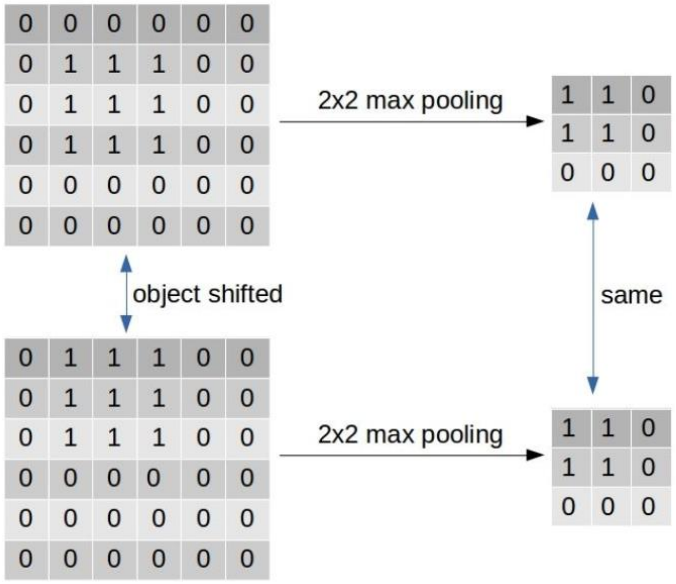
- ✓ CNN use multiple layers of learned feature detectors (Kernels running spatially all over the image)
- ✓ Features detectors are local, and each type is replicated across space
- ✓ Spatial domains get bigger in higher layers

- x Feature extractions layers are interleaved with subsampling layers that pool the outputs of nearby features detector of the same type

The motivations for pooling

- Reduces the number of inputs to the next layer of feature extractions (*reduces the size of activation maps*)
  - Allowing to have more types of feature and bigger domains, besides the computational cost reduction

- Gives a small amount of **translational invariance** at each level
  - Motivated by the fact that the final label needs to be viewpoint-invariant
  - Precise location of the most active feature is **thrown away**

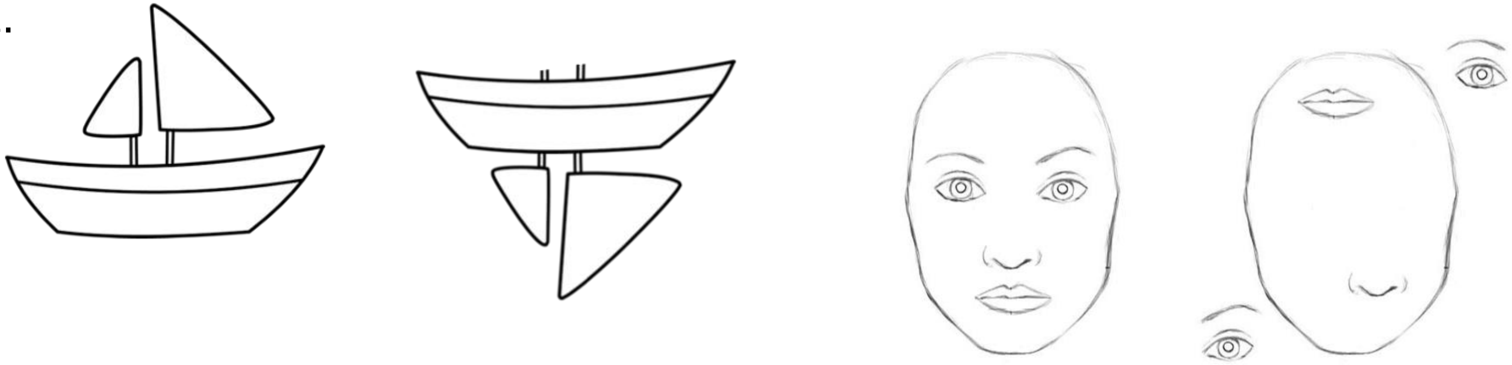


If an entity in the image is translated by a small amount, the activation map corresponding to that entity will shift equally. But, the max-pooled output of the activation map remains unaltered.

Without pooling CNNs would fit only for images or data which are very close to the training set.

Internal data representation of a convolutional neural network does not take into account important spatial hierarchies between simple and complex objects.

The following pictures may fool a ***simple*** CNN model in believing that this a good sketch of boat, human face, etc.



Sub-sampling loses the precise spatial relationships between higher-level parts such as a nose and a mouth. The precise spatial relationships are needed for identity recognition

Overlapping the sub-sampling pools mitigates this.

They cannot extrapolate their understanding of geometric relationships to radically new viewpoints

Invariance vs Equivariance

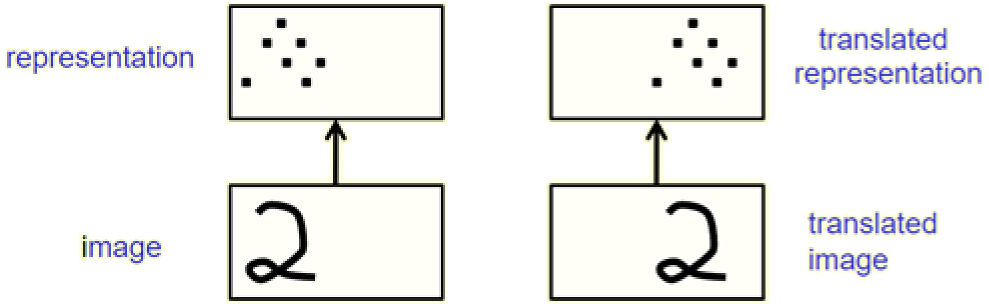
In general it is said that:

- an Operator is **invariant** with respect to a Transformation when the effect of the Transformation is not detectable in the Operator Output

- an Operator is **equivariant** with respect to a Transformation when the effect of the Transformation is detectable in the Operator Output

Sub-sampling tries to make the **neural activities** *invariant* to small changes in viewpoint.

- This is the wrong goal, motivated by the fact that the final label needs to be viewpoint-invariant.
- Its better to aim for **equivariance**: we want that changes in viewpoint lead to corresponding changes in neural activities.
- In the perceptual system, its the **weights** that code viewpoint-invariant knowledge, not the neural activities.



Without pooling, CNN give «place-coded» equivariance for discrete traslation

Extrapolating shape recognition to very different viewpoint

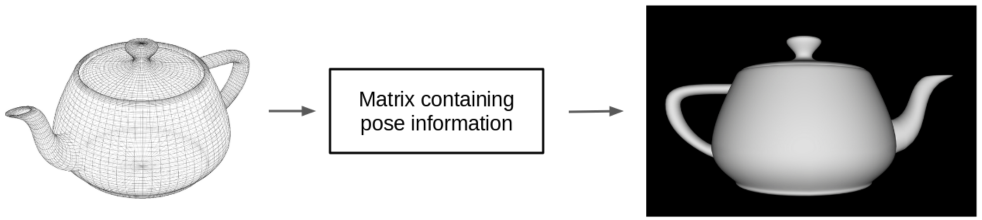
- To deal with invariance, current NNs train on different viewpoints
- This requires a lot of training data

Better approach:

- The manifold of images of the same shape is highly non-linear in the space of pixel intensities
- If we transform it to a space in which the manifold is globally linear, this allows massive extrapolation

There is a linear manifold (the one which Computer graphics uses). If we get from pixels to coordinate representation of pieces of objects, obtaining their *poses*, than everithing is linear in that.

Obtaining equivariance: Inverse computer graphics



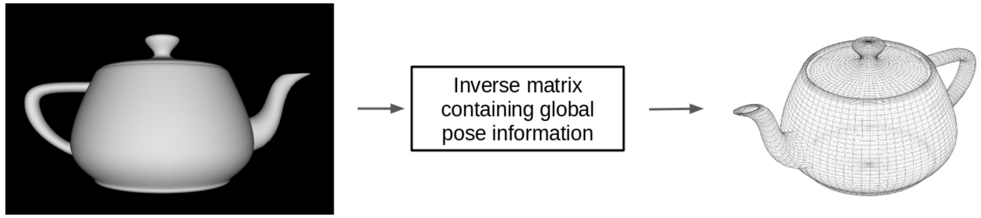
To go from a mesh object onto pixels on a screen, it takes the pose of the whole object, and multiplies it by a transformation matrix. This outputs the pose of the object's part in a lower dimension (2D), which is what we see on our screens.

Computer vision as *inverse computer graphics.*

So the higher levels of a vision system should look like the representations used in graphics.

Graphics programs use hierarchical models in which spatial structure is modeled by matrices (of weights), that represent the relationship between the object as a whole and the **pose** of the part.

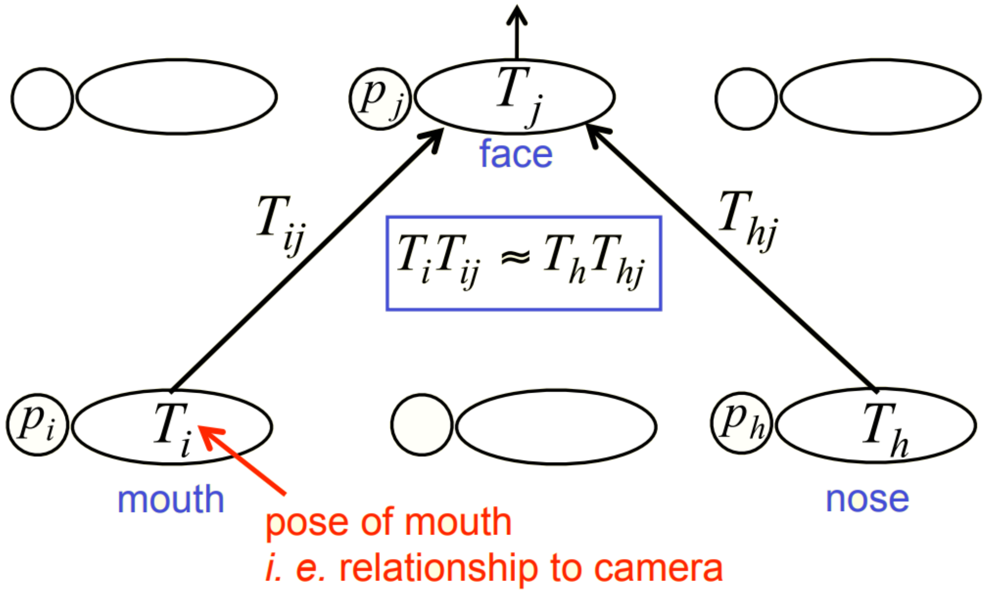
- These matrices are totally **viewpoint invariant**.
- However much the pose of the part has changed we can get back the pose of the whole using the same matrix of weights.



This gives complete independence (and translational invariance) between the viewpoints of the object in a matrix of weights, not in the neural activity (equivariance).

Advantages and way to proceed

- It becomes very easy for a model to understand that the thing that it sees is just another view of something that it has seen before.
- In this way it is possible to learn by **only using a fraction of the data that a CNN would use.**



Two layers in a hierarchy of parts

Coincidence filtering using the linear manifold

A higher level visual entity is present if several lower level visual entities can agree on their predictions for its pose.

(Shape recognition in computer vision of '80)

T_h : **pose** of the nose
p_h: probability that the nose is present
T_{hj} , ... : viewpoint invariant

Exactly what Capsules do.

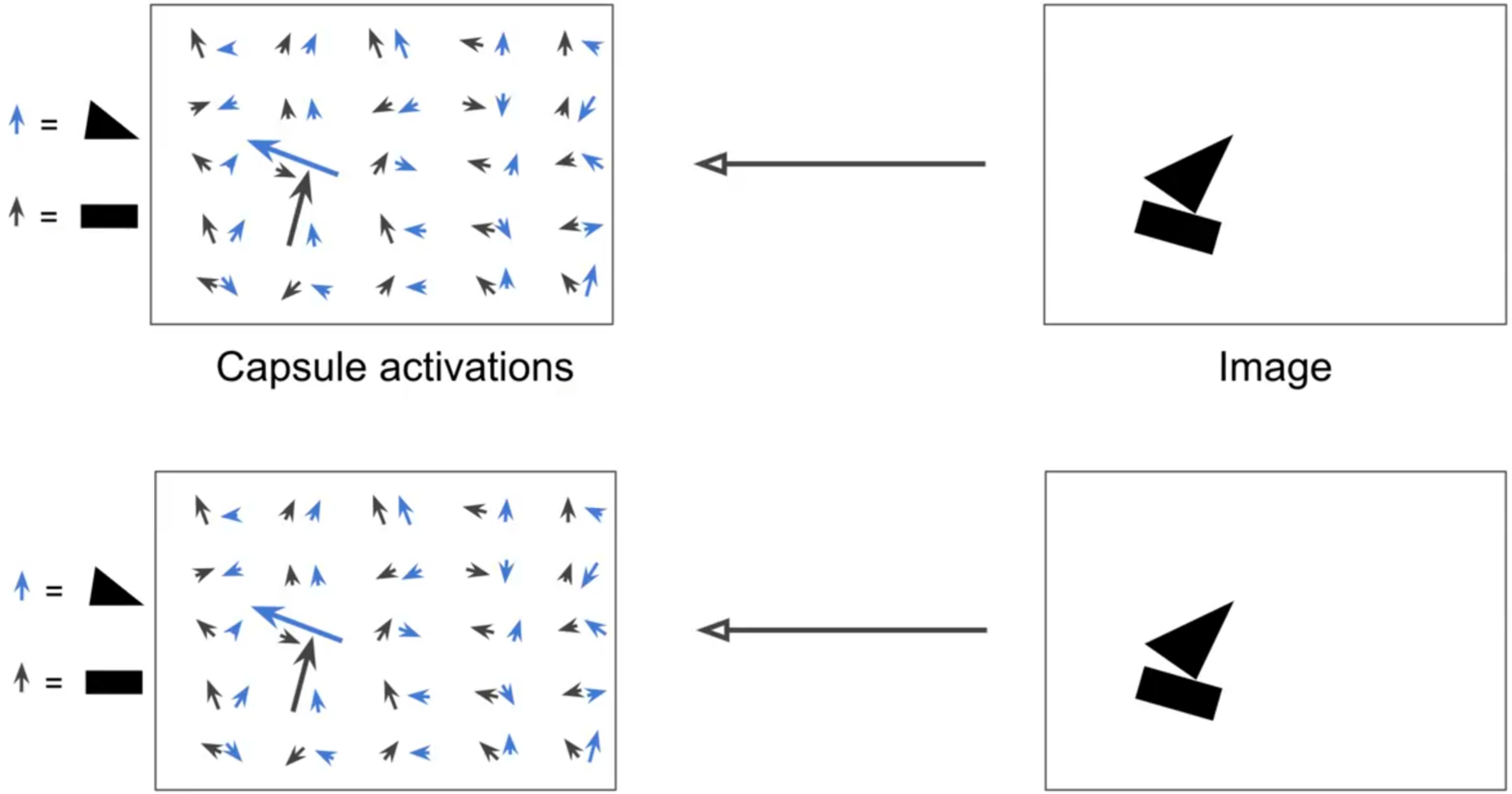
Capsule

From Transforming Autoencoders:
Instead of aiming for viewpoint invariance in the activities of "neurons" that use a single scalar output to summarize the activities of a local pool of replicated feature detectors, artificial neural networks should use local "capsules"

- Group of neurons that perform a lot of internal computation and then encapsulate the results of these computations into a **small vector** of highly informative outputs. Inspired by mini-column in brain.
- Each capsule learns to recognize an implicitly defined visual **entity** over a **limited domain** of viewing conditions and deformations
- It outputs two things (embedded in the vector):
  1. the **probability** that the entity is present within its limited domain
  2. a set of "instantiation parameters", the generalized **pose** of the object. That may include the precise position, lighting and deformation of the visual entity relative to an implicitly defined canonical version of that entity

In the last paper (*Dynamic Routing Between Capsules*), Capsules encode probability of detection of a feature **as the length** of their output vector.

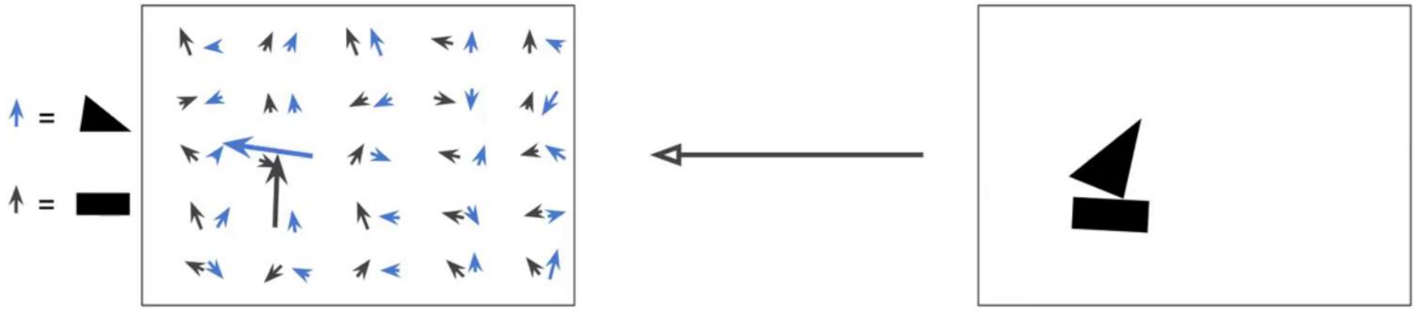
Capsule



Activation vector: **Length** = estimated probability of presence
Orientation = object's estimated pose parameters

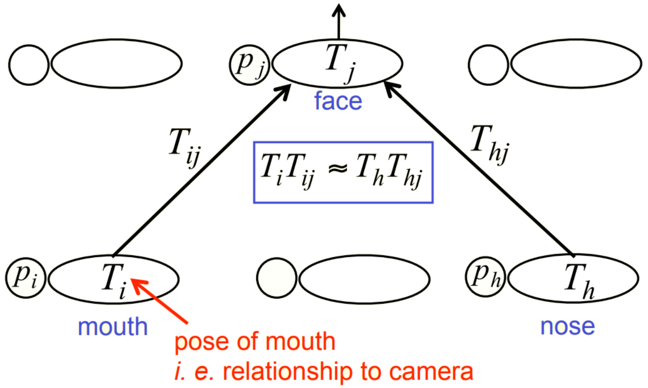
Capsule equivariance

- Capsules encode probability of detection of a feature as the length of their output vector. And the state of the detected feature is encoded as the direction in which that vector points to ("instantiation parameters").
- So when detected feature moves around the image or its state somehow changes, the probability still stays the same (length of vector does not change), but its orientation changes.
- This is what Hinton refers to as **activities equivariance**: neuronal activities will change when an object "moves over the manifold of possible appearances" in the picture. At the same time, the **probabilities of detection remain constant**, which is the form of invariance that we should aim at, and not the type offered by CNNs with max pooling.

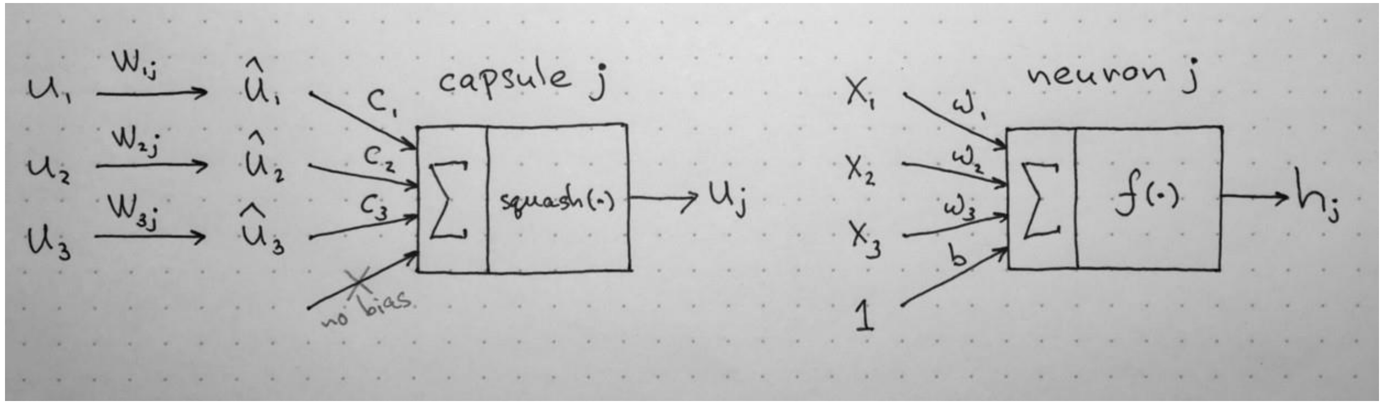


How does a capsule work?

They are organized in layers.



Let u_1, u_2, u_3 be the output **vectors** coming from capsules of the layer below. The vector is sent to all possible parents in the neural network.



Let us assume that lower level capsules detect eyes, mouth and nose respectively and out capsule detects face.

- These vectors then are multiplied by corresponding weight matrices W (**learned** during training) that encode important spatial and other relationships between lower level features (eyes, mouth and nose) and higher level feature (face). W performs an **affine transformation**
- We get the predicted position of the higher level feature, $\hat{u}_{j|i} = W_{ij} u_i$
  i.e. where the face should be according to the detected position of the eyes

Next intuition: if these 3 predictions of lower level features **point at the same position and state** of the face, then it must be a face there.

How does a capsule work?

Then we compute a weighted sum s_j with weights c_{ij} , **coupling coefficient** trained by dynamic routing (discussed next)

$$s_j = \sum_i c_{ij} \hat{u}_{j|i}$$

We apply a squashing function (a non-linear activation function) to scale the vector between 0 and unit length (its length represent a probability, as already stated). This do not change the vector direction (its **pose**).

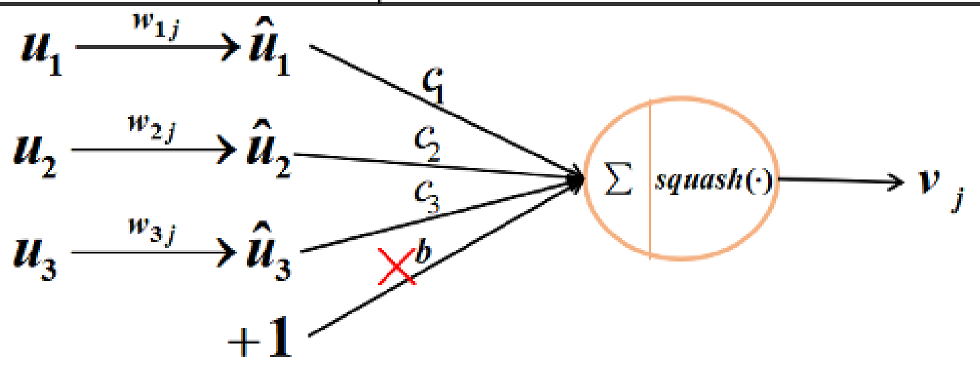
$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|}$$

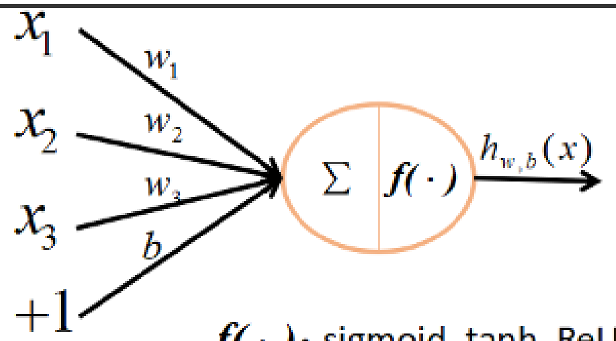
It shrinks small vectors to zero and long vectors to unit vectors. Therefore the likelihood of each capsule is bounded between zero and one.

$$\begin{aligned} v_j &\approx \|s_j\| s_j && \text{for } s_j \text{ is short} \\ v_j &\approx \frac{s_j}{\|s_j\|} && \text{for } s_j \text{ is long} \end{aligned}$$

Summary

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





$f(\cdot)$: sigmoid, tanh, ReLU, etc.

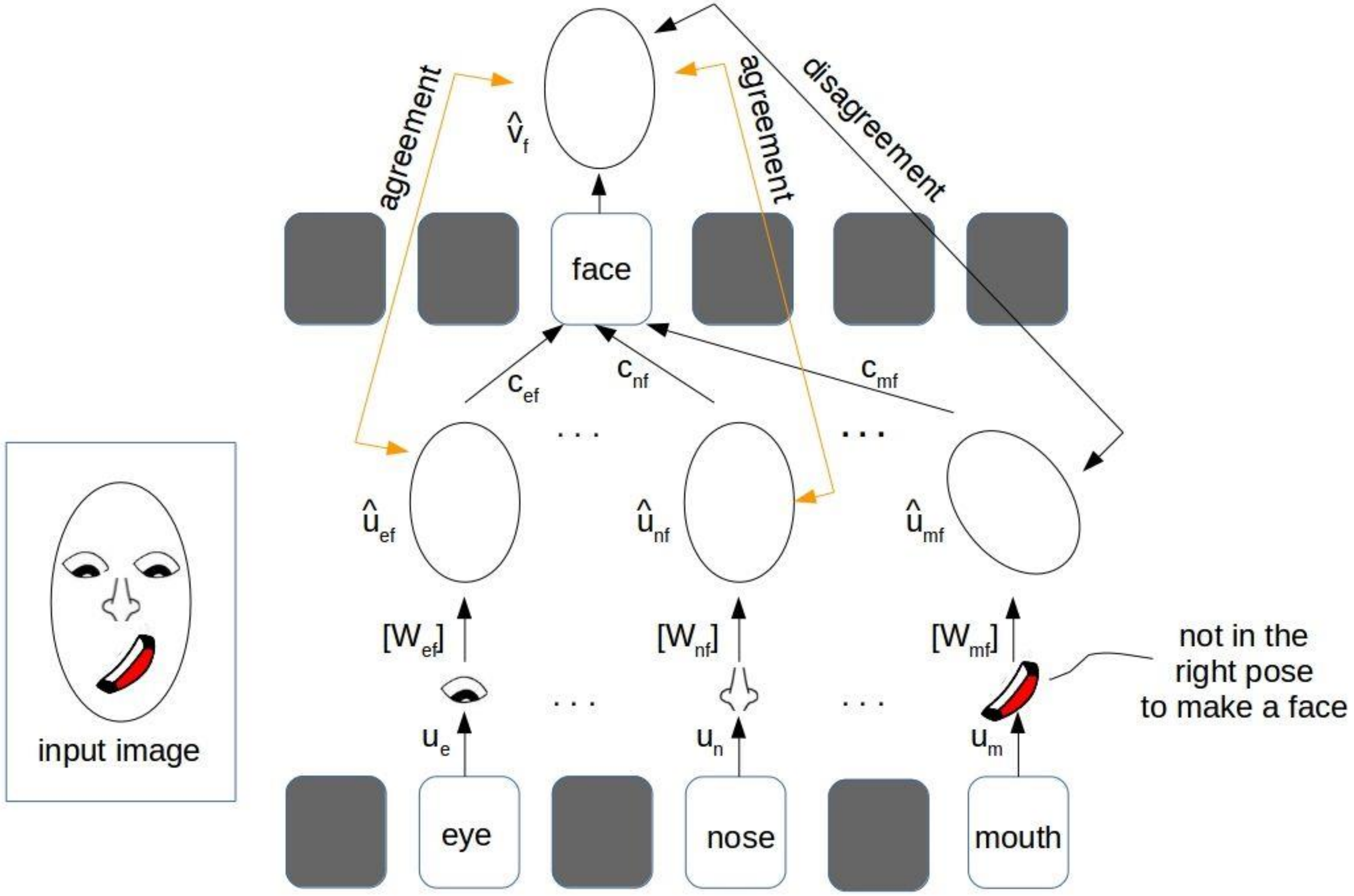
Capsule = New Version Neuron!
vector in, vector out VS. scalar in, scalar out

Routing by Agreement

- According to Hinton, when a visual stimulus is triggered, the brain has an inbuilt mechanism to *"route"* low level visual data to parts of the brain where it believes can handle it best.
- ConvNets perform routing via pooling layers, a very primitive way to do routing as it only attends to the most active neuron in the pool.

Capsule Networks is different as it tries to send the information to the capsule above it that is best at dealing with it.

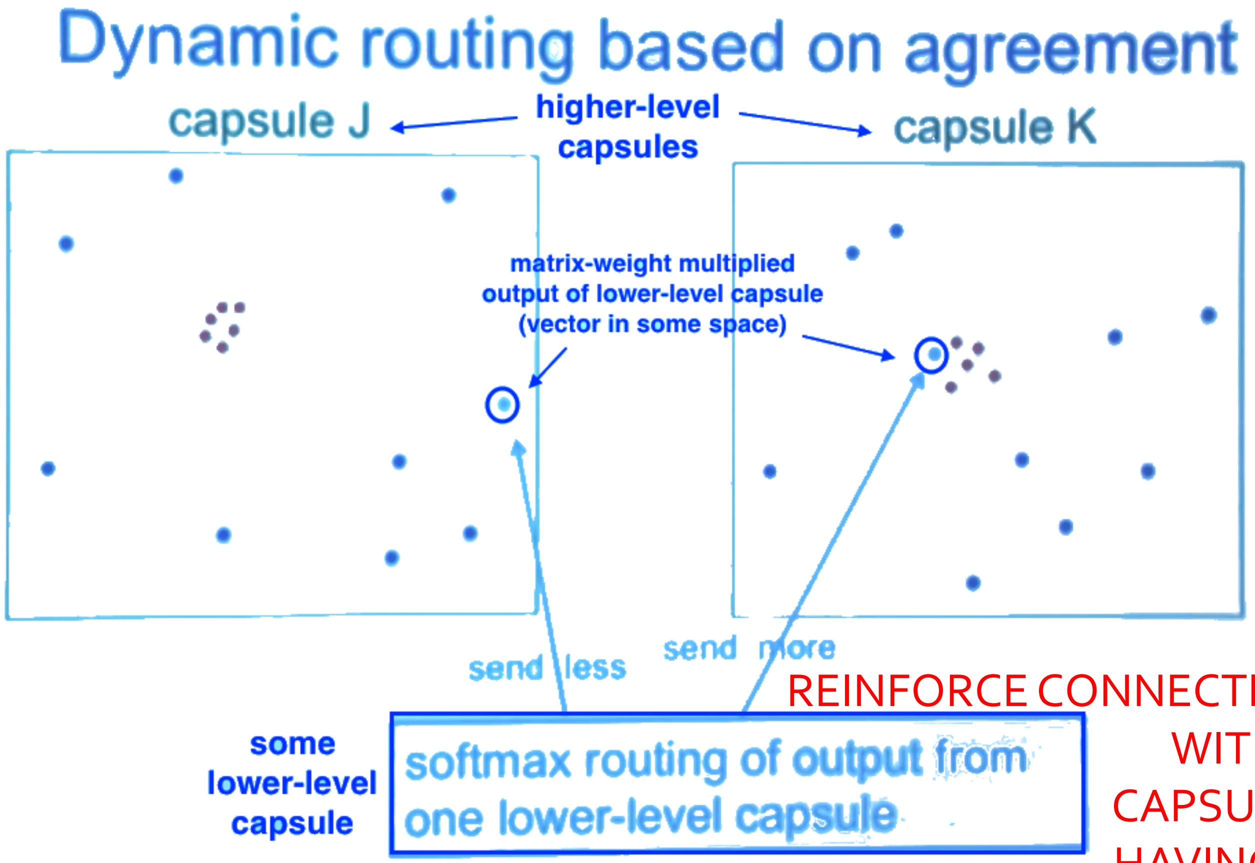
The parameters W_{ij} models a "**part-whole**" relationship between the lower and higher level entities



Iterative dynamic Routing Algorithm

High-dimensional coincidence
in multi-dim pose space

- A capsule receives multi-dim **prediction vectors** from caps in the layer below
- It looks for tight cluster of predictions
- It otputs:
  - High probability that an entity of this type exists in its domain
  - The center of gravity of the cluster, which is the generalized pose of that entity



REINFORCE CONNECTION WITH CAPSULE HAVING THE CLOSEST CENTROID

Lower level capsule will send its input to the higher level capsule that "agrees" with its input. This is the essence of the dynamic routing algorithm.

Iterative dynamic Routing Algorithm

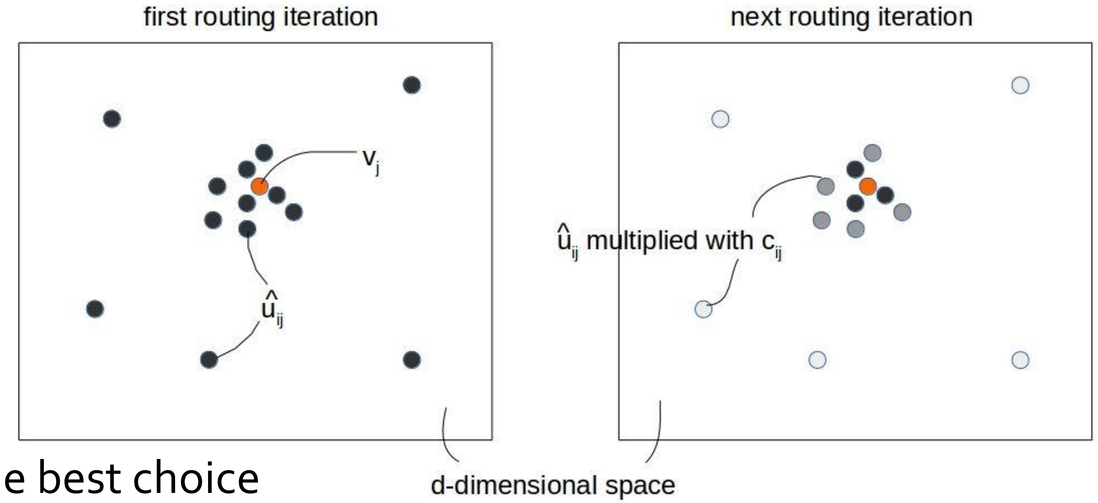
- Intuitively, prediction vector $\hat{u}_{j|i}$ is the prediction (**vote**) from the capsule **i** on the output of the capsule **j** above.
- If the activity vector v_j has close similarity with the prediction vector, we conclude that capsule **i** is highly related with the capsule **j**. (For example, the eye capsule is highly related to the face capsule.)

- Similarity measured with the "**agreement**" quantity $a_{ij}=\langle \hat{u}_{j|i}, v_j \rangle$.

- Judging by the values of a_{ij} we can then "*strengthen*" or "*weaken*" the corresponding connection strength by highering or lowering c_{ij} appropriately.

Procedure 1 Routing algorithm.

procedure ROUTING($\hat{\mathbf{u}}_{j|i}$, r, l)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \mathtt{softmax}(\mathbf{b}_i)$
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \mathtt{squash}(\mathbf{s}_j)$
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
  return $\mathbf{v}_j$



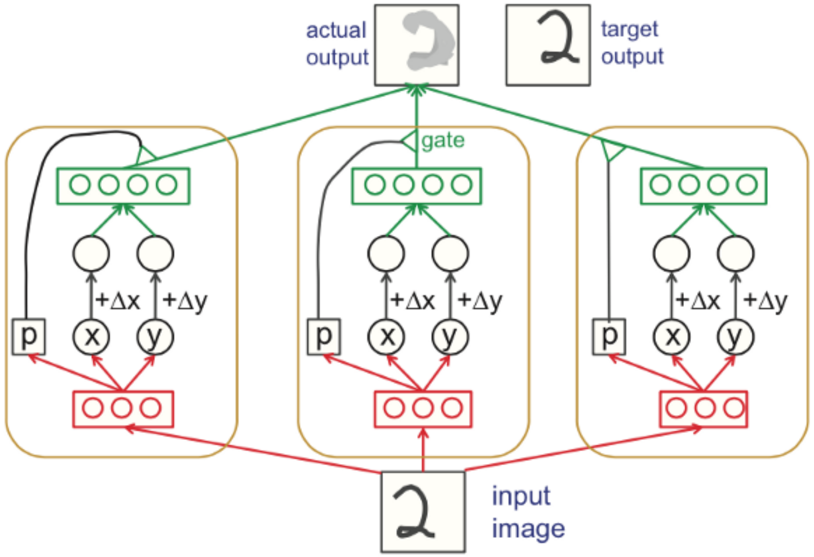
b_{ij} for each training example. 3 Routing iterations seems the best choice

By "*fading away*" the incoming connections that don't agree, we enforce the connection parameters W_{ij} to learn more prominent "**part-whole**" relationships, behaving like a **parse-tree.** Each active capsule will choose a caps in the layer above to be its parent in the tree.

Learning the first level of capsules: Transforming Autoencoders (2011)

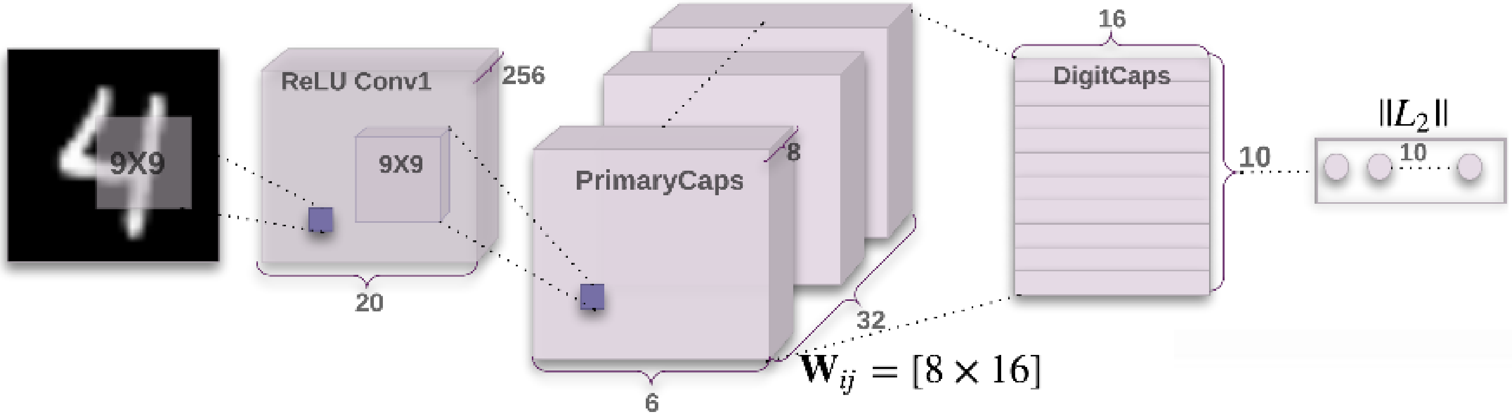
- The "capsules" that implement the lowest-level parts in the hierarchy need to extract **explicit pose** parameters **from pixel intensities**
- These capsules are quite easy to learn from pairs of transformed images if the neural net has direct, nonvisual access to the transformations
- In humans, for example, a saccade causes a pure translation of the retinal image and the cortex has non-visual access to information about eye-movements.
- We explain the idea using simple 2-D images and capsules whose only **pose outputs** are an x and a y position

- Once it has been learned, the net takes as inputs an image and desired shifts, Δx and Δy, and it outputs the shifted image
- Each capsule has its own logistic "recognition units" that act as a hidden layer for computing three numbers, x, y, and p
    - "Generation units", the capsule's contribution to the transformed image. Inputs are x + Δx and y + Δy, and the contributions from the capsule's generation units to the output image are multiplied by p, so inactive capsules have no effect

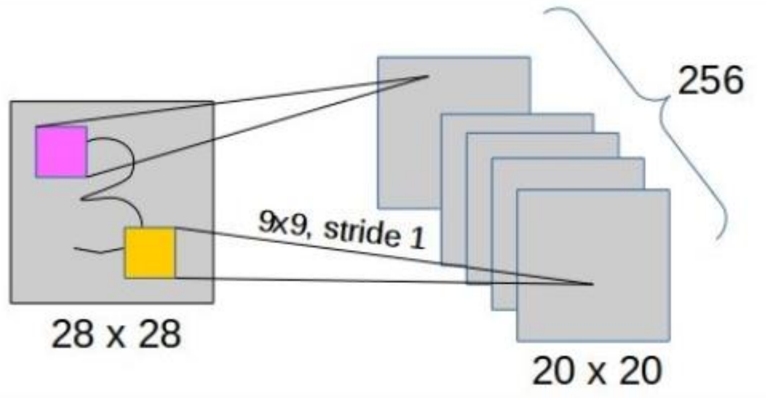


CapsNet (2017) for MNIST

- CNNs use translated replicas of learned feature detectors. This allows them to translate knowledge about good weight values acquired at one position in an image to other positions
- CapsNet replace the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement, we would still like to replicate learned knowledge across space
- All but the last layer composed by convolutional capsules

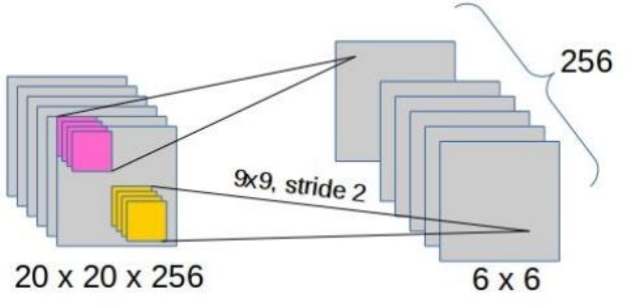


1. **First convolutional layer**: *This is an usual convolutional layer,* image 28 x 28 convolved by 256 kernels of shape 9 x 9. Output of this layer is 256 feature maps/activation maps of shape 20 x 20



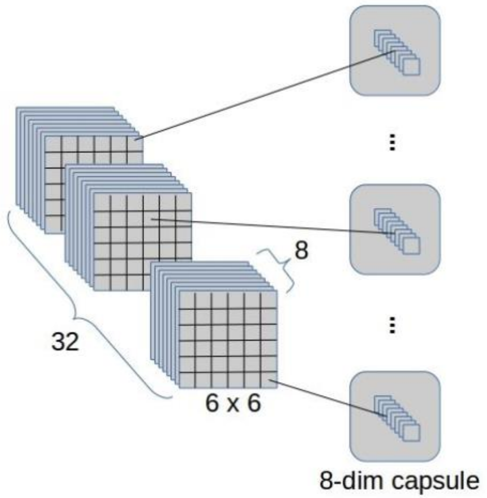
CapsNet (2017) for MNIST

2. **Second convolutional layer** or the **PrimaryCaps layer**:
   1. *another convolutional layer* which produces 256 activation maps of 6 x 6

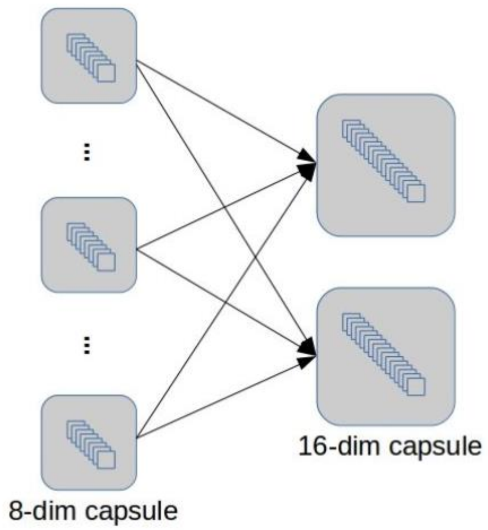


   2. Output of the second convolutional layer (6 x 6 x 256) interpreted as a set of 32 "*capsule activation maps*" with capsule dimension 8.

A total of 6*6*32 = 1152 capsules (each of dimension 8)



CapsNet (2017) for MNIST



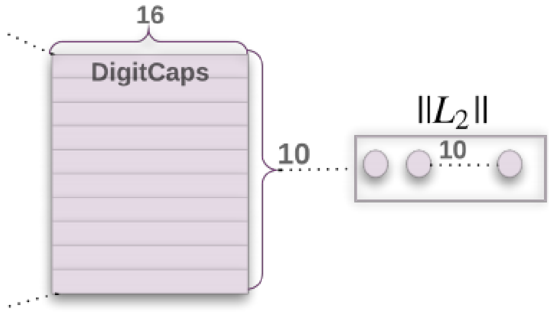
3. **Capsule-to-capsule layer** or **DigitCaps layers:**

- The 1152 (lower level) capsules are connected to 10 (higher levels)
- capsules (a total of 1152*10 = 11520 weight matrices Wij)
- The 10 higher level capsules (of dimension 16) represent the 10 final

"*digit/class entities*"

- This layer also has the "*dynamic routing*" in it.

The loss function



Capsules use a separate margin loss L_c for each category c of digit capsules:

$$L_c = T_c \max(0, m^+ - ||v_c||)^2 + \lambda(1 - T_c)\max(0, ||v_c|| - m^-)^2$$

T_c=1 if an object of class c is present. m_+=0.9 and m_-=0.1., λ=0.5 down-weighting for absent digit classes

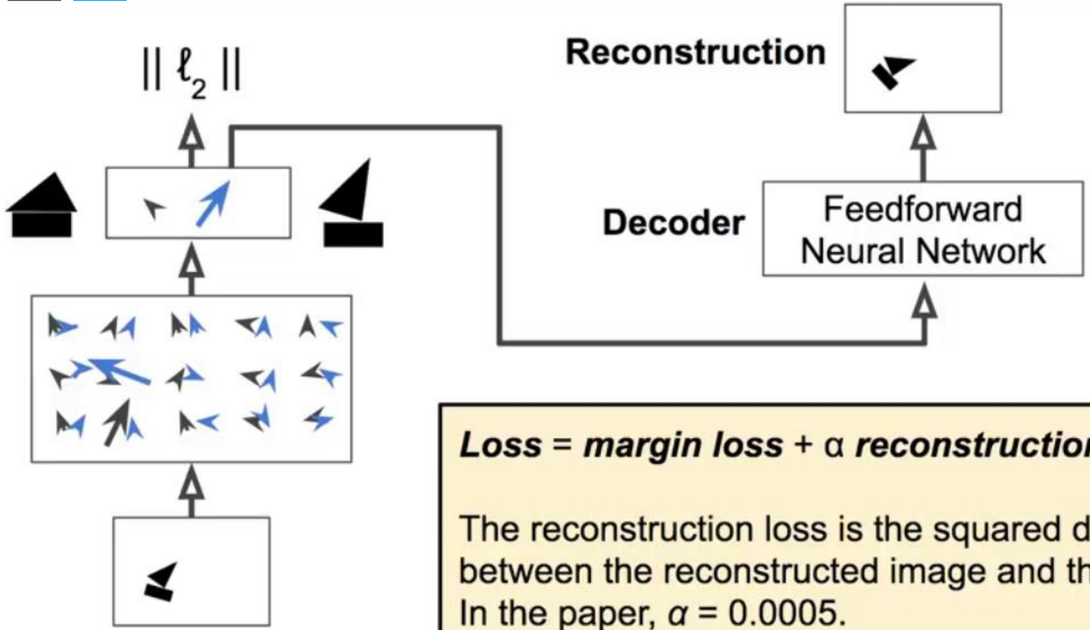
Translated to english : if an object of class c is present, then $||v_c||$ should be no less than 0,9. If not then $||v_c||$ should be no more than 0,1

Total loss is the sum of losses of all digit capsules

Reconstruction as a regularization method

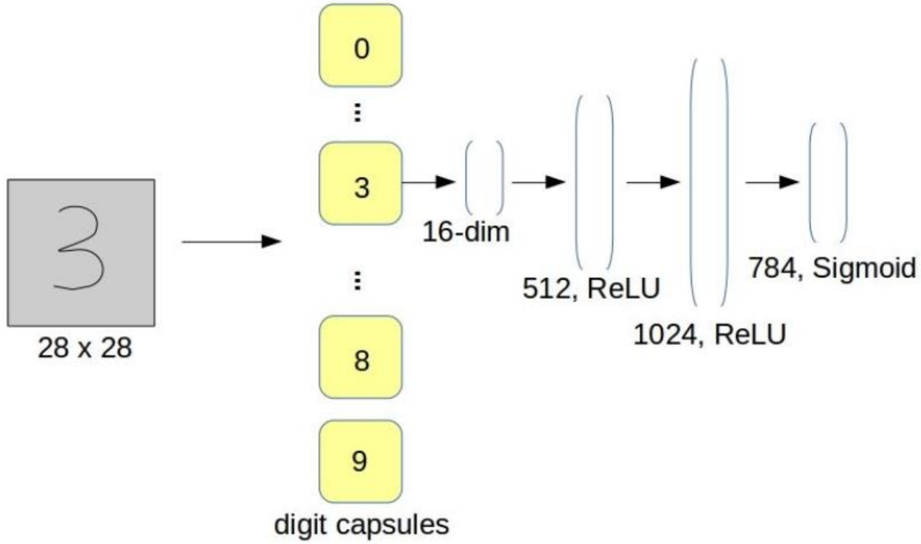
- Capsule Networks use a reconstruction loss as a regularization method to **encourage the digit capsules to encode the instantiation parameters** of the input digit
- In order to reconstruct the input from a lower dimensional space, the Encoder and Decoder needs to **learn a good matrix representation to relate the relationship between the latent space and the input**

During training, we mask out all but the activity vector of the correct digit capsule. Then we use this activity vector to reconstruct the input image.



Loss = ***margin loss*** + α ***reconstruction loss***

The reconstruction loss is the squared difference between the reconstructed image and the input image. In the paper, $\alpha = 0.0005$.



Reconstruction as a regularization method

To summarize:

- using the reconstruction loss as a regularizer, the Capsule Network is able to learn a global linear manifold between a whole object and the pose of the object and its parts as a matrix of weights via **unsupervised learning**.
- the *translation invariance* is encapsulated in the matrix of weights, and not during neural activity, making the neural network *translation equivariance*.

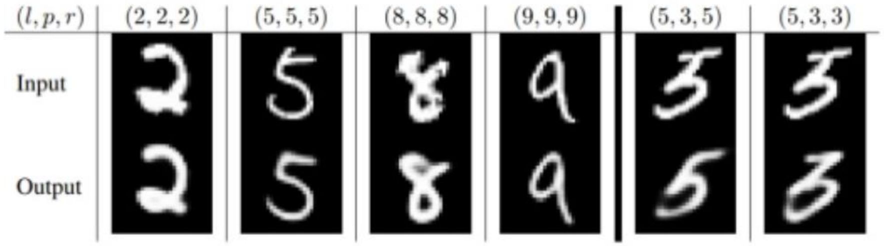
Experimental Results

The model achieves state-of-the-art performance on MNIST and is considerably better than a convolutional net at recognizing highly overlapping digits

MNIST Dataset

- **Prediction and Reconstruction Example**

(l, p, r) = label, prediction, reconstruction target



- **Classification Accuracy**

Method	Routing	Reconstruction	MNIST (%)	MultiMNIST (%)
Baseline	-	-	0.39	8.1
CapsNet	1	no	$0.34_{\pm 0.032}$	-
CapsNet	1	yes	$0.29_{\pm 0.011}$	7.5
CapsNet	3	no	$0.35_{\pm 0.036}$	-
CapsNet	3	yes	$\mathbf{0.25}_{\pm 0.005}$	**5.2**

MATRIX CAPSULES WITH EM ROUTING

- On the smallNORB benchmark, capsules reduce the number of test errors by 45% compared to the state-of-the-art. Capsules also show far more resistance to white box adversarial attack than our baseline convolutional neural network.

The smallNORB dataset (LeCun et al. (2004)) has gray-level stereo images of 5 classes of toy: airplanes, cars, trucks, humans and animals. There are 10 physical instances of each class which are painted matte green. 5 physical instances of a class are selected for the training data and the other 5 for the test data. Every individual toy is pictured at 18 different azimuths (0-340), 9 elevations and 6 lighting conditions, so the training and test sets each contain 24,300 stereo pairs of 96x96 images. We selected smallNORB as a benchmark for developing our capsules system because it is carefully designed to be a pure shape recognition task that is not confounded by context and color, but it is much closer to natural images than MNIST.

Table 1: The effect of varying different components of our capsules architecture on smallNORB.

Routing iterations	Pose structure	Loss	Coordinate Addition	Test error rate
1	Matrix	Spread	Yes	9.7%
2	Matrix	Spread	Yes	2.2%
3	Matrix	Spread	Yes	**1.8%**
5	Matrix	Spread	Yes	3.9%
3	Vector	Spread	Yes	2.9%
3	Matrix	Spread	No	2.6%
3	Vector	Spread	No	3.2%
3	Matrix	Margin[1]	Yes	3.2%
3	Matrix	CrossEnt	Yes	5.8%
Baseline CNN with 4.2M parameters				5.2%
CNN of Cireşan et al. (2011) with extra input images & deformations				2.56%
Our Best model (third row), with multiple crops during testing				**1.4%**

Pros

- Reaches high accuracy on MNIST, and promising on CIFAR10
- Requires less training data
- Position and pose information are preserved (equivariance)
- This is promising for image segmentation and object detection
- Routing by agreement is great for overlapping objects (explaining away)
- Capsule activations nicely map the hierarchy of parts
- Offers robustness to affine transformations
- Activation vectors are easier to interpret (rotation, thickness, skew…)

Cons

- Not state of the art on CIFAR10 (but it's a good start)
- Not tested yet on larger images (e.g., ImageNet): will it work well?
- Slow training, due to the inner loop (in the routing by agreement algorithm)
- A CapsNet cannot see two very close identical objects
    - This is called "crowding", and it has been observed as well in human vision

References

- **Transforming auto-encoders ,** Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. In International Conference on Artificial Neural Networks, pages 44–51. Springer, **2011**.
- **Dynamic Routing Between Capsules,** Sara Sabour, Nicholas Frosst, Geoffrey E Hinton, *Nov* **2017**
- **(Matrix capsules with EM routing,** Anonymous authors, Paper under double-blind review)
- https://www.slideshare.net/kyuhwanjung/vuno-dl-seminarcapsnetskyuhwanjung20171109
- https://kndrck.co/posts/capsule_networks_explained/
- https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66
- Aurélien Géron, Capsule Networks (CapsNets) – Tutorial