

# CapsuleNet

Capsules to rescue CNN: informative vectors instead of a single scalar output!



- Introduction
- Motivation - Challenges of CNNs
- Capsules
- Dynamic Routing
- Squashing function
- CapsNet Architecture
- Results
  - Mnist
  - CIFAR10
  - Extra: smallINOPRB
- Conclusions

# CapsuleNet(2017)

- Geoffrey Hinton & Google et al.
- Dynamic routing between capsules (588 citations)
- Matrix capsules with EM routing (110 citations)
- MNIST, Cifar-10 classification and reconstruction

arXiv:1710.09829v2 [cs.CV] 7 Nov 2017

## Dynamic Routing Between Capsules

Sara Sabour  
Geoffrey E. Hinton  
Caglar S. Sengür  
Tamas Frazor  
sasa@saur, frazor, geof@hinton@google.com

### Abstract

A capsule is a group of neurons whose activity vector represents parameters of a specific type of entity such as an object or a part of an object. The length of the activity vector represents the probability that the entity exists and its orientation represents its instantiation parameters. Active capsules make predictions, via transformation matrices, for the one or more higher-level capsules. When multiple predictions exist, a capsule becomes active. We show that a discriminatively trained, multi-system address state-of-the-art performance on MNIST and even on a convolutional net at accepting highly overlapping digits. It results in an unimodal output by agreement resolution. A local prefix is used to output to higher-level capsules whose activity is scalar product with the prediction coming from the lower-level cap

### 1 Introduction

Human vision ignores irrelevant details by using a carefully discerned set to ensure that only a tiny fraction of the optic array is ever processed at any given time. In a poor model to understanding how much of our knowledge of the sequence of locations and how much we glean from a single fixation, it is assumed that a single fixation gives us much more than just a single detected. We assume that our multi-layer visual system creates a piece-wise like network. We assume that the lower-level visual system is composed of a set of small groups of neurons called "capsules" (Dumais et al., 2011) and each capsule is connected to the layer above by a dynamically adjusting matrix. However, we shall assume that, for a single fixation, a piece-wise multilayer neural network like a capsule is carried from a rock. Each layer is composed of an active capsule. Using an iterative routing process, each active capsule in the layer above is fed a piece in the layer below. The higher-level iterative process will be solving the problem of assigning parts to wholes.

The activities of the neurons within an active capsule represent the features of the entity that is present in the image. These properties can include many different features such as pose (position, size, orientation), deformation (velocity). One very special property is the existence of the instantiated entity in the first place. In this paper we explore an interesting alternative which is to use a vector of instantiation parameters to represent the existence of the entity and

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach

Published as a conference paper at ICLR 2018

## MATRIX CAPSULES WITH EM ROUTING

Geoffrey Hinton, Sara Sabour, Nicholas Frazor  
Geoff Hinton  
sasa@saur, frazor, geof@hinton@google.com

### ABSTRACT

A capsule is a group of neurons whose outputs represent different properties of the same entity. Each layer in a capsule network contains many capsules. We describe a version of capsules in which each capsule has a logistic unit to represent the presence of an entity and a dot matrix which could learn to represent the relationship between that entity and the system (the pose). A capsule in one layer votes for the pose matrix of each different capsule in the layer above by multiplying its own pose matrix by trainable step-wise invariant transformation matrices that could learn to represent part-whole relationships. Each of these votes is weighted by an assignment coefficient. These coefficients are iteratively updated for each capsule using the Expectation Maximization algorithm such that the output of each capsule is routed to a capsule in the layer above that receives a cluster of similar votes. The transformation matrices are trained discriminatively by backpropagating through the iterated networks of EM between each pair of adjacent capsule layers. On the MNIST88 benchmark, capsules reduce the number of lost errors by 45% compared to the state-of-the-art. Capsules also show far more invariance to whole-body adversarial attacks than our baseline convolutional neural network.

### 1 INTRODUCTION

Conventional neural nets are based on the simple fact that a vision system needs to use the same knowledge of all locations in the image. This is achieved by using the weights of feature detectors so that features learned at one location are available at other locations. Convolutional capsules extend the sharing of knowledge across locations to include knowledge about the part-whole relationships that characterize familiar shapes. Viewpoint changes have complicated effects on part intensities but simple linear effects on the pose matrix that represents the relationship between an object or object part and the viewer. The aim of capsules is to make good use of this underlying theory, both for dealing with viewpoint variations and for recognizing viewpoint-invariant features.

Capsules use high-dimensional output vectors to represent a familiar object can be detected by looking for agreement between votes from its pose matrix. These votes come from parts that have already been detected. A part prediction is used by multiplying its own pose matrix by a learned transformation matrix that represents the suspected invariant relationship between the part and the whole. As the viewpoint changes, the pose matrices of the parts and the whole will change in a coordinated way so that any agreement between votes from different parts will persist.

Finding tight clusters of high-dimensional votes that agree in a unit of landmark votes is one way of solving the problem of assigning parts to wholes. This is inessential because we cannot find the high-dimensional pose space in the way the low-dimensional transformation space is guided to facilitate coordination. To solve this challenge, we use a fast iterative process called "routing-by-agreement" that updates the probabilities with which a part is assigned to a whole based on the intensity of the vote coming from other parts that are assigned to that whole. This is a powerful generalization principle that allows knowledge of familiar shapes to drive segmentation, rather than just using low-level cues such as proximity or agreement in color or velocity. An important difference between capsules and standard neural nets is that the activation of a capsule is based on a comparison between multiple incoming pose predictions whereas in a standard neural net it is based on a comparison between a single incoming activity vector and a learned weight vector.

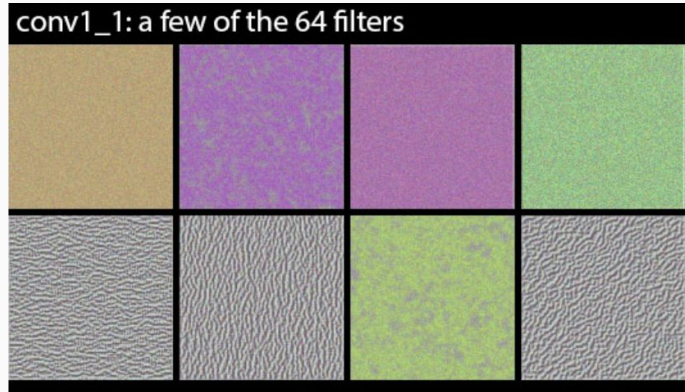
Capsule is a better representation of neurons than convolution.

Because you achieve viewpoint invariance in the activities of neurons.

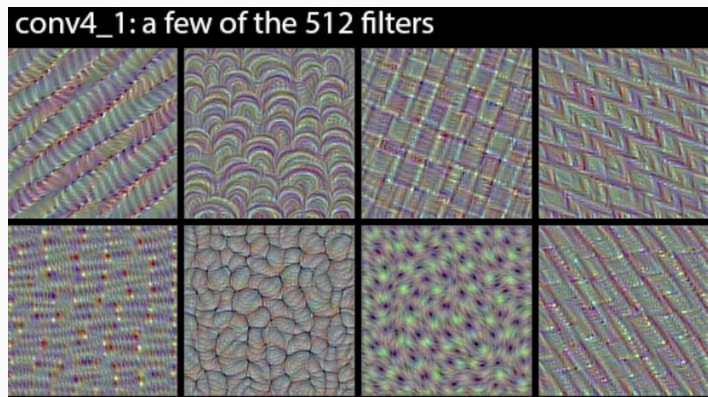
In English,  
when you see a car, you should be able to tell that it is a car from an arbitrary viewpoint.

# Motivation – Challenges of CNNs

- Kernels filter features from input



- Lower layers learn basic features, such as edges, corners



- Higher layers learn complex features

# Motivation – Challenges of CNNs



**goose**

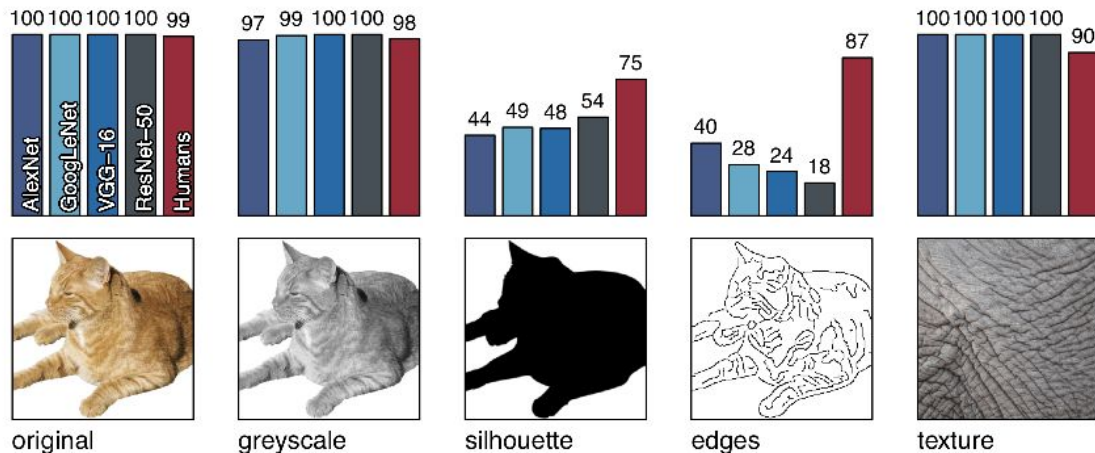


**ostrich**

- Input that maximizes a specific class
- Does not look like a real image at all

Figure 1: **Numerically computed images, illustrating the class appearance models, learnt by a ConvNet, trained on ILSVRC-2013.** Note how different aspects of class appearance are captured in a single image. Better viewed in colour.

# Motivation – Challenges of CNNs



Data augmentation can help:

- Flip
- Rotation
- Translation
- Crop
- Added noise
- Contrast
- Brightness
- Shear angle
- Style transfer
- ...

# Motivation – Challenges of CNNs



(a) Texture image

81.4% **Indian elephant**  
10.3% indri  
8.2% black swan



(b) Content image

71.1% **tabby cat**  
17.3% grey fox  
3.3% Siamese cat



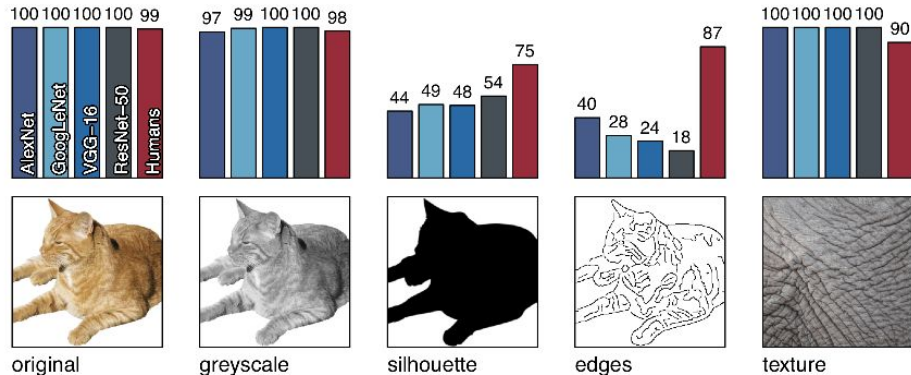
(c) Texture-shape cue conflict

63.9% **Indian elephant**  
26.4% indri  
9.6% black swan

- CNNs rely on texture too much

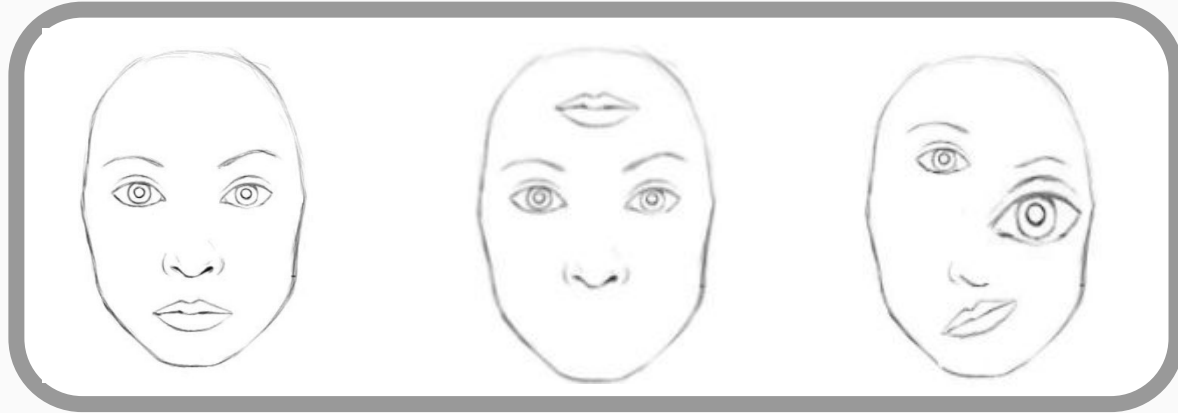
Data augmentation can help:

- Flip
- Rotation
- Translation
- Crop
- Added noise
- Contrast
- Brightness
- Shear angle
- Style transfer
- ...

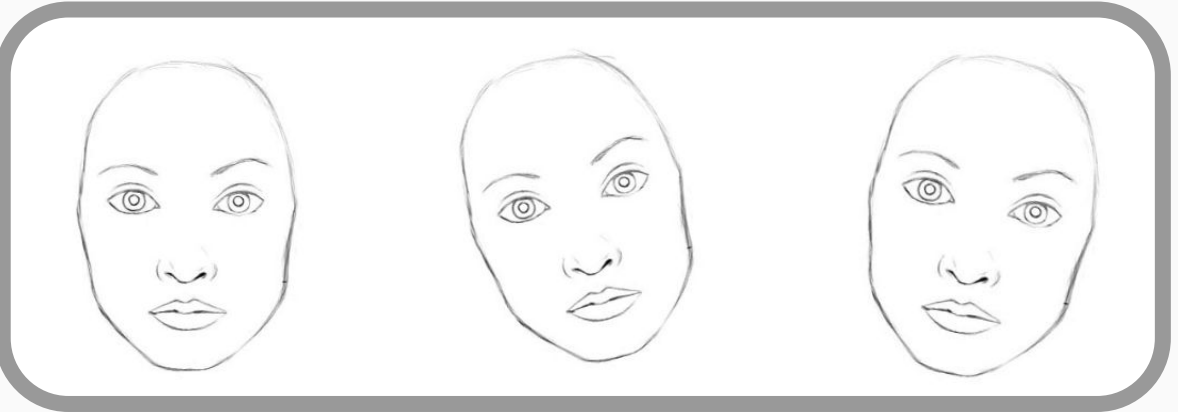




## Motivation – Challenges of CNNs



- CNN are easily fooled. All of these are recognized as faces



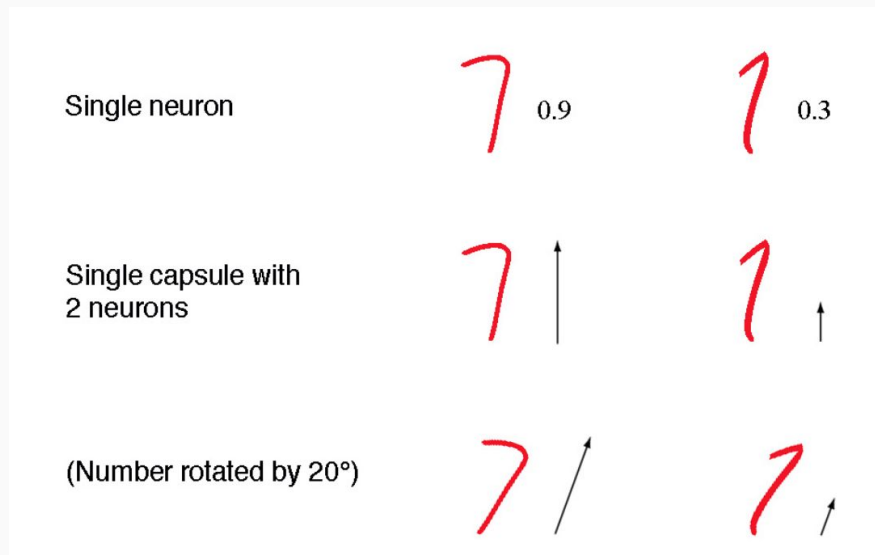
- CNN cannot easily extrapolate. This requires augmentation.

## Motivation – Challenges of CNNs

- Kernels output scalars
  - Little orientational and relative spatial relationships between features
- Max pooling loses valuable information
  - Weak spatial hierarchies between simple and complex object

# Capsules – emulate neurons better with a capsule!

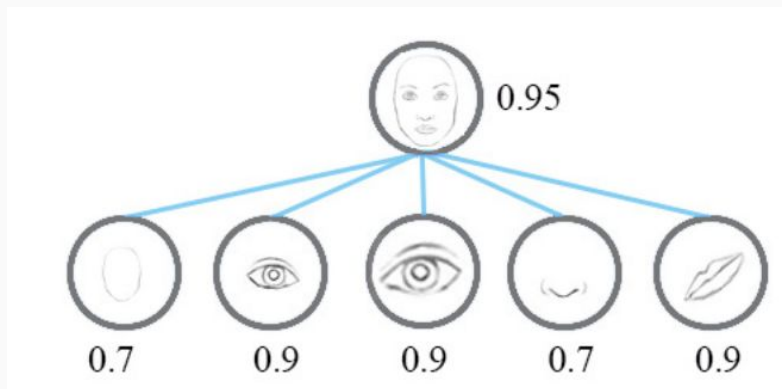
- Activation outputs a vector instead of a scalar
  - Length: probability that the entity is present within its limited domain
  - Direction: “instantiation parameters” of the input (e.g. pose, lighting and etc.)
  - Even if the direction (pose) changes, the length (probability) may stay the same.
    - Activity Equivariance



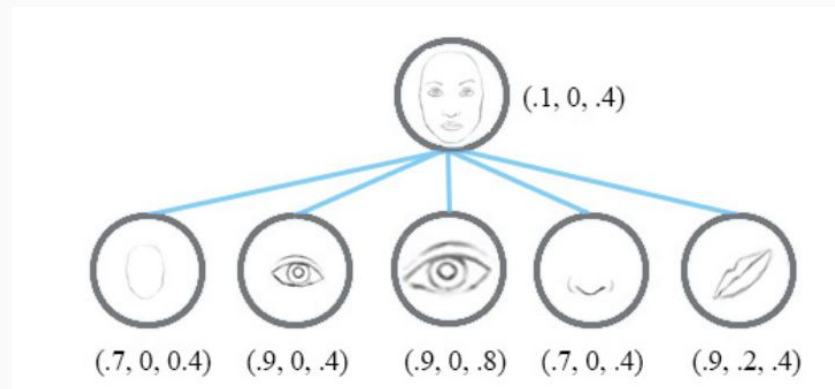
# Capsules – how does our brains work?

- We decompose hierarchical representations and do pattern matching.
- The representation is view-angle invariant.

- Takes a vector as input and outputs a vector
- Output vector encodes information about feature transformations



Traditional Convolutional Layer  
(scalar output)



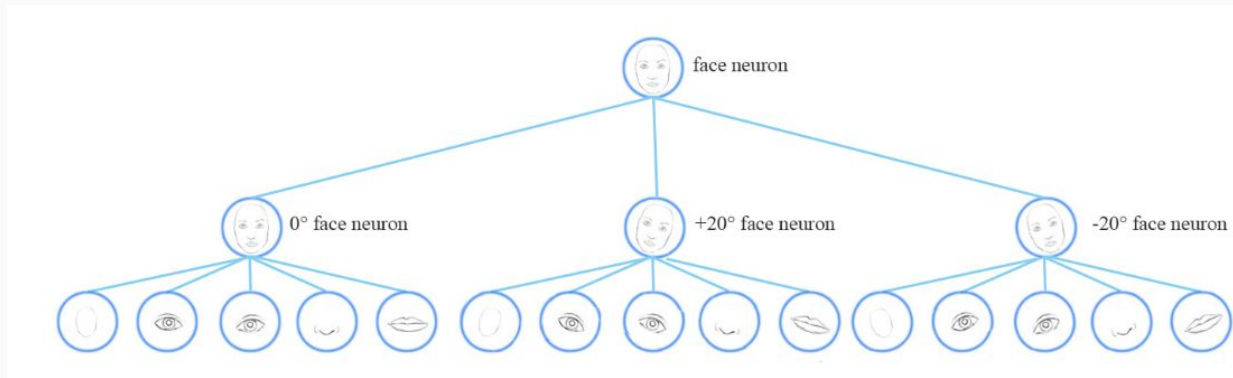
Capsule Layer  
(vector output)

## Capsule in a nutshell

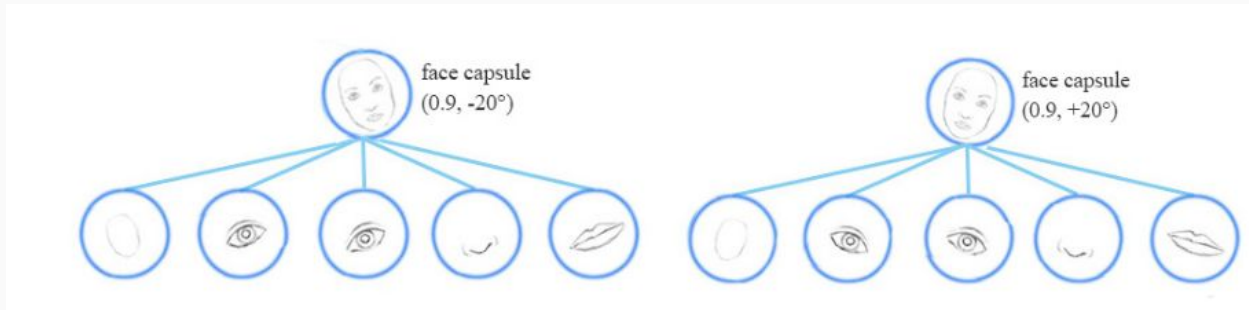
Takes a vector as input and outputs a vector

# Capsules – how does our brains work?

- We decompose hierarchical representations and do pattern matching.
- The representation is view-angle invariant.



Traditional Convolutional  
Layer  
(scalar output)



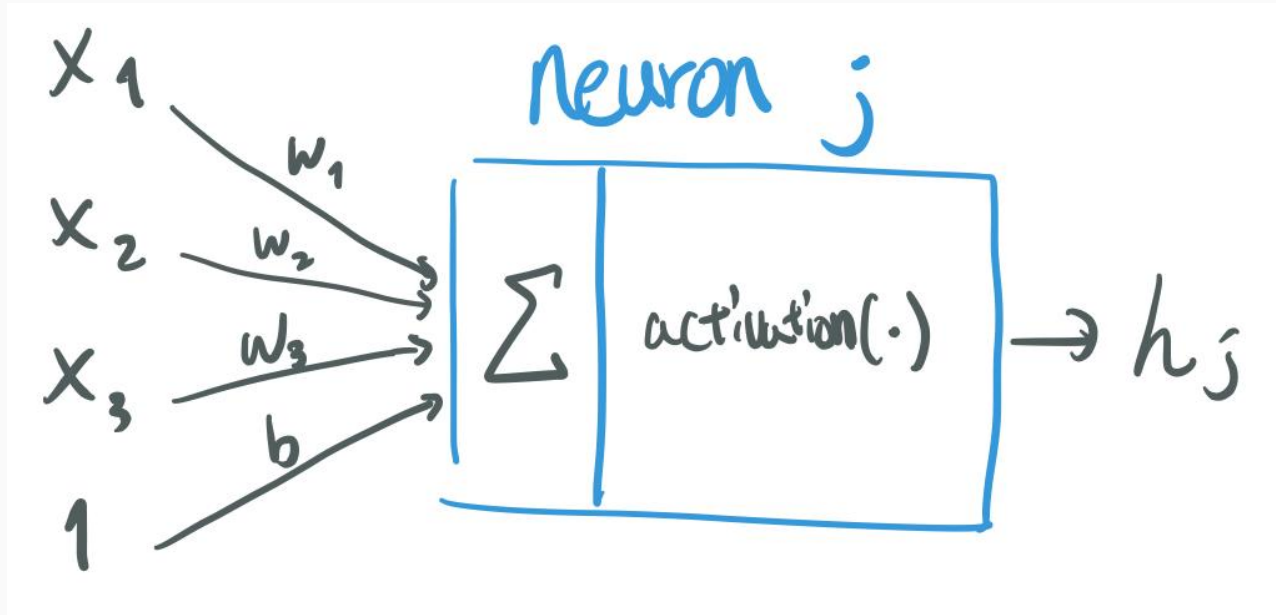
Capsule Layer  
(vector output)

Less parameters

# Intuition of Capsule

-

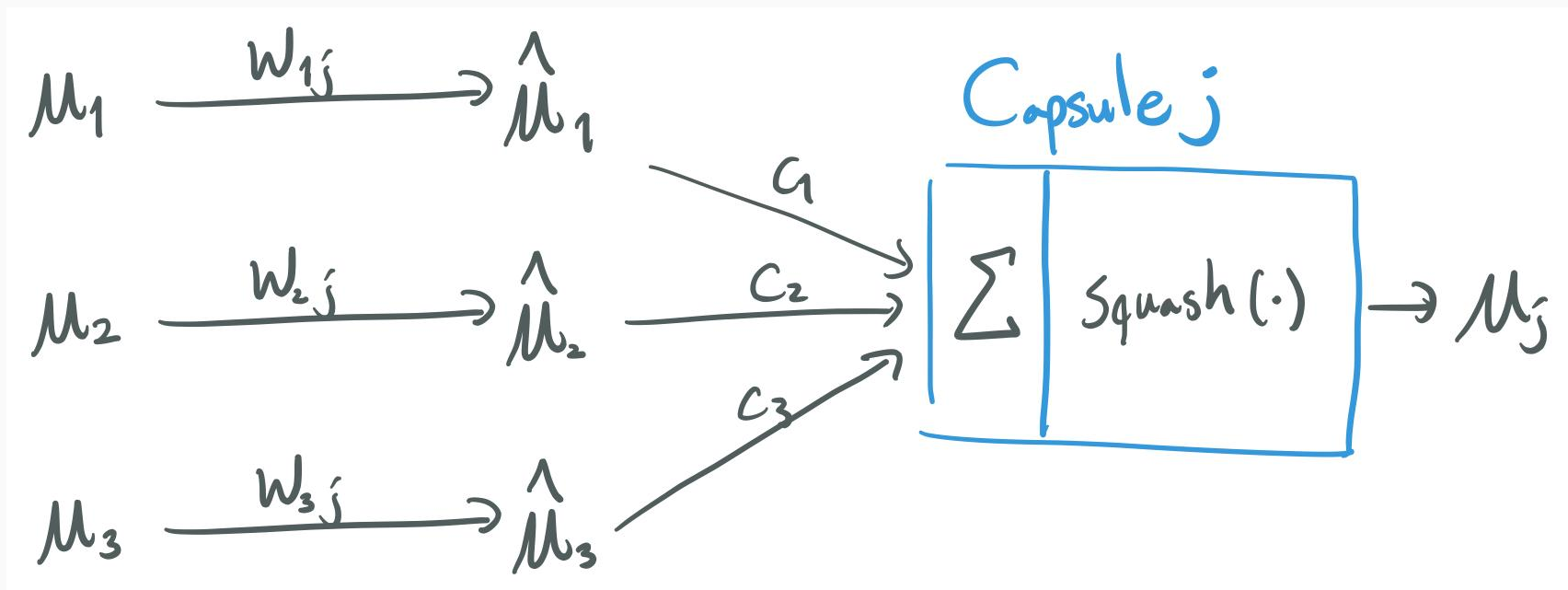
## How does a capsule work? – Traditional Neuron



$X_n$ : a scalar from previous convolution layer. Represents feature activation level

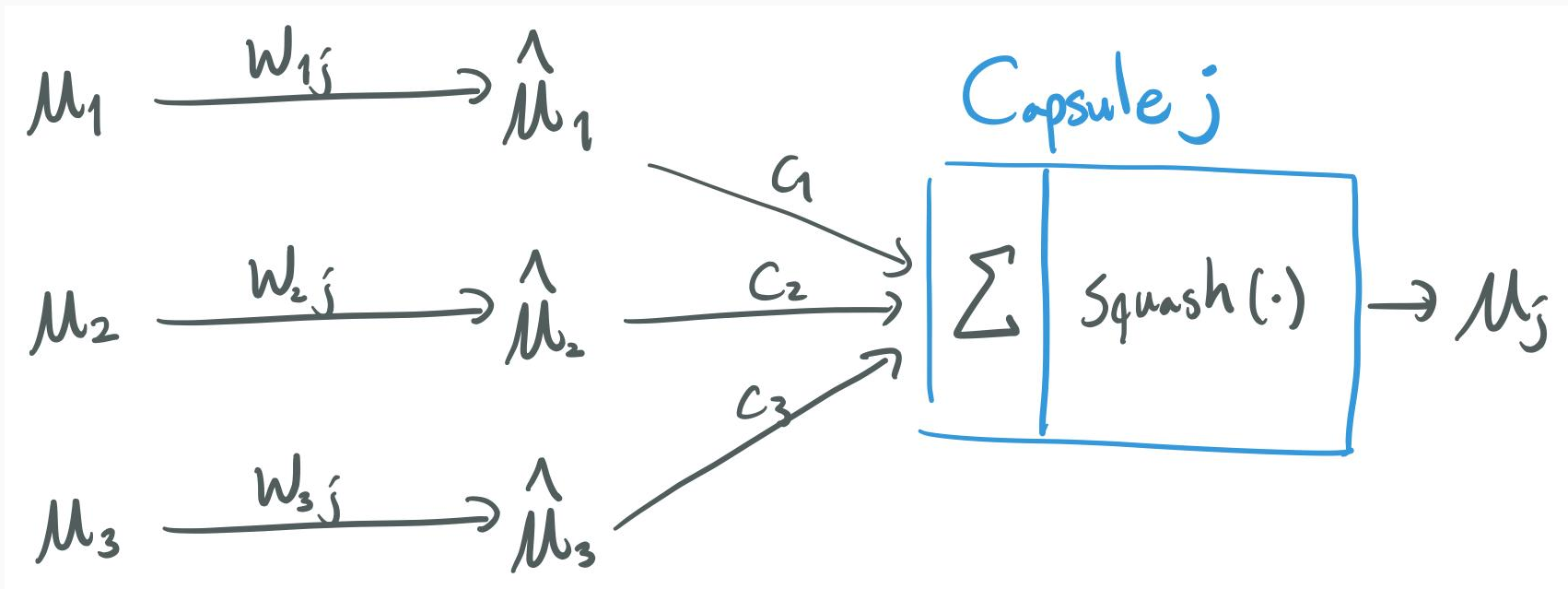


## How does a capsule work? – Capsule



- Weight matrices encode spatial and other relationships between lower level features and higher level features.
- Output vector is a predicted position of the higher level feature given the lower level feature.

## How does a capsule work?



- Non-negative scalar weight  $c_n$  is determined using "dynamic routing".
- $\text{Sum}([c_1, \dots, c_n]) = 1$
- $\text{Len}([c_1, \dots, c_n]) = \text{\#Number of the next level capsules}$

# How does a capsule work? – Capsule vs. Traditional Neuron

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( $h_j$ )	

## How does a capsule work? – Dynamic Routing

---

### Procedure 1 Routing algorithm.

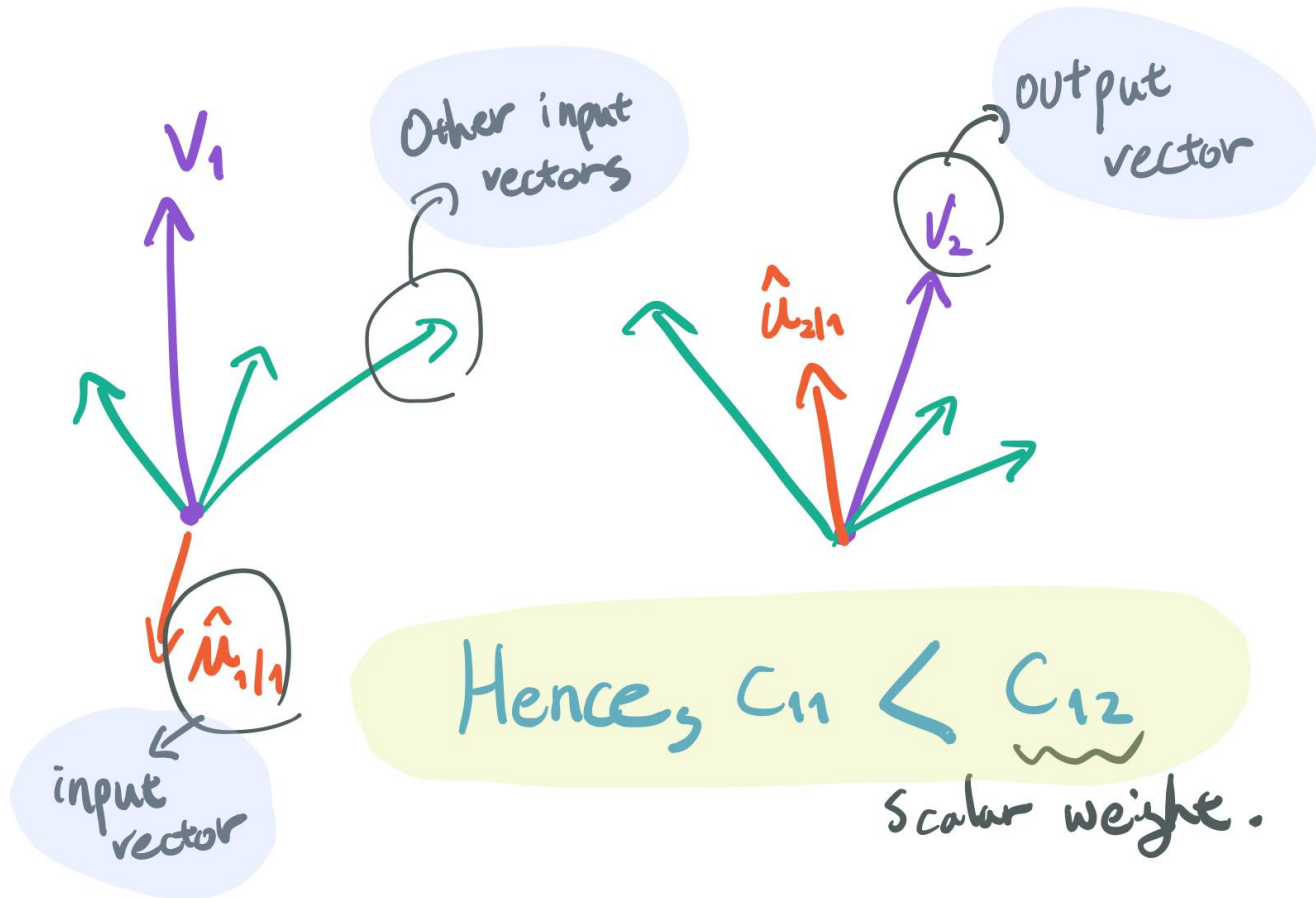
---

```
1: procedure ROUTING( $\hat{\mathbf{u}}_{j|i}, r, l$ )
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  $l$ :  $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$  ▷ 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  $(l + 1)$ :  $\mathbf{v}_j \leftarrow \text{squash}(\mathbf{s}_j)$  ▷ squash computes Eq. 1
7:     for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$ 
return  $\mathbf{v}_j$ 
```

---

- $\hat{\mathbf{u}}$ : output of previous level capsule
- $r$ : routing iteration, (3 is recommended)
- $l$ : previous level
- $\mathbf{v}_j$ : output of next level capsule
- $b_{ij}$ : temporary coefficient holder. At the end, it's stored to  $c_{ij}$

# How does a capsule work? – Dynamic Routing



## How does a capsule work? – Squashing as nonlinearity

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

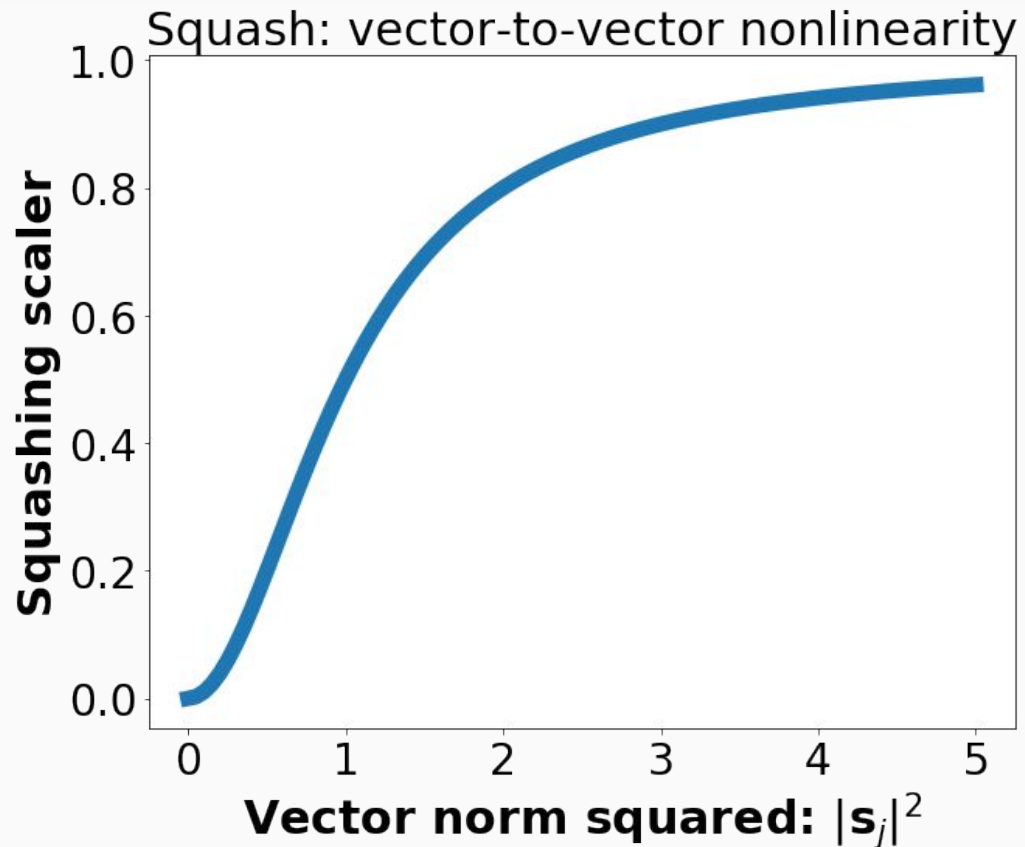
additional “squashing”      unit scaling

- Length of short vectors  $\Rightarrow \sim 0$
- Length of long vectors  $\Rightarrow \sim 1$

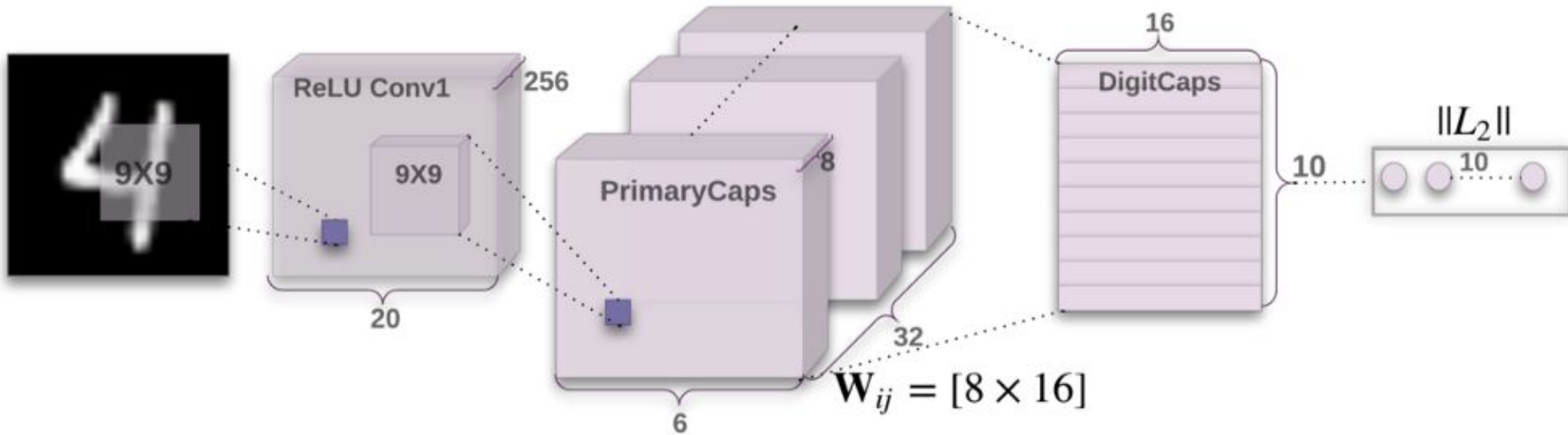
## How does a capsule work? – Squashing as nonlinearity

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

additional “squashing”



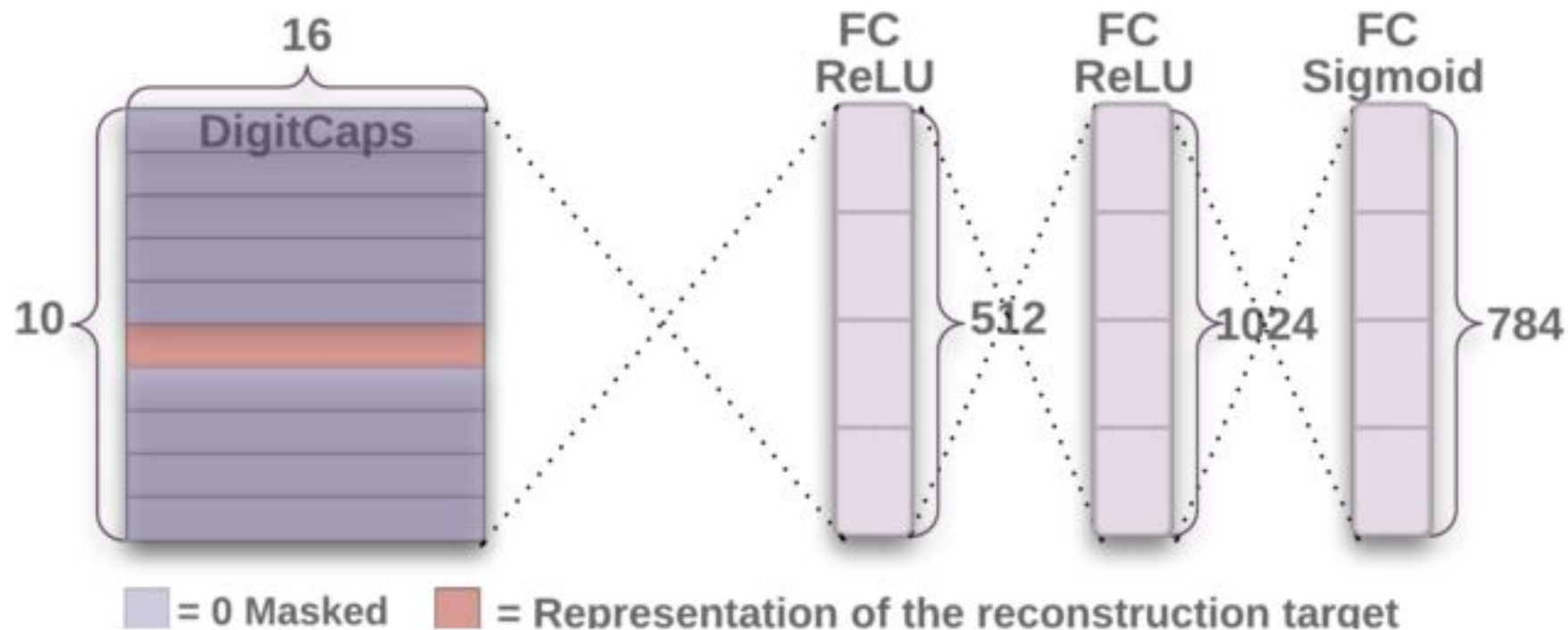
## CapsNet architecture – Encoder (classifier)



- 2D Convolutional layer: Convert pixel intensities to the activities of local feature detectors
- PrimaryCaps layer (convolutional): Invert rendering process
- DigitCaps layer (fully connected):



# CapsNet architecture – Decoder (reconstruction)



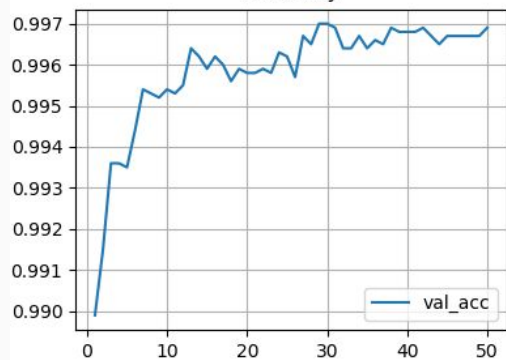
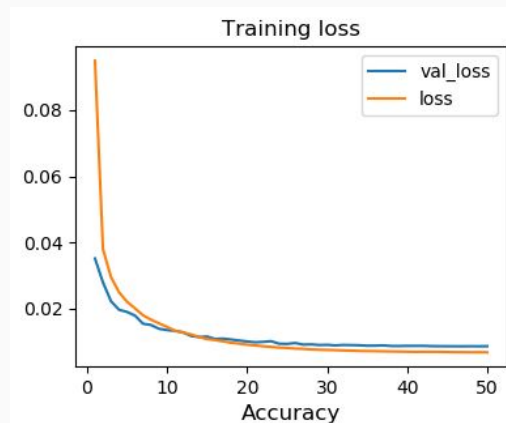
$$L_k = T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda (1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2$$

- Calculate loss for each capsule at the top-level digit capsule,
  - i.e. for each class
- $T_k = 1$  iff a class exists in an image
- $m^+$ : 0.9
- $m^-$ : 0.1
- $\lambda$ : down-weighting for initial learning iterations
- Total loss:  $\text{Sum}([L_1, \dots, L_k])$






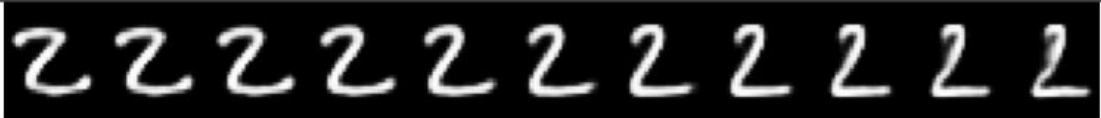
$$\text{Loss} = \text{Loss\_margin} + 0.0005 * \text{MSE}(\text{reconstructed\_image}, \text{input\_image})$$

# Experiment – MNIST

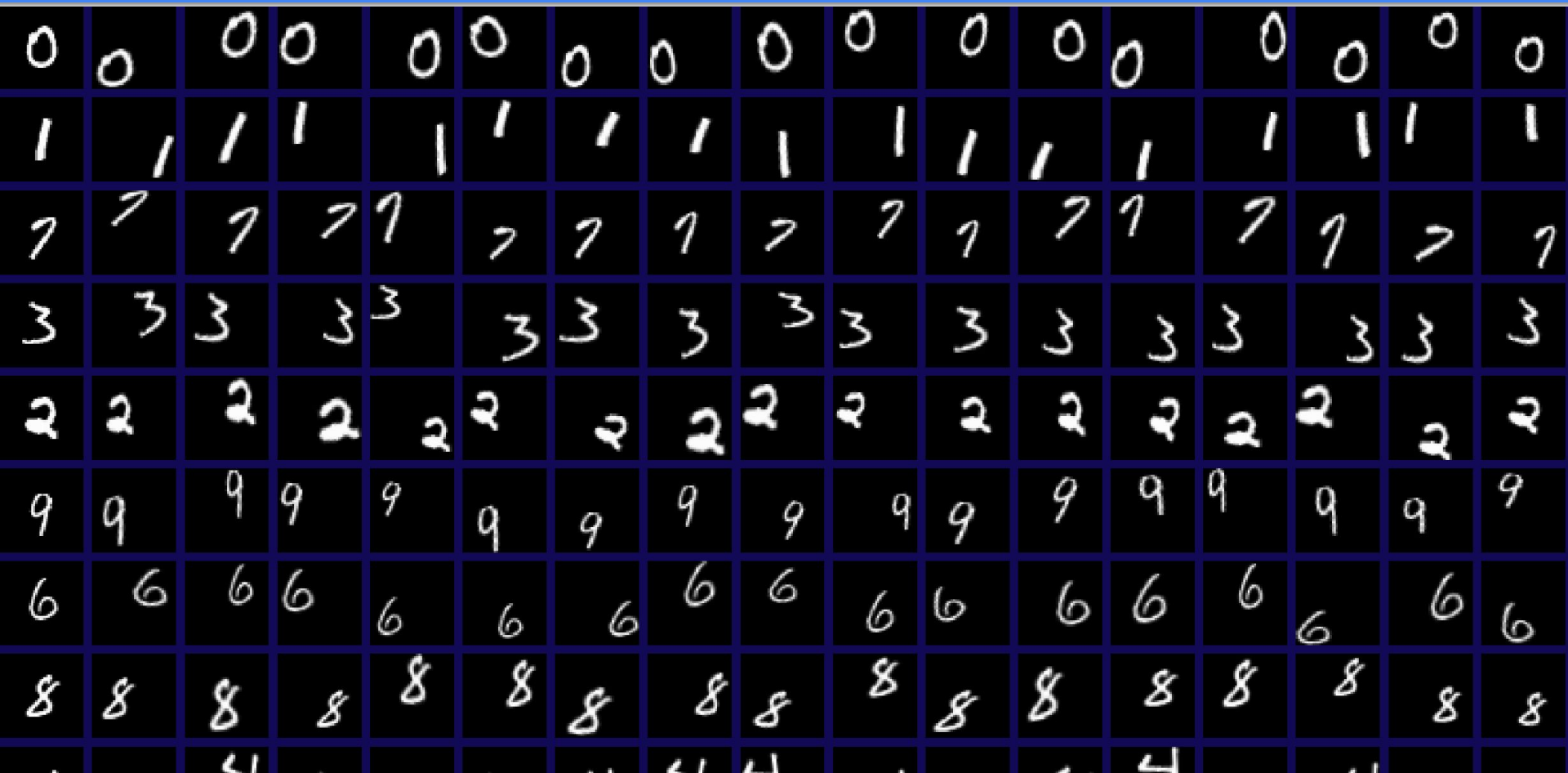
- accuracy: 99.7%
- loss: 0.00855



## CapsNet architecture – Interpretable activation vectors

Scale and thickness	
Localized part	
Stroke thickness	
Localized skew	
Width and translation	
Localized part	

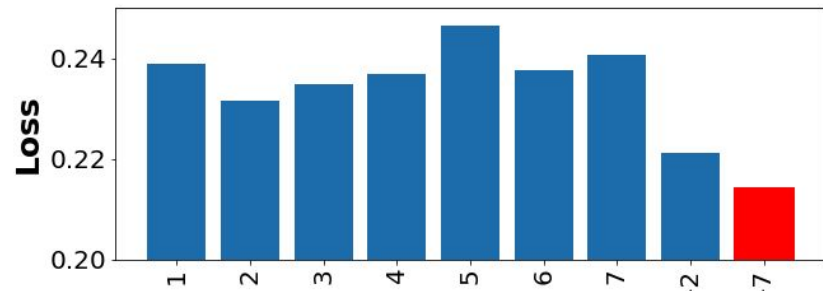
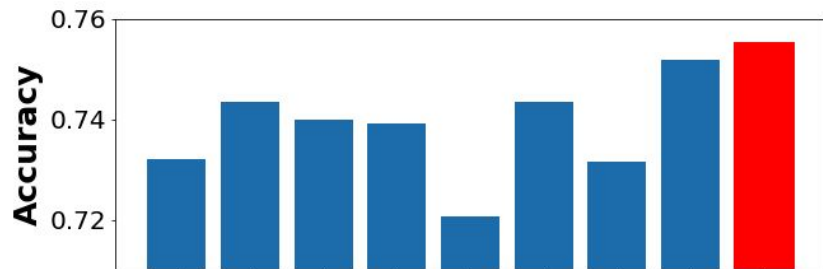
# CapsNet architecture – Robustness to affine transformation



# Experiment – CIFAR-10

- 32x32x3 image classification
- 10 classes, SOTA: 99%, Paper: 89.4%

CapsNet CIFAR-10



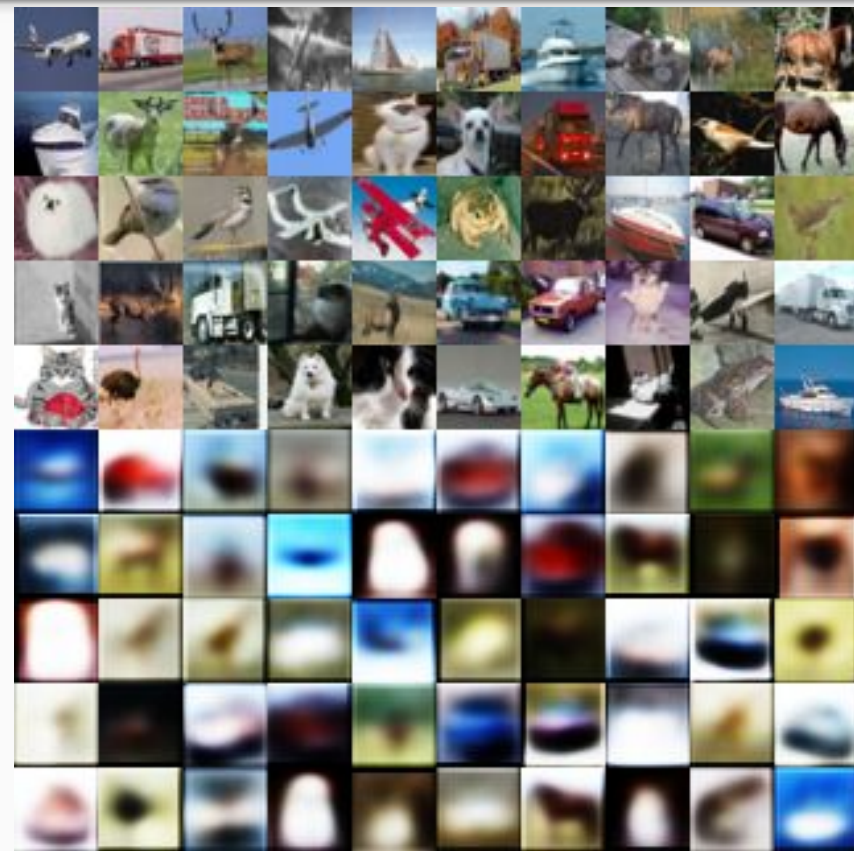
Model ID



# Experiment – CIFAR-10



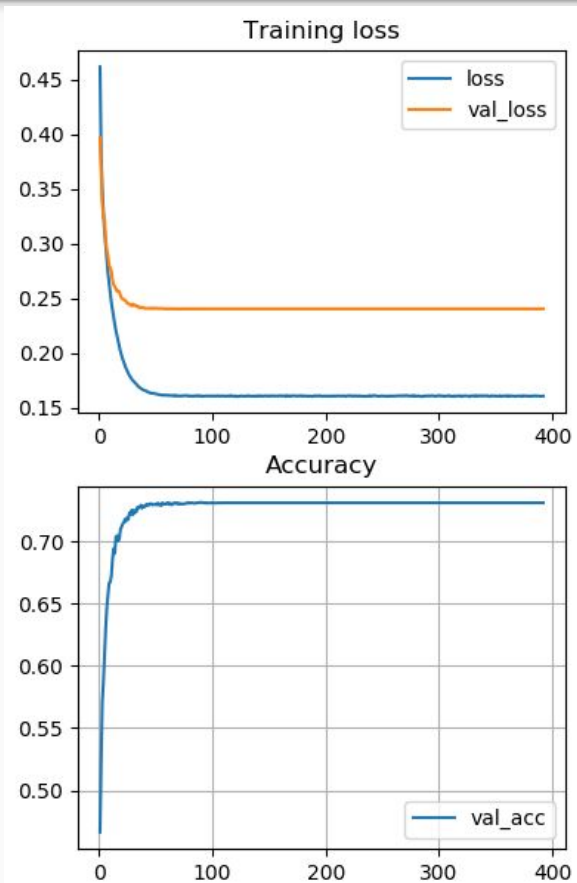
1st Epoch



1000th Epoch



# Experiment – CIFAR-10



1000th Epoch

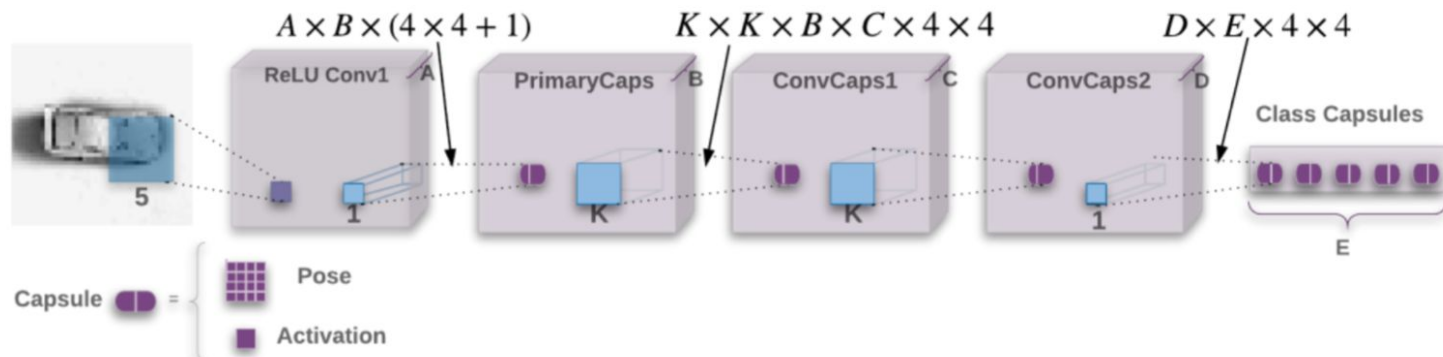
## Extra - smallNORB (Dynamic Routing with EM)



- *smallNORB* dataset (48 600 images)
  - 96x96 images
  - 5 classes
  - 10 instances per class
  - 18 azimuths per instance
  - 9 elevations per instance
  - 6 lightning conditions

# Extra - Dynamic Routing with EM

- EM algorithm instead of dynamic routing



Test set	Azimuth		Elevation	
	CNN	Capsules	CNN	Capsules
Novel viewpoints	20%	13.5%	17.8%	12.3%
Familiar viewpoints	3.7%	3.7%	4.3%	4.3%

- Capsules are convolutions with block non-linearity and routing
- Capsules require less parameters than conv (6.8M vs. 35.4M)
  - However, the routing procedure involves slow iterations
- Capsules try to build better model hierarchical relationships inside of internal knowledge representation of an NN.
- Nonetheless, capsule networks are not very popular yet.

# References

- <https://pechyonkin.me/capsules-4/>
- <https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-capsule-networks>
- <https://github.com/sekwiatkowski/awesome-capsule-networks>
- <https://www.youtube.com/watch?v=pPN8d0E3900>
- <https://jhui.github.io/2017/11/03/Dynamic-Routing-Between-Capsules/>
- <https://jhui.github.io/2017/11/14/Matrix-Capsules-with-EM-routing-Capsule-Network/>
- Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic Routing Between Capsules." *ArXiv:1710.09829 [Cs]*, October 26, 2017. <http://arxiv.org/abs/1710.09829>.
- Hinton, Geoffrey, Sara Sabour, and Nicholas Frosst. "MATRIX CAPSULES WITH EM ROUTING," 2018, 15. <https://openreview.net/pdf?id=HJWLfGWRb>
- Geirhos, Robert, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. "ImageNet-Trained CNNs Are Biased towards Texture; Increasing Shape Bias Improves Accuracy and Robustness," September 27, 2018. <https://openreview.net/forum?id=Bygh9j09KX>.
- Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps." *ArXiv:1312.6034 [Cs]*, December 20, 2013. <http://arxiv.org/abs/1312.6034>.
- Hernández-García, Alex, and Peter König. "Do Deep Nets Really Need Weight Decay and Dropout?" *ArXiv:1802.07042 [Cs]*, February 20, 2018. <http://arxiv.org/abs/1802.07042>.
- <https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html>