Capsule Neural Networks

Klim Markelov

Author

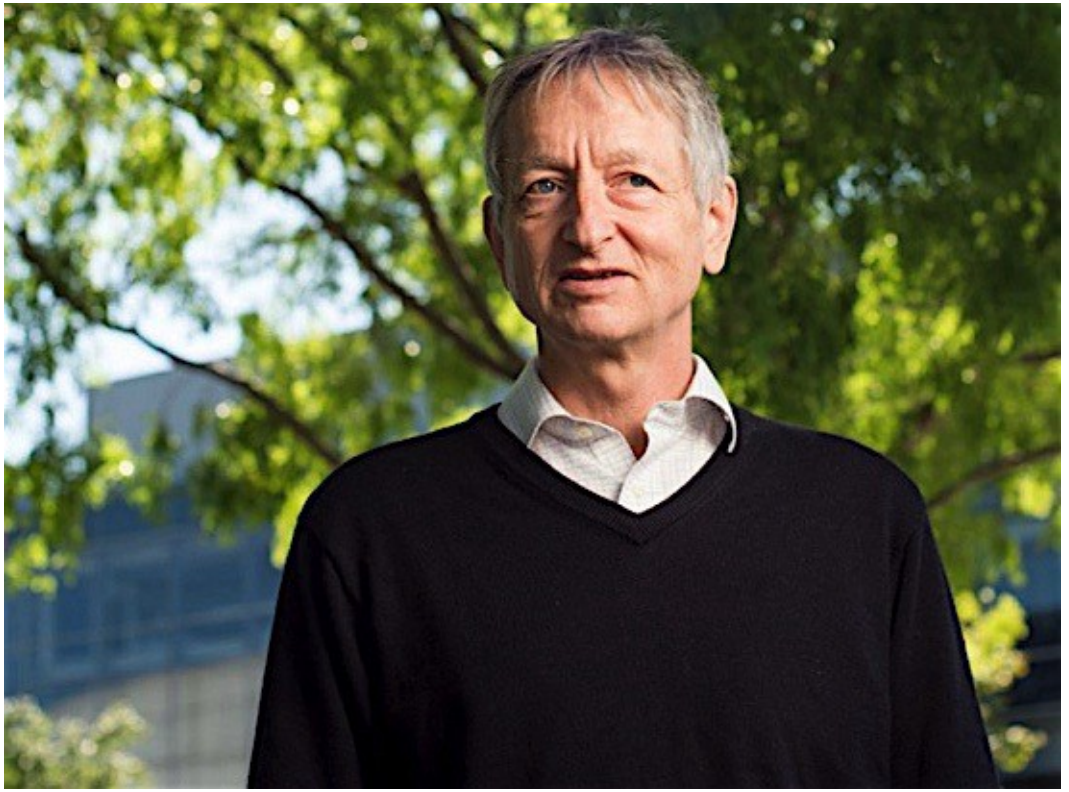
Geoffrey Hinton

PhD in artificial intelligence

One of the developers of back-propagation algorithm

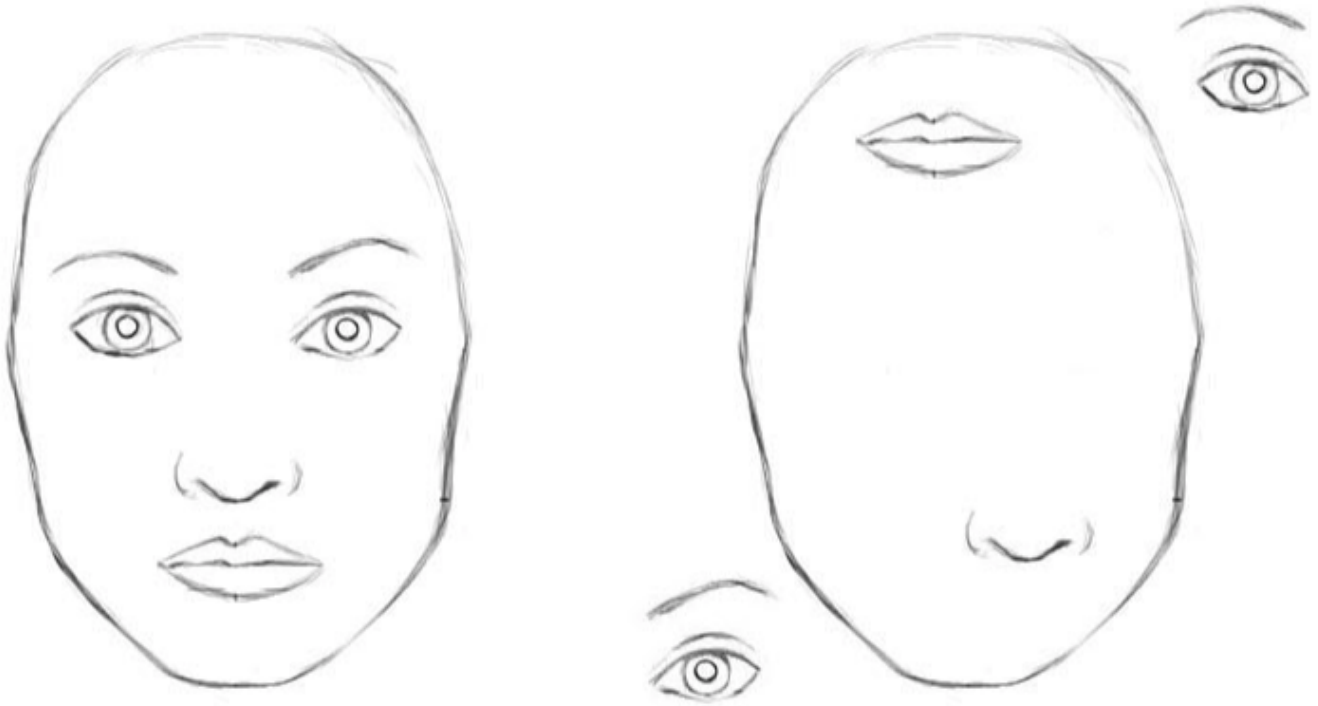
Leading figure in the deep learning community

“Godfather of Deep Learning”



Geoffrey Hinton is looking into the distance and thinking about Capsule Networks

Whats wrong with CNN



To a CNN, both pictures are similar, since they both contain similar elements.

Hinton about MaxPooling

“The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster.”

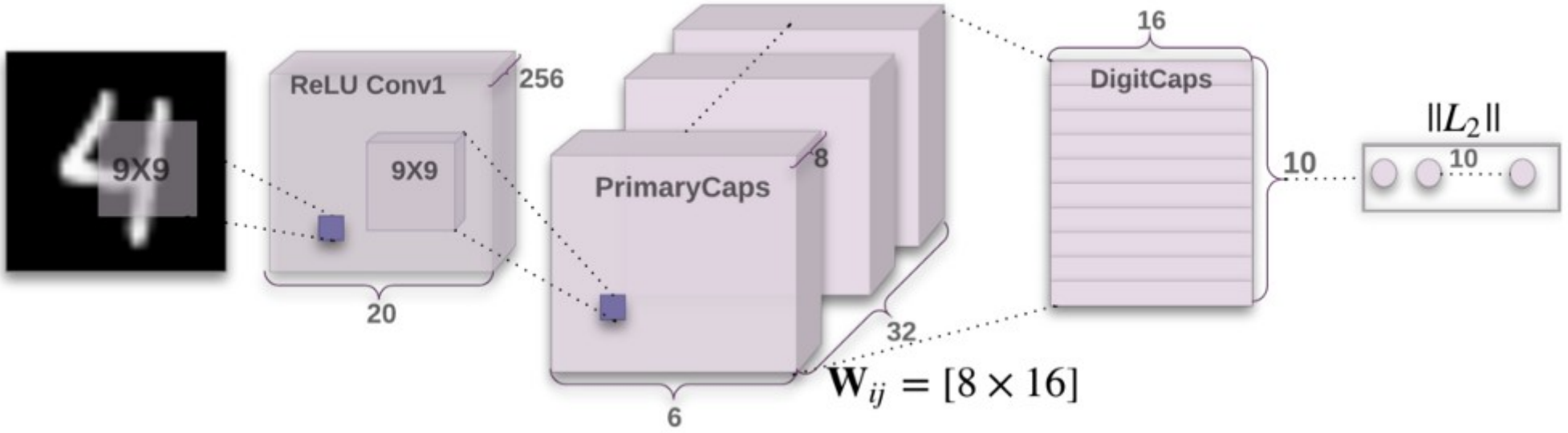
Key problem of CNN

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

Main Hinton’s proposal

Hinton argues that in order to correctly do classification and object recognition, it is important to preserve hierarchical pose relationships between object parts.

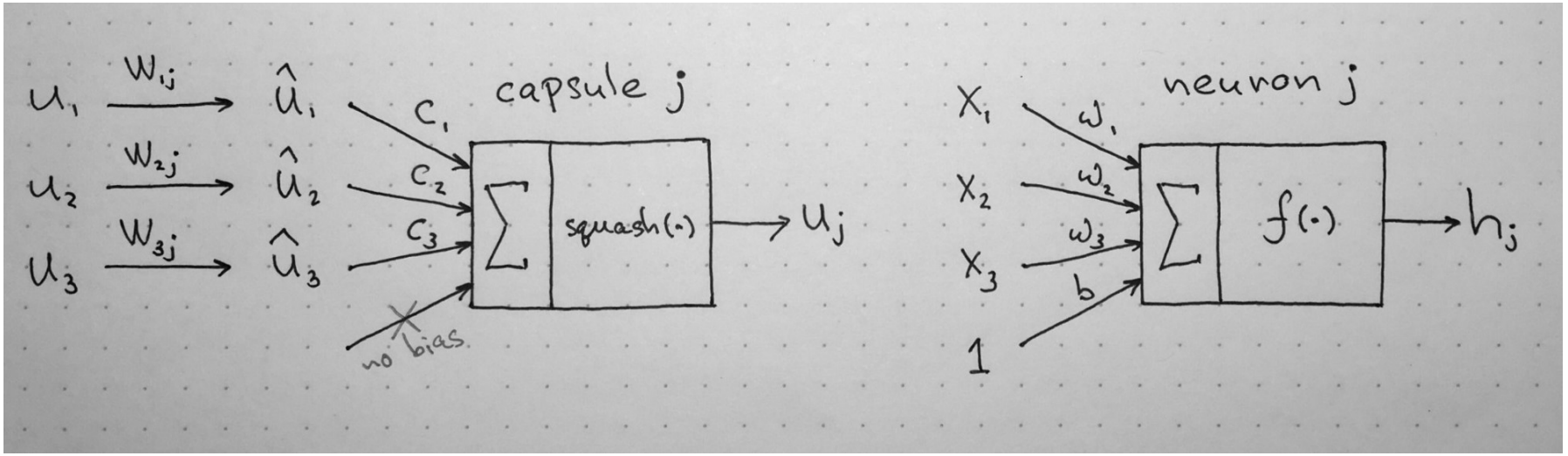
Capsule Network for MNIST



So it's clear I think. This picture describes all

What is one capsule

Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\widehat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\widehat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$	$h_j = f(a_j)$
Output		vector($\mathbf{v}_j$)	scalar(h_j)



Difference between AN and Capsule

Artificial Neuron:

1. scalar weighting of input scalars
2. sum of weighted input scalars
3. scalar-to-scalar nonlinearity

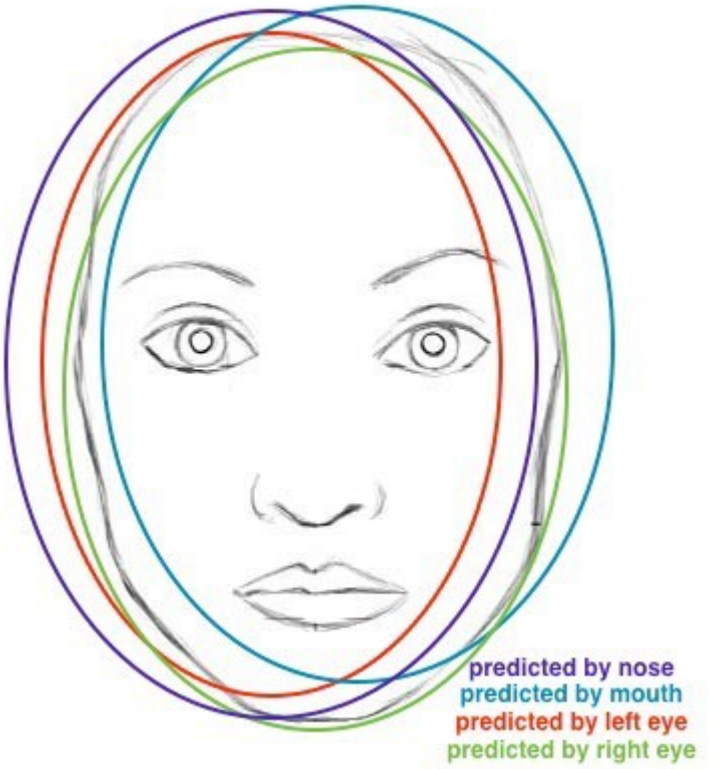
Capsule:

1. matrix multiplication of input vectors
2. scalar weighting of input vectors
3. sum of weighted input vectors
4. vector-to-vector nonlinearity

Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\hat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$	$h_j = f(a_j)$
Output		vector($\mathbf{v}_j$)	scalar(h_j)

Matrix multiplication of input vector

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i$$



Predictions for face location of nose, mouth and eyes capsules closely match: there must be a face there.

Squash

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

additional "squashing"

unit scaling

Don’t forget

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.

Dynamic routing

Procedure 1 Routing algorithm.

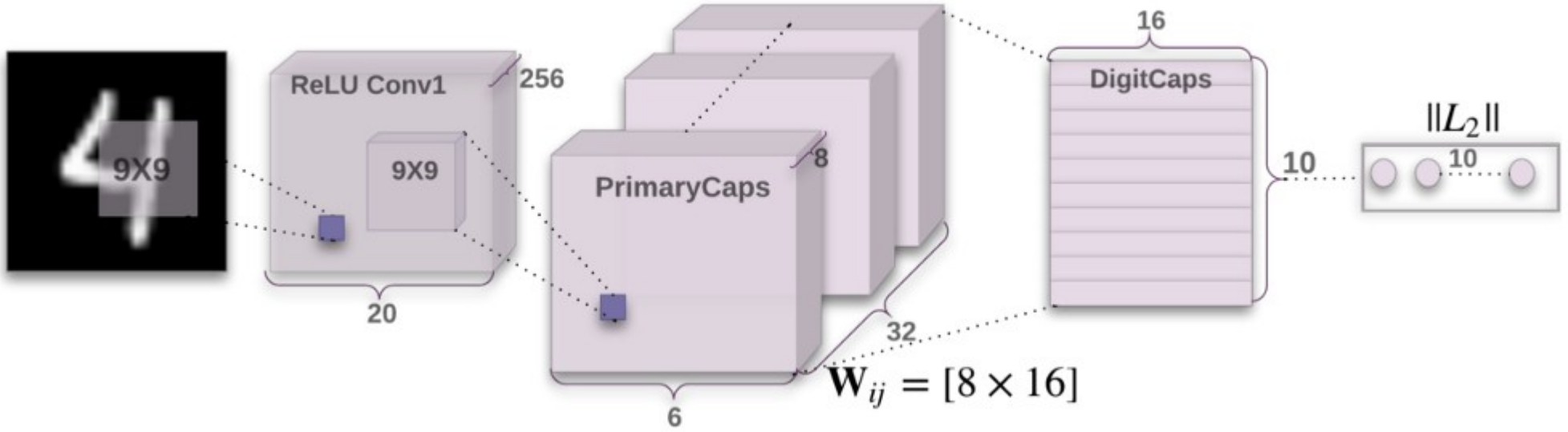
1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
7: 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 $\mathbf{v}_j$

Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\widehat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\widehat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$	$h_j = f(a_j)$
Output		vector($\mathbf{v}_j$)	scalar(h_j)

How many routing operations to use

- More iterations tends to overfit the data
- It is recommended to use 3 routing iterations in practice

Capsule Network for MNIST



The CapsNet has 2 parts: **encoder** and **decoder**. The first 3 layers are encoder, and the second 3 are decoder:

Layer 1. Convolutional layer
Layer 2. PrimaryCaps layer
Layer 3. DigitCaps layer
Layer 4. Fully connected #1
Layer 5. Fully connected #2
Layer 6. Fully connected #3

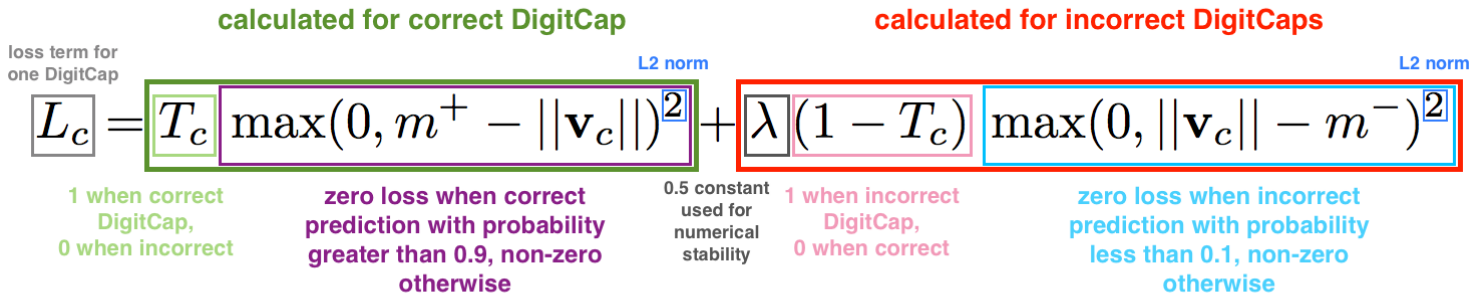
Encoder

Layer 1. Convolutional layer
Input: 28x28 image (one color channel).
Output: 20x20x256 tensor.
Number of parameters: 20992.

Layer 2. PrimaryCaps layer
Input: 20x20x256 tensor.
Output: 6x6x8x32 tensor.
Number of parameters: 5308672.

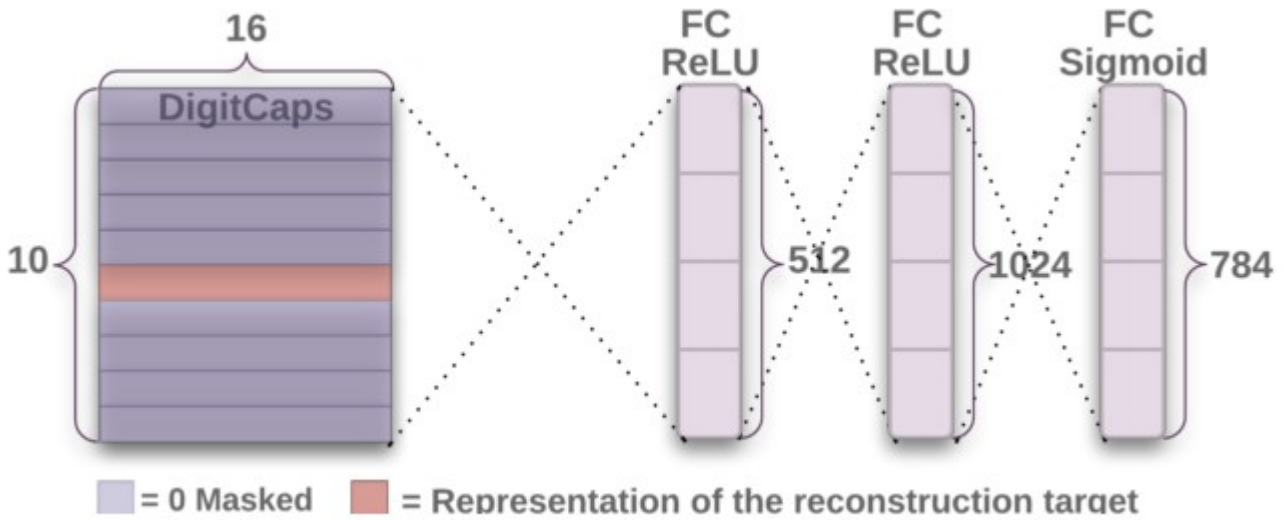
Layer 3. DigitCaps layer
Input: 6x6x8x32 tensor.
Output: 16x10 matrix.
Number of parameters: 1497600.

CapsNet Loss Function



Note: correct DigitCap is one that matches training label, for each training example there will be 1 correct and 9 incorrect DigitCaps

Decoder



Layer 4. Fully connected #1
Input: 16x10.
Output: 512.
Number of parameters: 82432.

Layer 5. Fully connected #2
Input: 512.
Output: 1024.
Number of parameters: 525312.

Layer 6. Fully connected #3
Input: 1024.
Output: 784 (which after reshaping gives back a 28x28 decoded image).
Number of parameters: 803600.

Thank you!