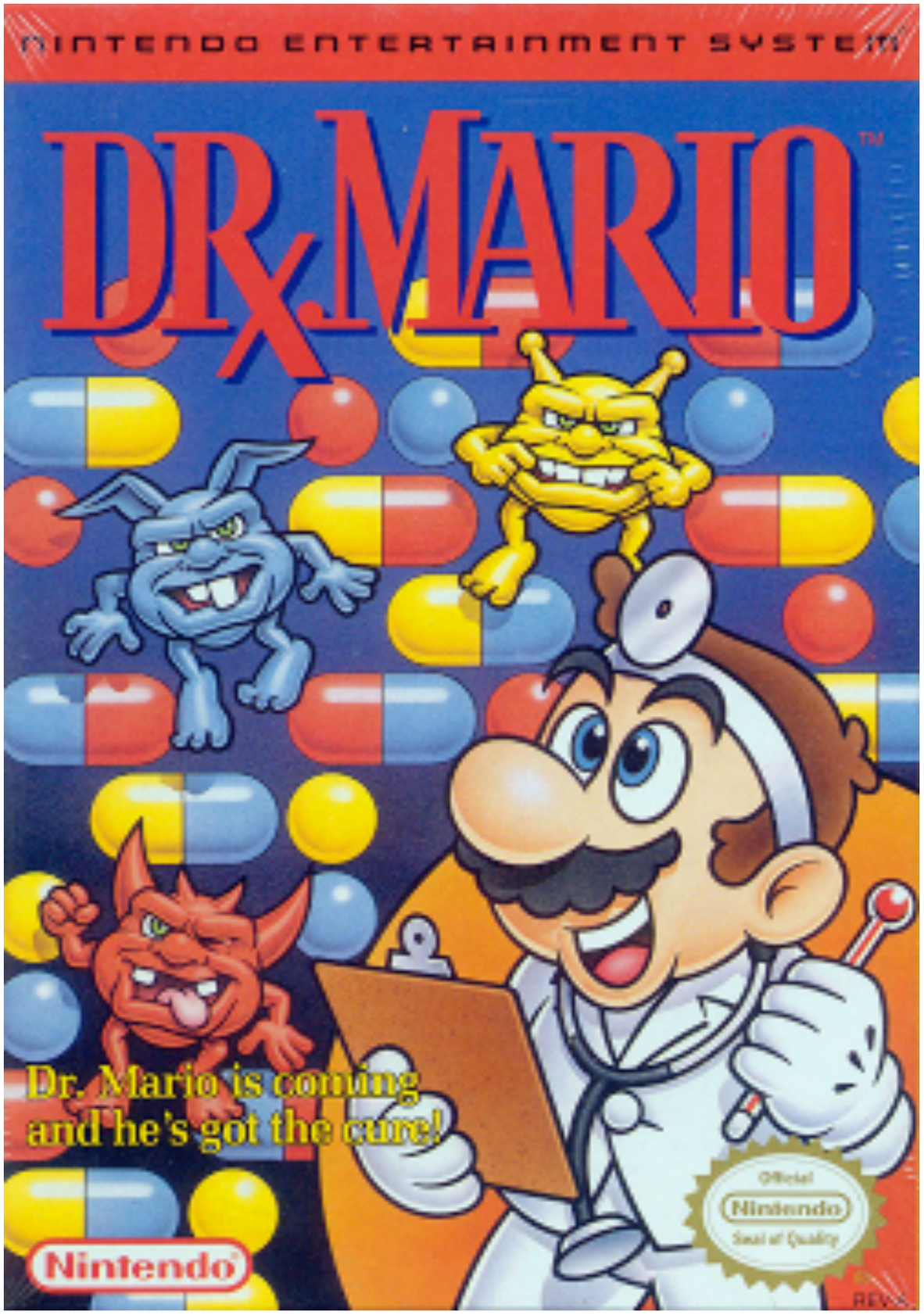
Capsule Networks

Eric Mintun



Motivation

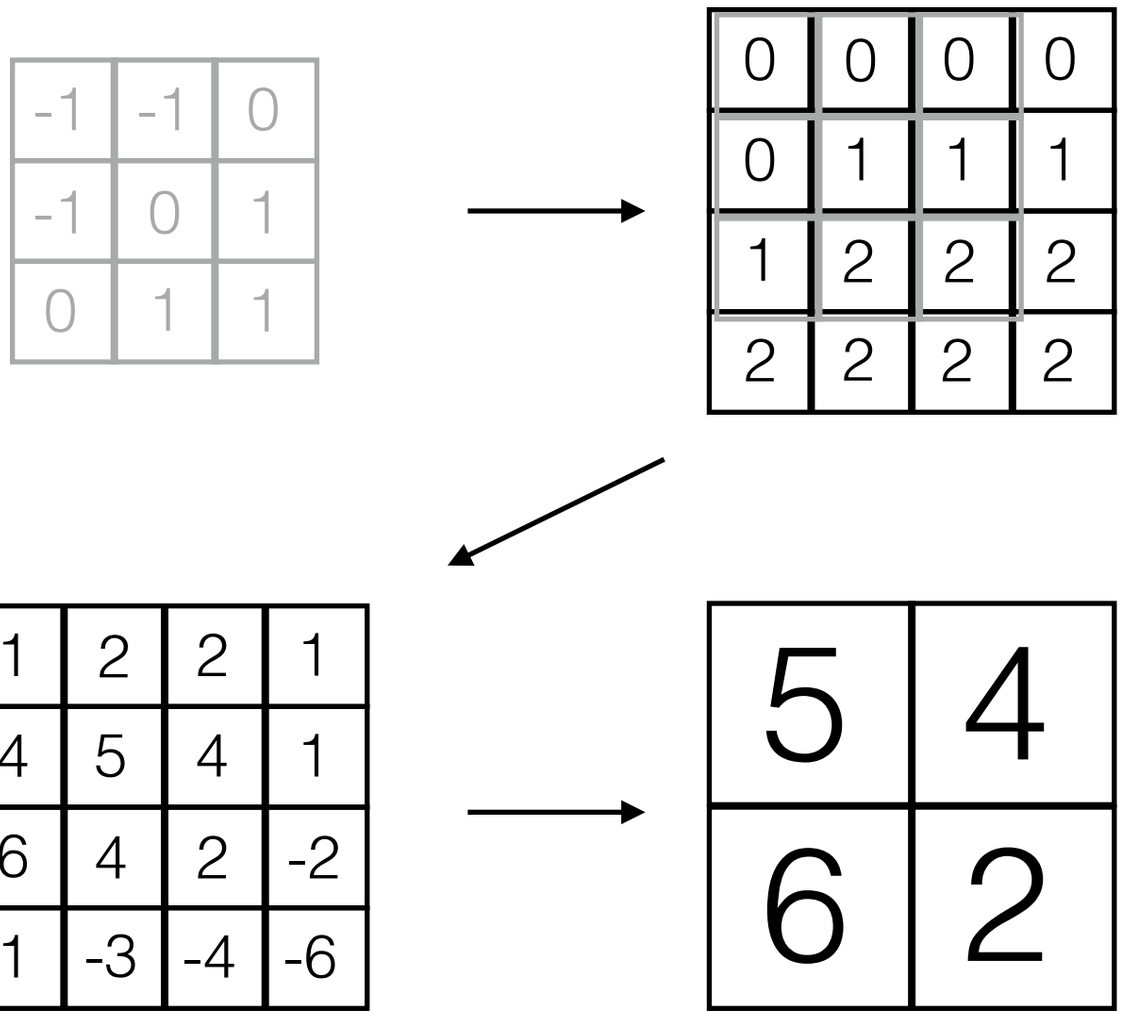
- An improvement* to regular Convolutional Neural Networks.

- Two goals:
  - Replace max-pooling operation with something more intuitive.
  - Keep more info about an activated feature.

- Not new, but recent interest because of state-of-the-art results in image segmentation and 3D object recognition.

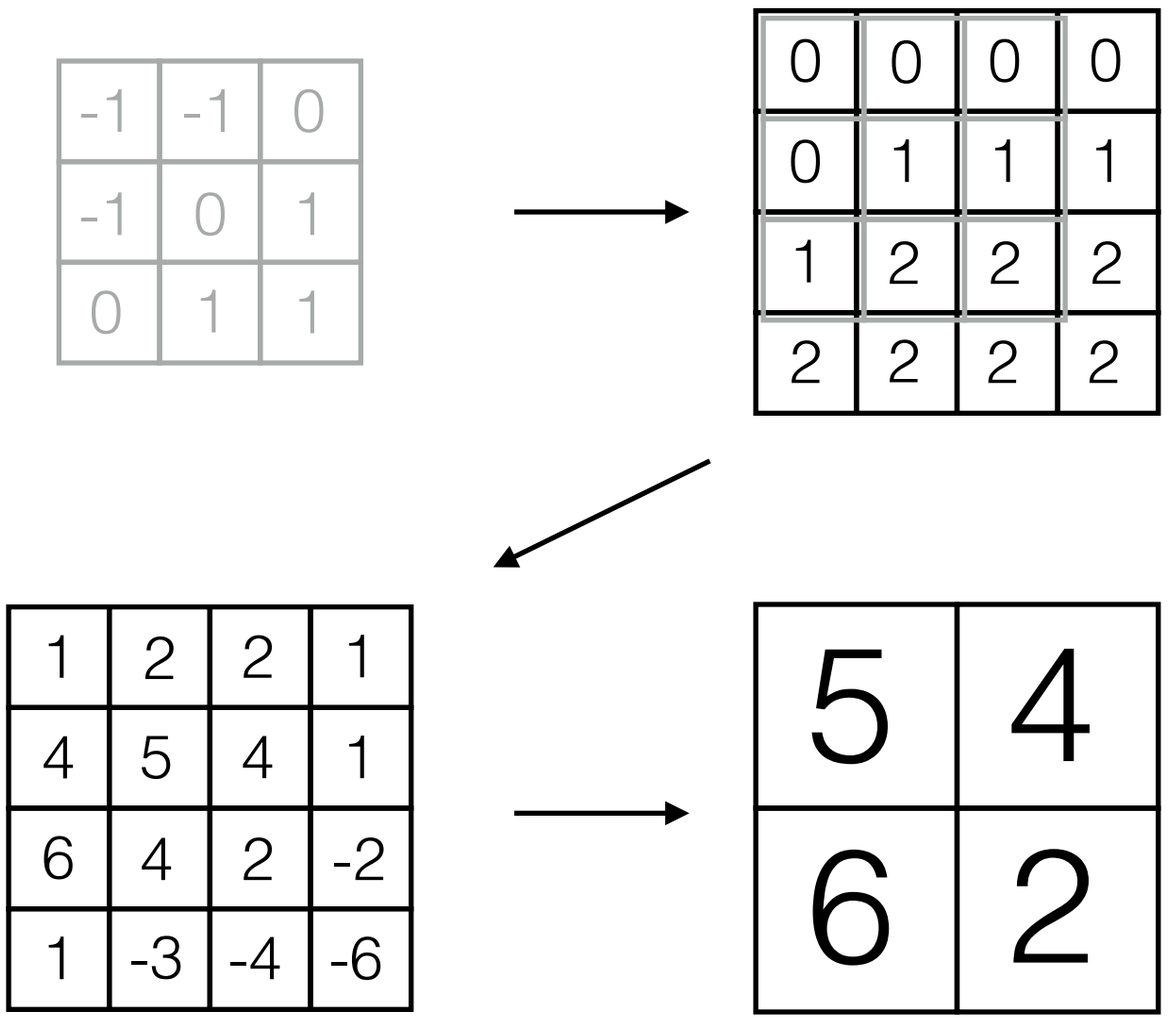
*Your milage may vary

CNN Review



- CNN architecture bakes in translation invariance.
- Convolution looks for same feature at each pixel.
- Max-pooling throws out location information.

CNN Issues



- Only involves position of feature, not orientation.
- Translation is a linear transform, but CNN doesn't represent this.
- Grid representation inefficient when features are rare.
- Intermediate translation invariance is bad.

Capsules

- Capsules have two steps:
- Apply pose transform between all lower capsules and upper capsules:

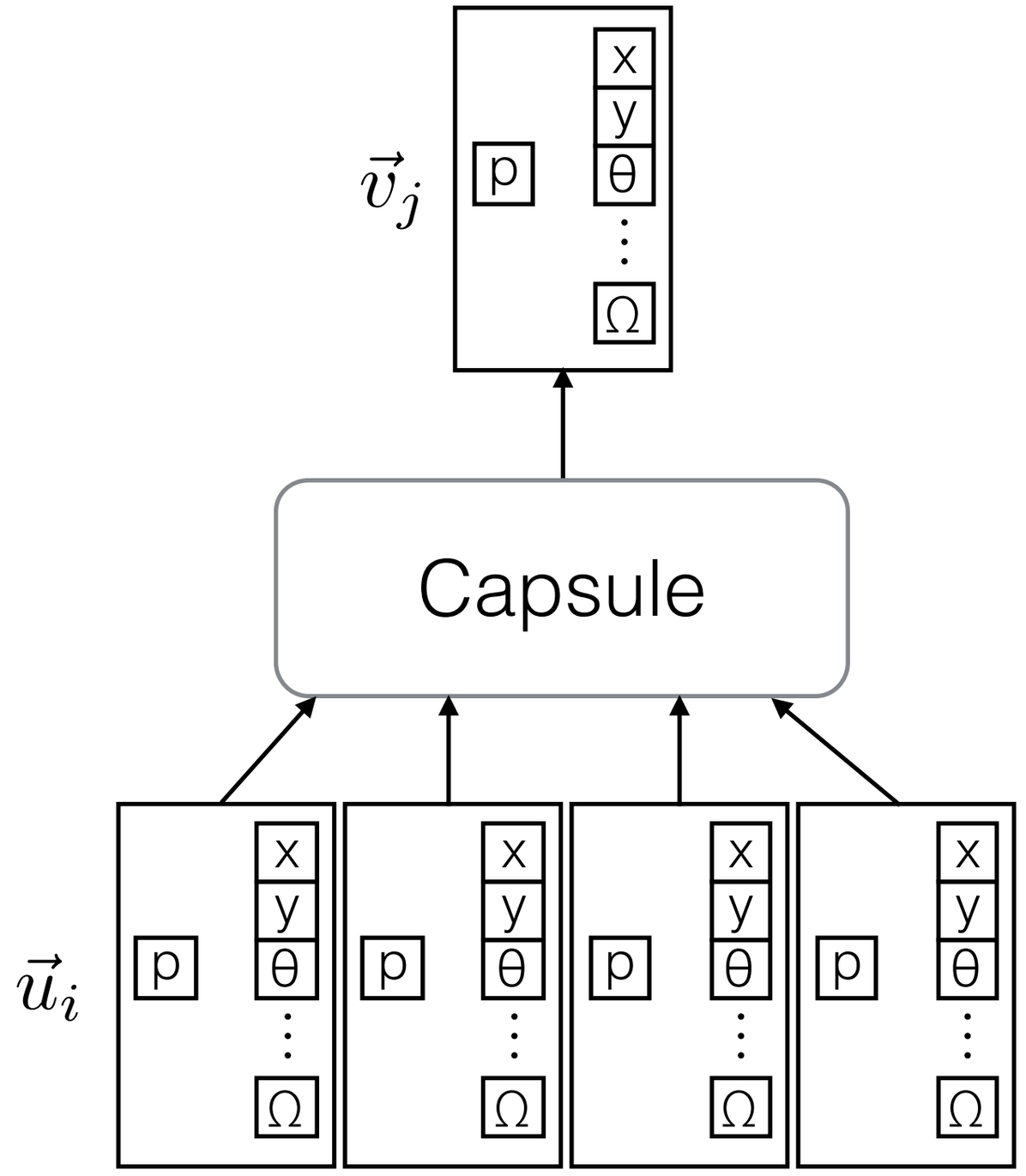
$$\hat{u}_{ij} = W_{ij}\vec{u}_i$$

  - Transformation matrices learned by back propagation.

- Route lower level capsules to higher level capsules:

$$v_j = \sum_i c_{ij}\hat{u}_{ij}\ , \qquad \sum_i c_{ij} = 1$$

  - Weights determined dynamically.
  - Activations factor into this step.



Pose Transformations

- W_{ij} : given pose of feature i, what is predicted pose of higher level feature j?

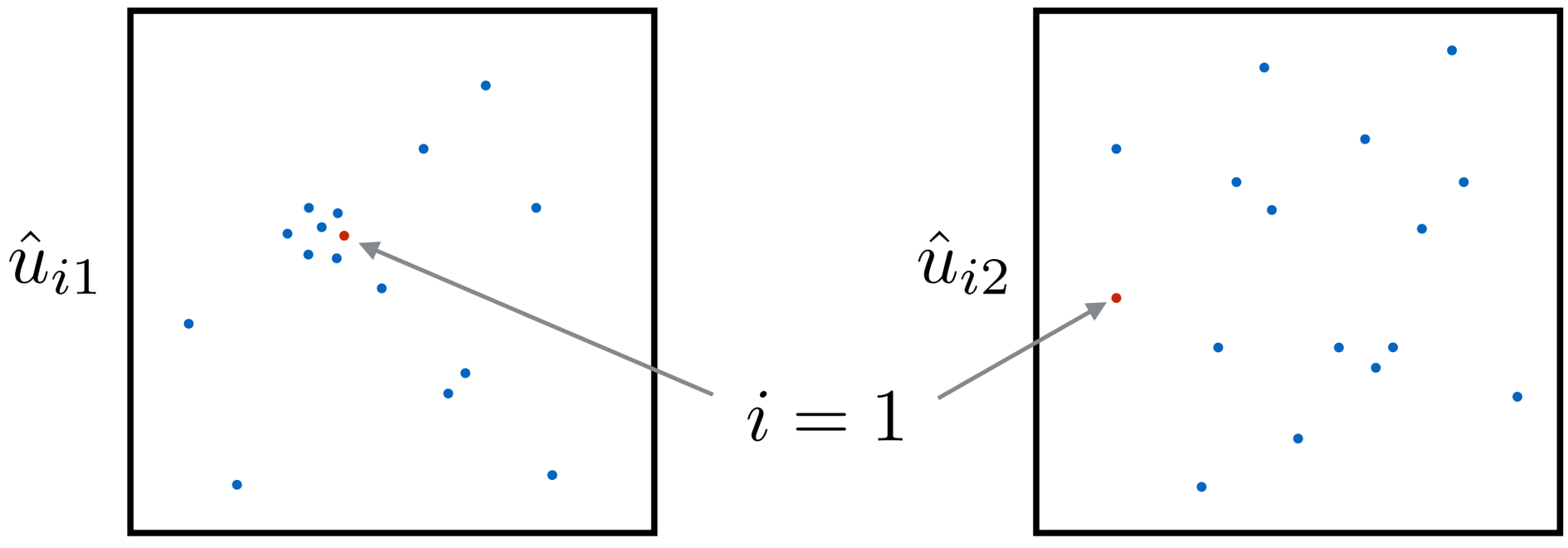
$i = 1$ $j = 1$ $j = 2$

W_{11} : rotate 135˚ CCW, rescale by 1, translate (0,-1).

W_{12} : rotate 45˚ CCW, rescale by 2, translate (0,-4).

Routing

- c_{ij}: which feature j does feature i think it is a part of.
- Determined via "routing by agreement": if many features i predict the same pose for feature j, it is more likely j is the correct higher level feature.



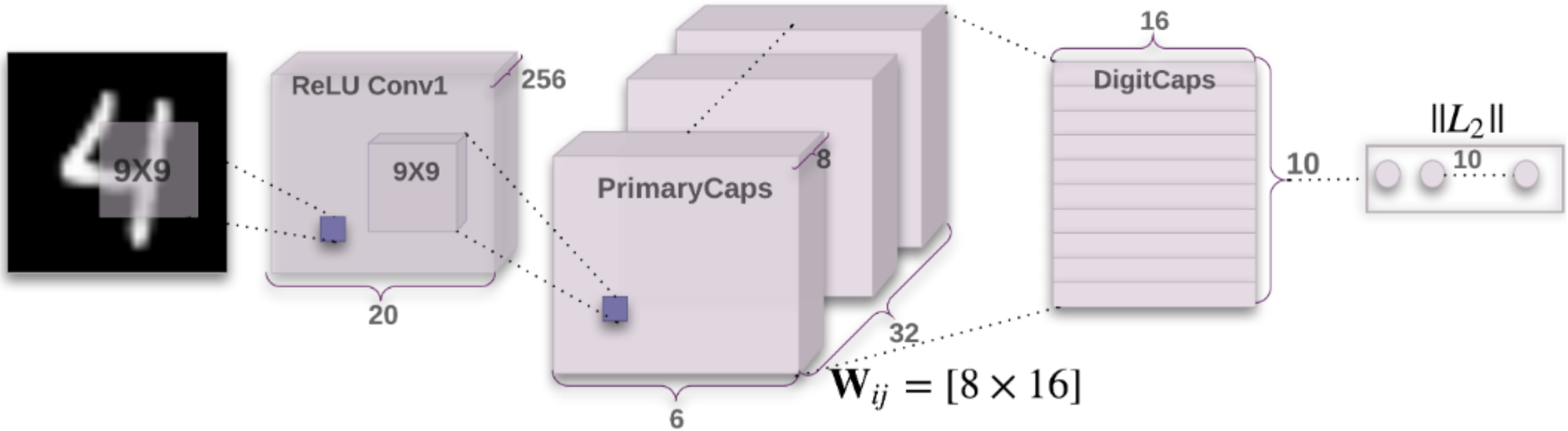
Increase c_{11}, decrease c_{12}.

Specific Models

- Two separate papers give different explicit models.
- Model 1, from “Dynamic Routing Between Capsules”, Sabour, Frosst, Hinton, 1710.09829)
    - State-of-the-art image segmentation
    - Few capsule layers
    - Generic poses with simple routing
- Model 2, from “Matrix capsules with EM routing,” anonymous authors, openreview.net/pdf?id=HJWLfGWRb
    - State-of-the-art 3D object recognition
    - More capsule layers
    - Structured poses with more advanced routing

Model 1

- From “Dynamic Routing Between Capsules”, Sabour, Frosst, Hinton, 1710.09829



- Get pixels into capsule poses using convolutions and backprop.
- ReLU between convolutions. Second convolution has stride 2.

Primary Capsules

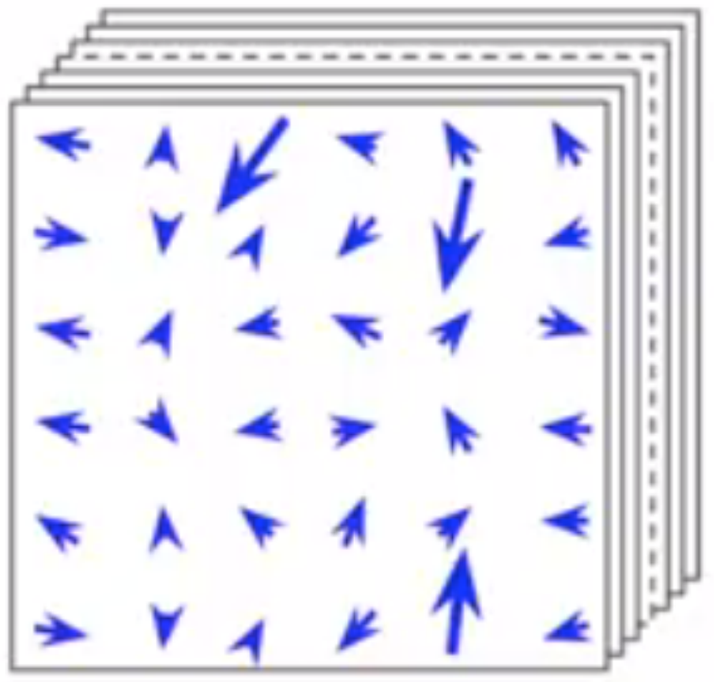
- Cool visual description of primary capsules in Aurélien Géron's "How to implement CapsNets using TensorFlow" (youtube.com/watch?v=2Kawrd5szHE)

- One class detects line beginnings where pose is line direction:

Input*

*background gradient not part of input, but is because I took a screenshot of a youtube video.

Primary capsule activation



Routing

- No separate activation probability, stored in length of pose vector. Squash pose vector to [0,1]:

$$\vec{s}_j = \sum_i c_{ij} \hat{u}_{ij} \qquad \vec{v}_j = \frac{||\vec{s}_j||^2}{1 + ||\vec{s}_j||^2} \frac{\vec{s}_j}{||\vec{s}_j||}$$

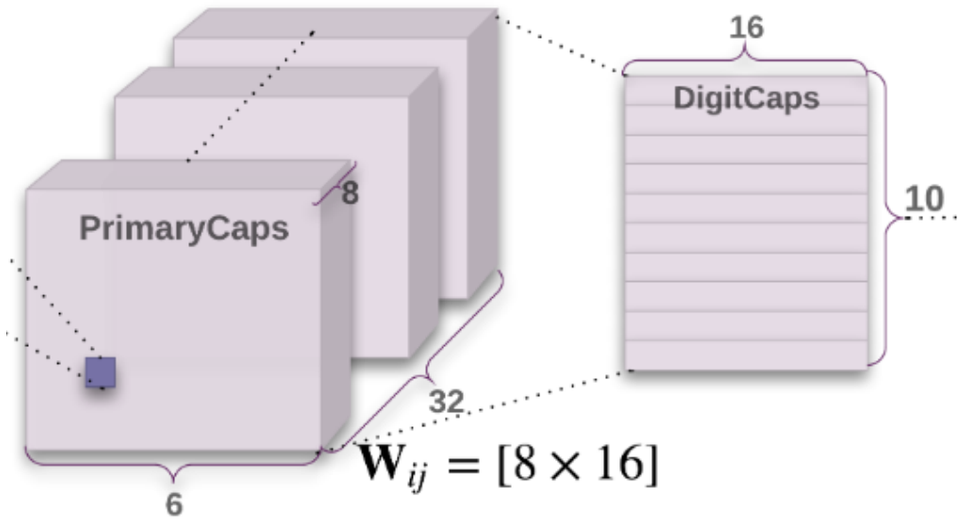
- Assume uniform initial routing priors, calculate $\vec{v}_j$.

$$b_{ij} = 0 \qquad c_{ij} = \text{softmax}(b_{ij})$$

- Update routing coefficients:

$$b_{ij} \leftarrow b_{ij} + \vec{v}_j \cdot \hat{u}_{ij}$$

- Iterate 3 times.



$\vec{u}_i \qquad \hat{u}_{ij} = W_{ij}\vec{u}_i \qquad \vec{v}_j$

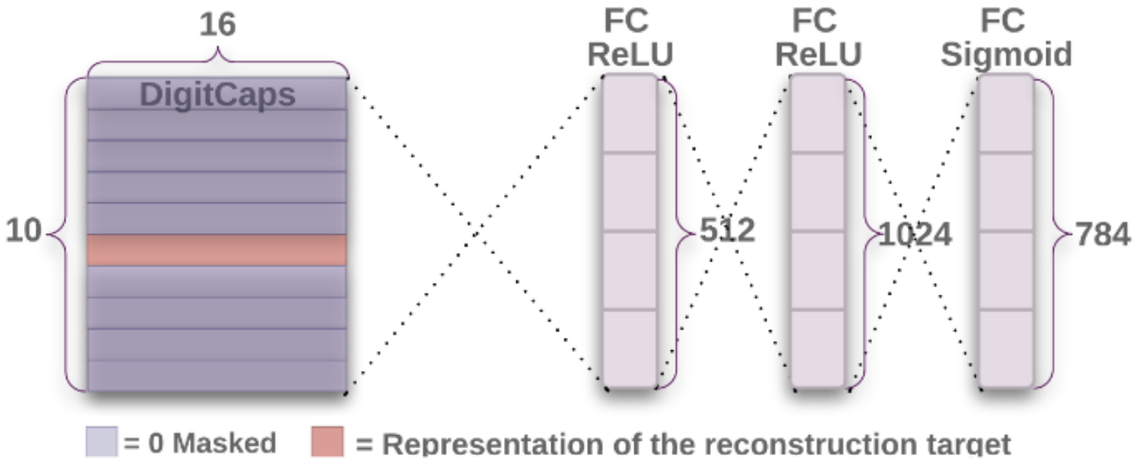
Loss

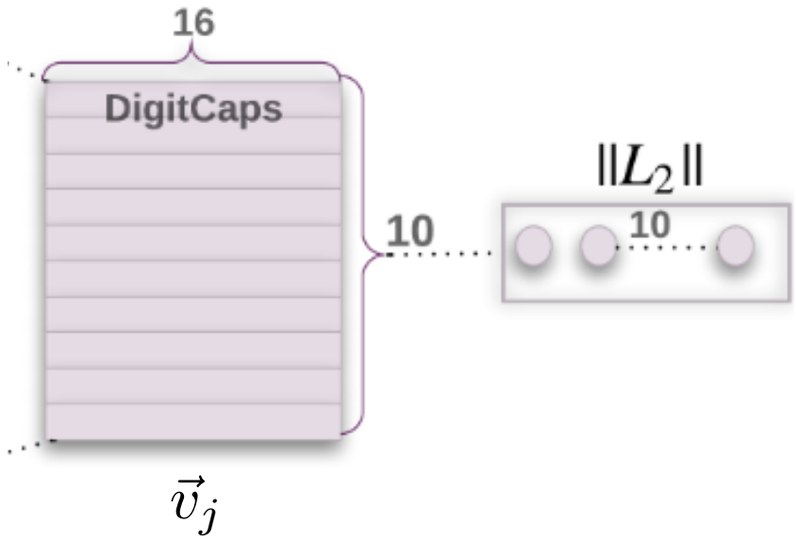
- Two forms of loss. Margin loss:

$$L = \sum_j \left[T_j \max(0, 0.9 - ||\vec{v}_j||)^2 + 0.5(1 - T_j)\max(0, ||\vec{v}_j|| - 0.1)^2 \right]$$

$$T_j = \left\{ \begin{array}{ll} 1 & \text{if digit present} \\ 0 & \text{otherwise} \end{array} \right\}$$

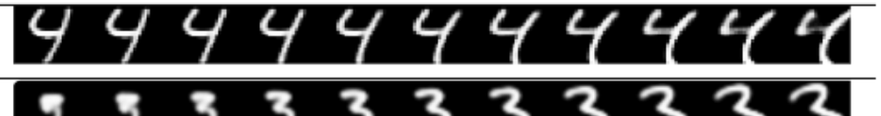
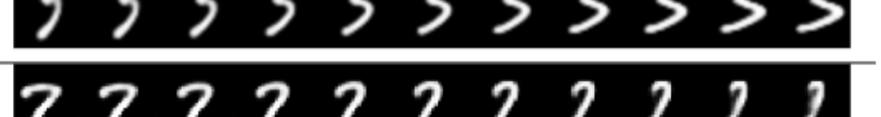
- Reconstruction loss:





Results on MNIST

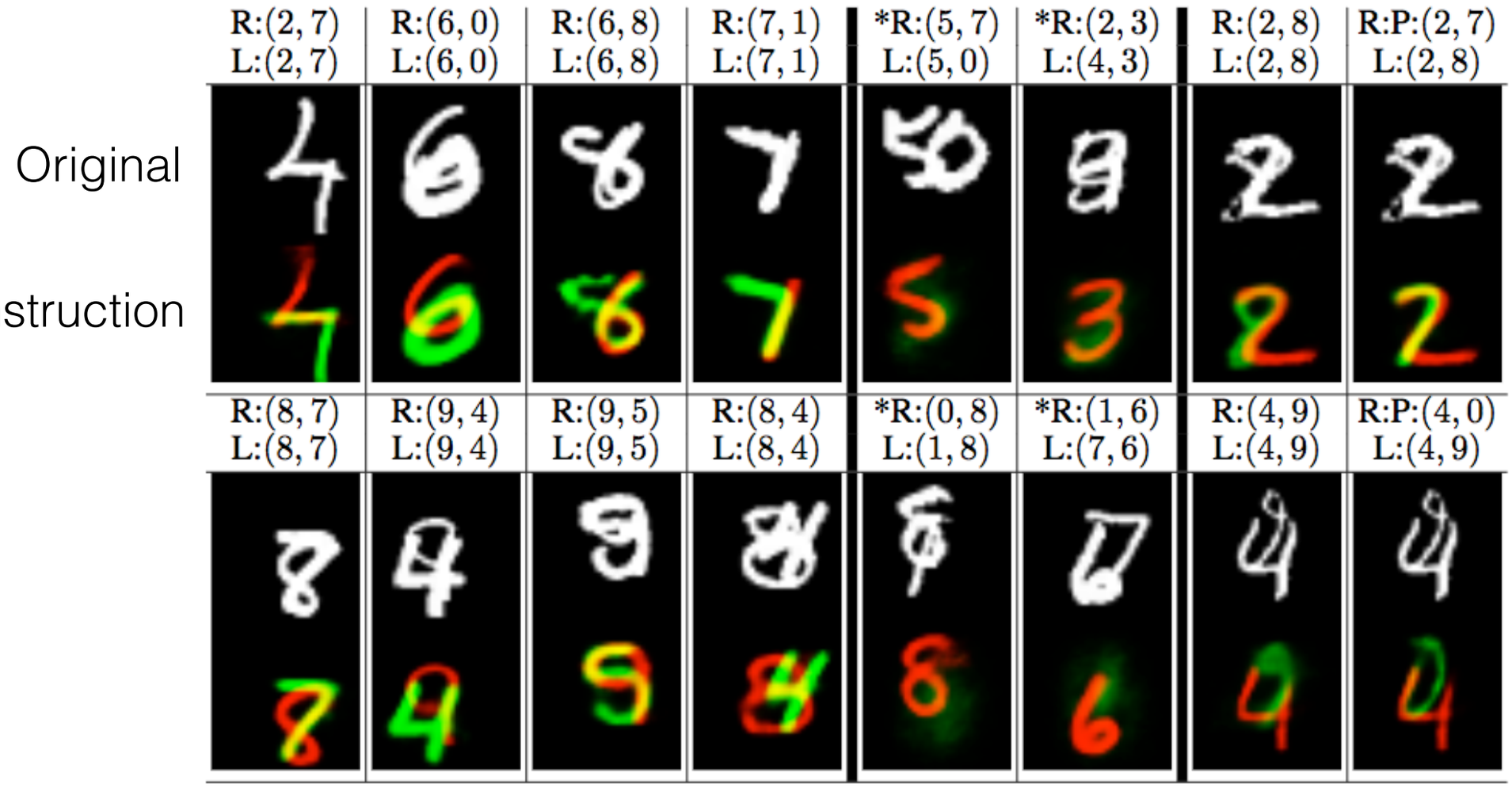
- 0.25% error rate, competitive with CNNs.
- Examples of capsule pose parameters:

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

- On unseen affine transformed digits (affNIST), 79% accuracy vs 66% for CNN.

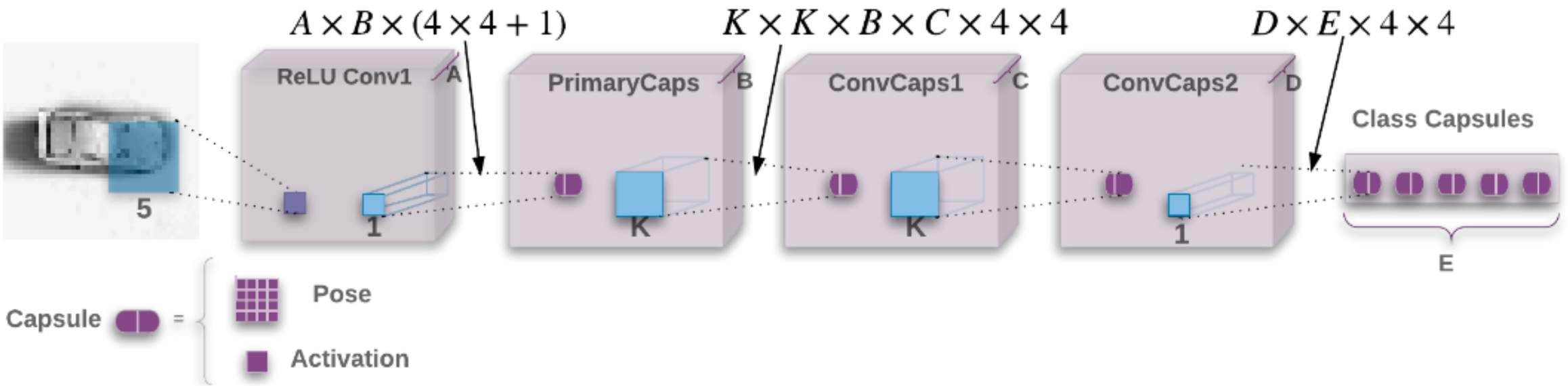
Image Segmentation

- Trained on two MNIST digits with ~80% overlap, classifies pairs with 5.2% error rate, compared to CNN error of 8.1%.



Model 2

- From “Matrix capsules with EM routing,” anonymous authors (openreview.net/pdf?id=HJWLfGWRb)



- Organize pose as 4x4 matrix + activation logit instead of vector. Transformation weights are a 4x4 matrix.

- Primary capsules’ poses are learned linear transform of local features. Activation is sigmoid of learned weighted sum of local features.

- Convolutional capsules share transformation weights and see poses from a local kernel.

EM Routing

- Model higher layer as mixture of Gaussians that explains lower layer's poses.

M

- Start with uniform routing priors c_{ij}, weight by the activations of the lower capsules a_i:

  $r_{ij} = c_{ij} a_i$

- Determine mean and variance:

  $$\mu_{jh} = \frac{\sum_i r_{ij} \hat{u}_{ijh}}{\sum_i r_{ij}} \qquad \sigma^2_{jh} = \frac{\sum_i r_{ij} \left(\hat{u}_{ijh} - \mu_{jh}\right)^2}{\sum_i r_{ij}}$$ per pose component h

- Activate upper capsule as:

  $$a_j = \text{sigmoid}\left[\lambda\left(\beta_a - \sum_h (\beta_v + \log(\sigma_{jh})) \sum_i r_{ij}\right)\right]$$

  β_a, β_v learned by backprop.
  λ fixed schedule.

E

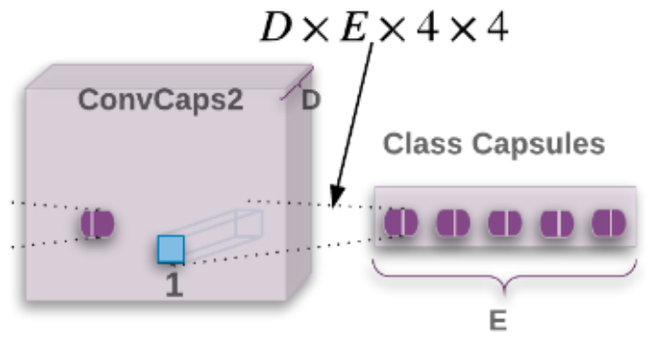
- Calculate new routing coefficients:

  $$p_{ij} = \frac{1}{\sqrt{2\pi \sum_h \sigma^2_{ijh}}} e^{\frac{-\sum_h (\hat{u}_{ijh} - \mu_{jh})^2}{2\sigma^2_{ijh}}} \qquad c_{ij} = \frac{a_j p_{ij}}{\sum_j a_j p_{ij}}$$

- Iterate 3 times.

Last Layer and Loss

- Connection to class capsules uses coordinate addition scheme:
  - Weights shared across locations, like convolutional layer.
  - Explicit (x,y) offset of kernel added to first two elements of pose passed to class capsules.



- Spread loss:

$$L = \sum_{j \neq t} (\max(0, m - (a_t - a_j))^2$$ for target class t

- Margin m increases linearly from 0.2 to 0.9 during training.

Test Dataset

- smallNORB dataset: 96x96 greyscale images of 5 classes of toy (airplanes, cars, trucks, humans, animals) with 10 physical instances of each toy, 18 azimuthal angles, 9 elevation angles, and 6 lighting conditions per training and test set. Total of 48,600 images each.

Results

- Downscale smallNORB to 48x48, randomly crop to 32x32.

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[2]	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**%

[2] Loss from model 1

Novel Viewpoints

- Case 1: train on middle 1/3 azimuthal angles, test on remaining 2/3 azimuthal angles.

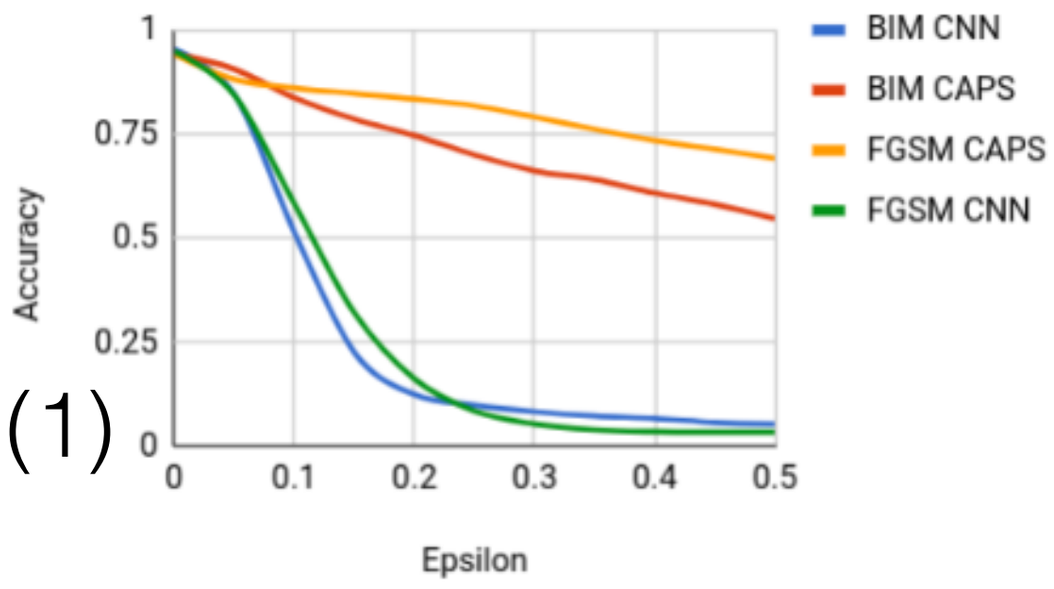
- Case 2: train on lower 1/3 elevation angles, test on higher 2/3 elevation angles.

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%

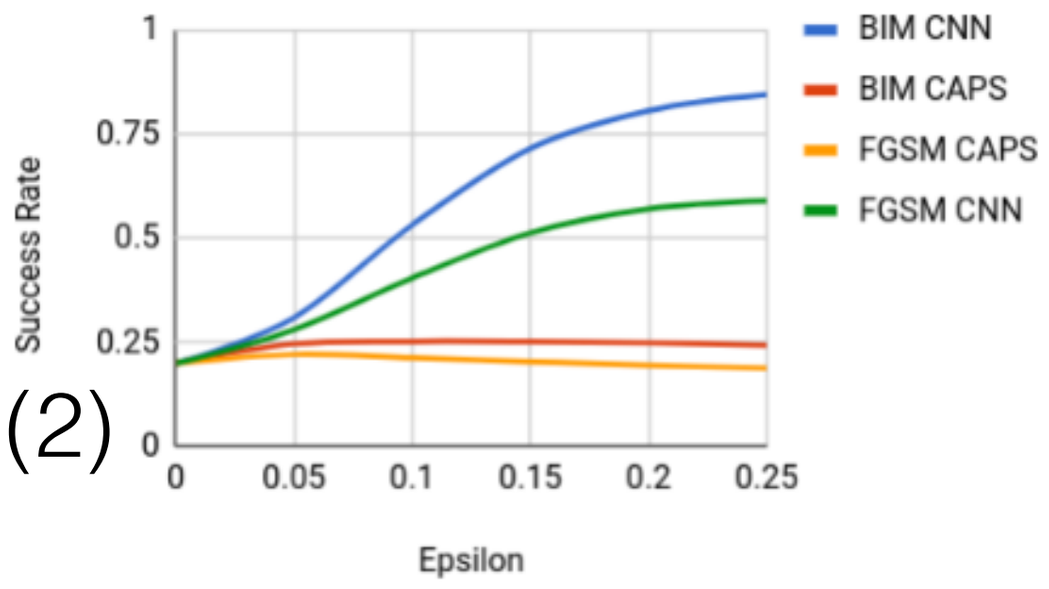
Adversarial Robustness

- FGSM adversarial attack: compute gradient of output w.r.t. change in pixel intensity, then modify each pixel by small ε in direction that either (1) maximizes loss, or (2) maximizes classification probability of wrong class.
- BIM adversarial attack: same thing but with several steps.

(1)



(2)



- No improvement on images generated by adversarial CNN.

Downsides

- Capsule networks are *really* slow. Shallow EM routed network take 2 days to train on laptop, comparable CNN takes 30 minutes.

- Poor performance (~11% error) on CIFAR10; generally bad at complex images.

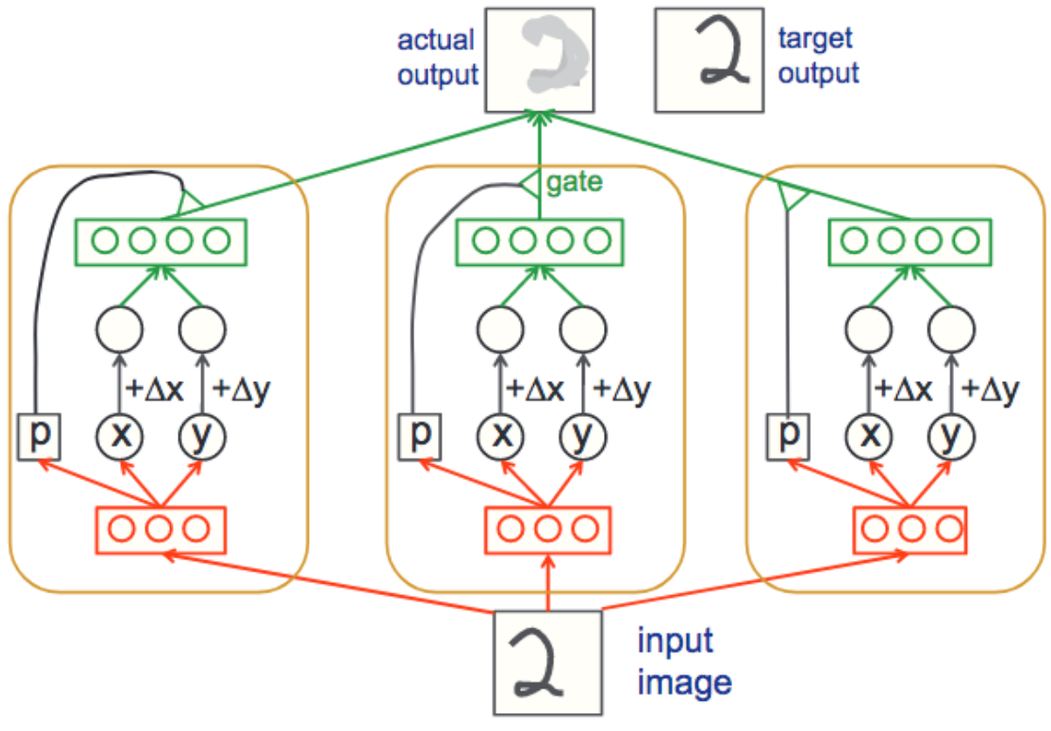
- Can’t handle multiple copies of the same object (crowding).

Conclusions

- Capsule networks explicitly learn the relative poses of objects.
- State-of-the-art performance on image segmentation and 3D object recognition
- Poor performance on complicated images, also very slow.
- Little studied… unknown if these issues can be improved upon.

Transforming Auto-encoders

- With unlabeled data and ability to explicitly transform poses, can learn capsules via auto-encoder:



- Then connect capsules to factor analyzers, can get competitive error rate on MNIST with ~25 labelled examples.