Dynamic Routing Between Capsules

Changzhi Sun

Dynamic Routing Between Capsules

Sara Sabour **Nicholas Frosst**

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com

Outline

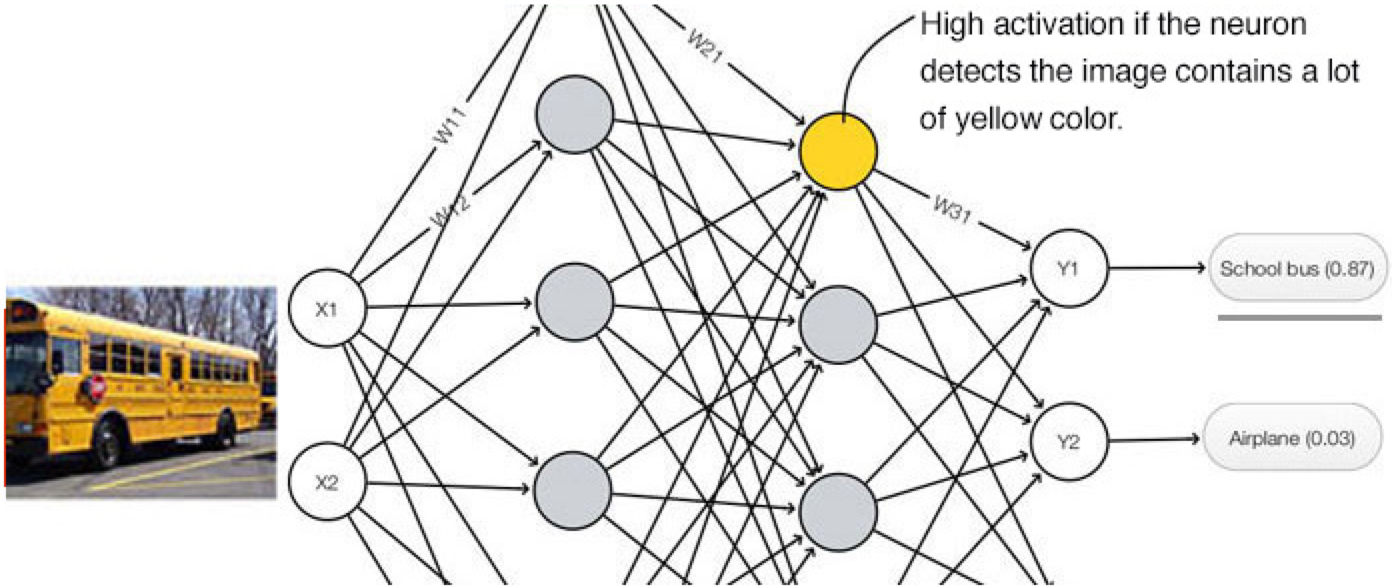
1.Background

2.Capsules

3.Experments

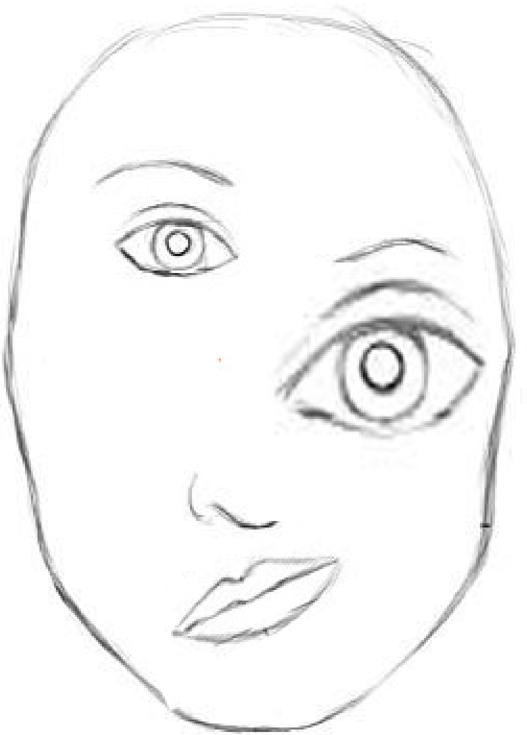
4.Conclusions

CNNs



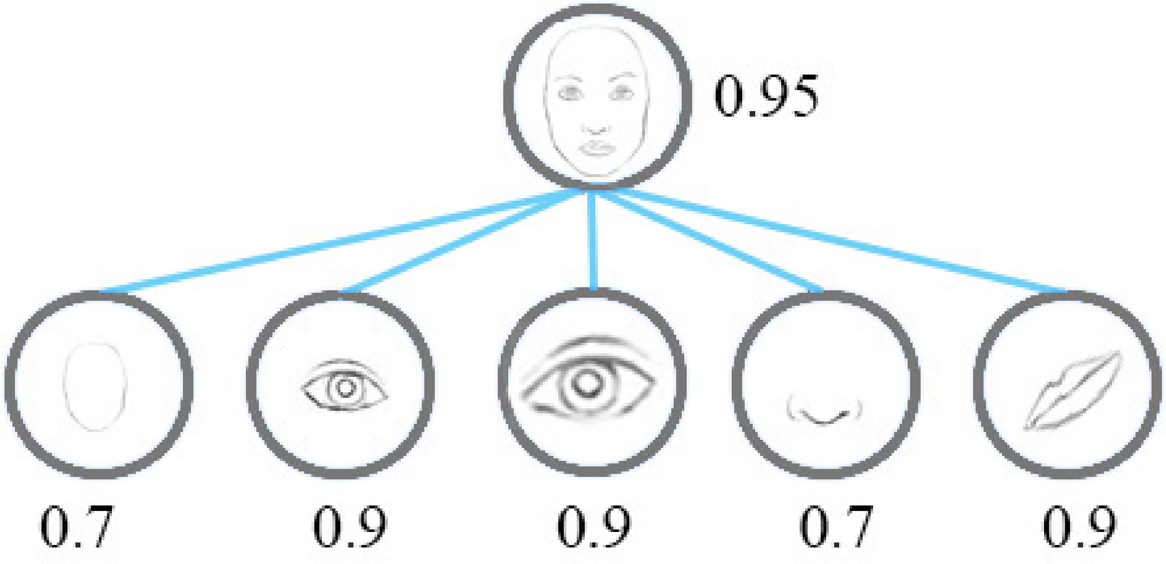
- **good**: detecting features
- **bad**: spatial relationships among features (size, orientation)

CNNs

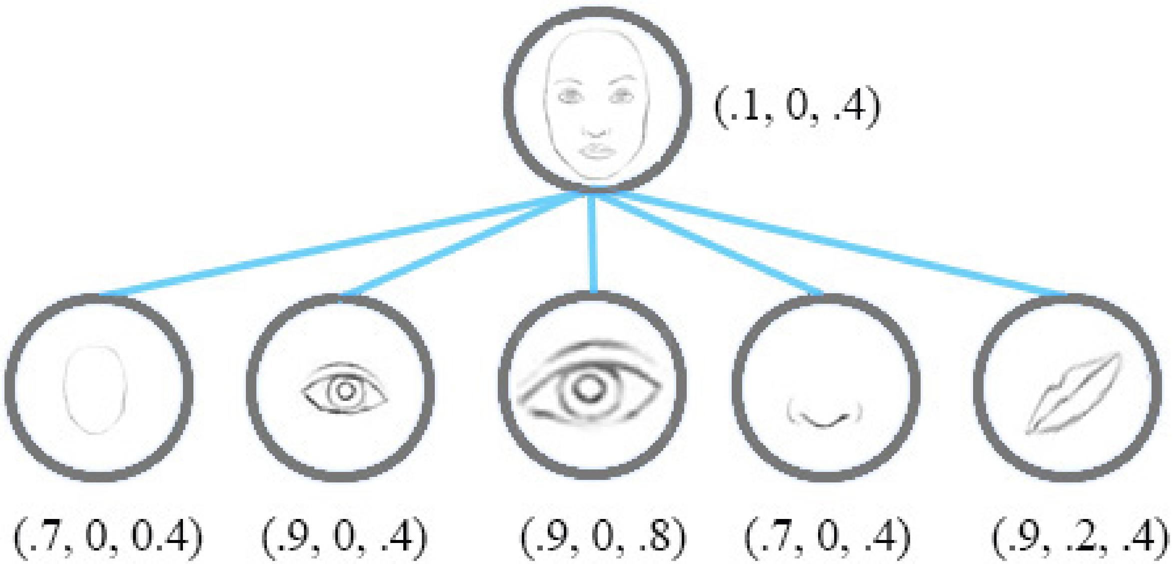


- may fool a simple CNN model
- good sketch of human face

Neurons

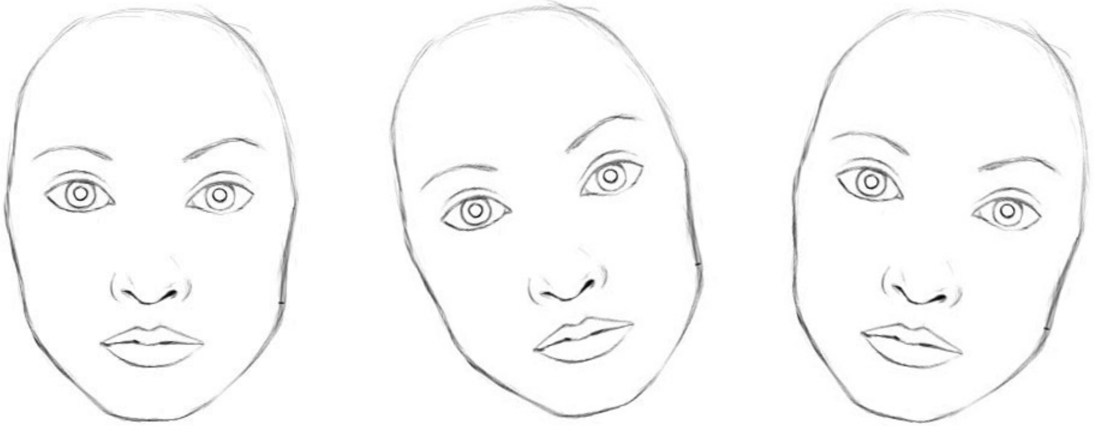


Capsules



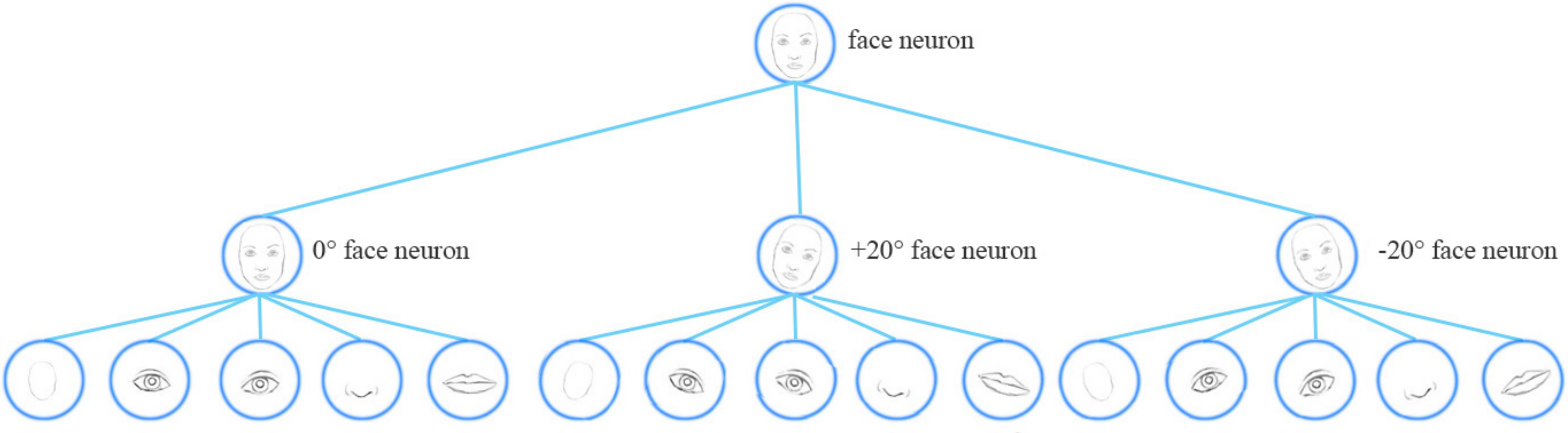
- [`likelihood` , `orientation` , `size`]
- detect consistence in the `orientation` and `size`
- capsules output a **vector** instead of a single scaler value

Invariant



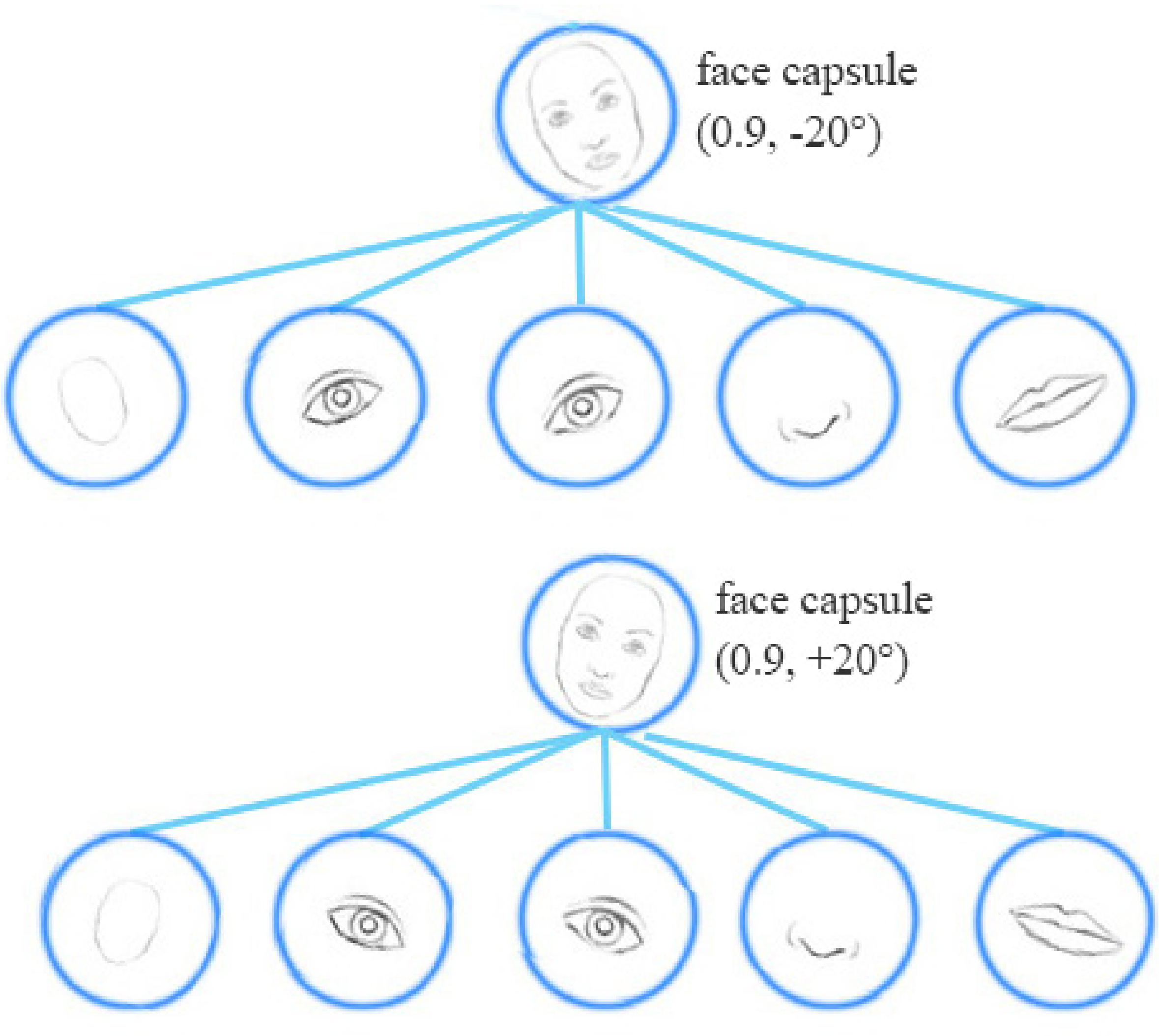
- how to train a face detection neuron for different orientations ?

Invariant



- add more conv layers and features
- memorize rather than generalize
- large training data

Equivariance



Equivariance vs Invariance

- **Invariance** is the detection of features regardless of the variants
- **Equivariance** is the detection of objects that can transform to each other (for example, detecting faces with different orientations)

Capsules

- A **capsule** is a group of neurons that not only capture the likelihood but also the parameters of the specific feature
- We call the output vector of a capsule as the **activity vector** with magnitude represents the probability of detecting a feature and its orientation represents its parameters (properties)

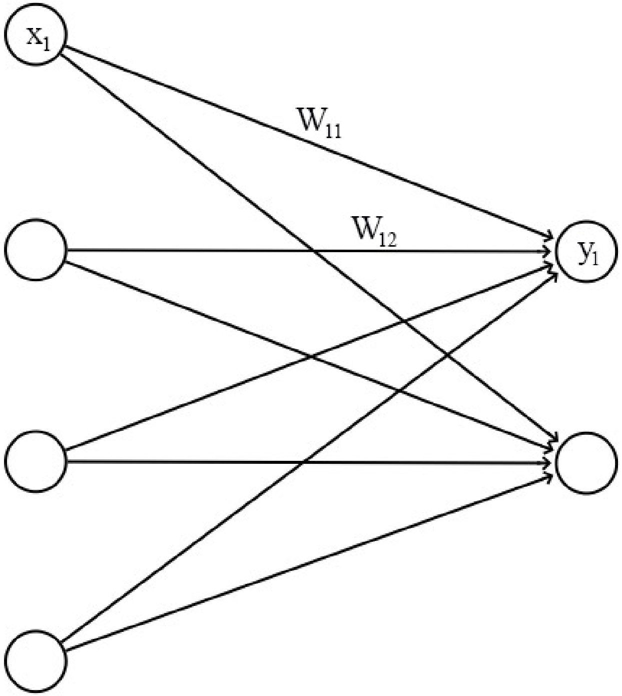
Capsules

Single neuron

Single capsule with
2 neurons

(Number rotated by 20°)

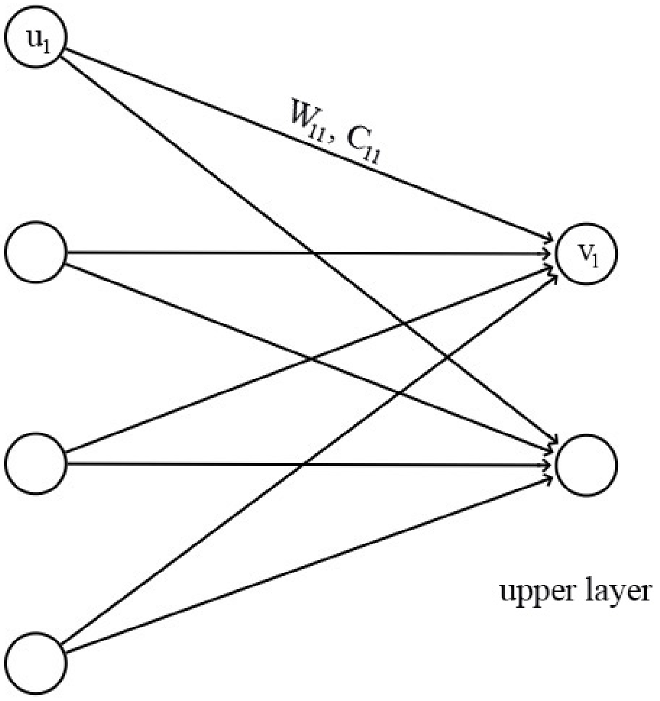
Fully Connected NN



$$z_j = \sum_i W_{ij} x_i$$

$$y_j = ReLU(z_j)$$

Compute the Output of a Capsule



- the input $\mathbf{u}_i$ and the output $\mathbf{v}_i$ are **vectors**

Compute the Output of a Capsule

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

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$$

$$\sum_j c_{ij} = 1$$

- c_{ij} are **coupling coefficients**, trained by the iterative dynamic routing process
- **squashing function**

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

Routing-by-agreement

Procedure 1 Routing algorithm.

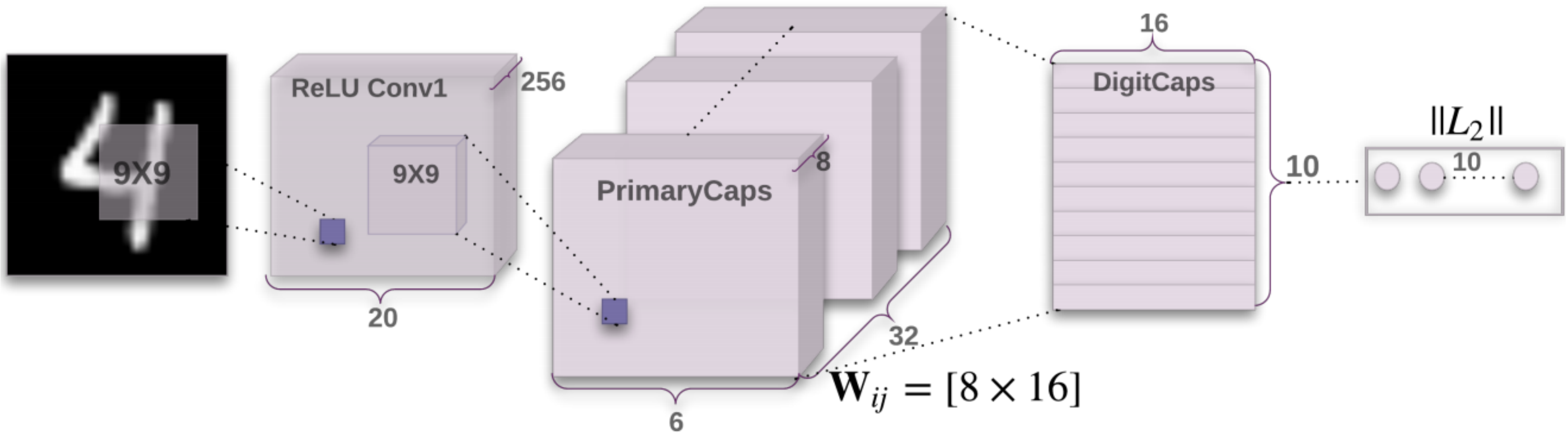
```
procedure ROUTING(û_{j|i}, r, l)
    for all capsule i in layer l and capsule j in layer (l + 1): b_{ij} ← 0.
    for r iterations do
        for all capsule i in layer l: c_i ← softmax(b_i)
        for all capsule j in layer (l + 1): s_j ← Σ_i c_{ij} û_{j|i}
        for all capsule j in layer (l + 1): v_j ← squash(s_j)
        for all capsule i in layer l and capsule j in layer (l + 1): b_{ij} ← b_{ij} + û_{j|i}.v_j
    return v_j
```

Loss Function (Margin Loss)

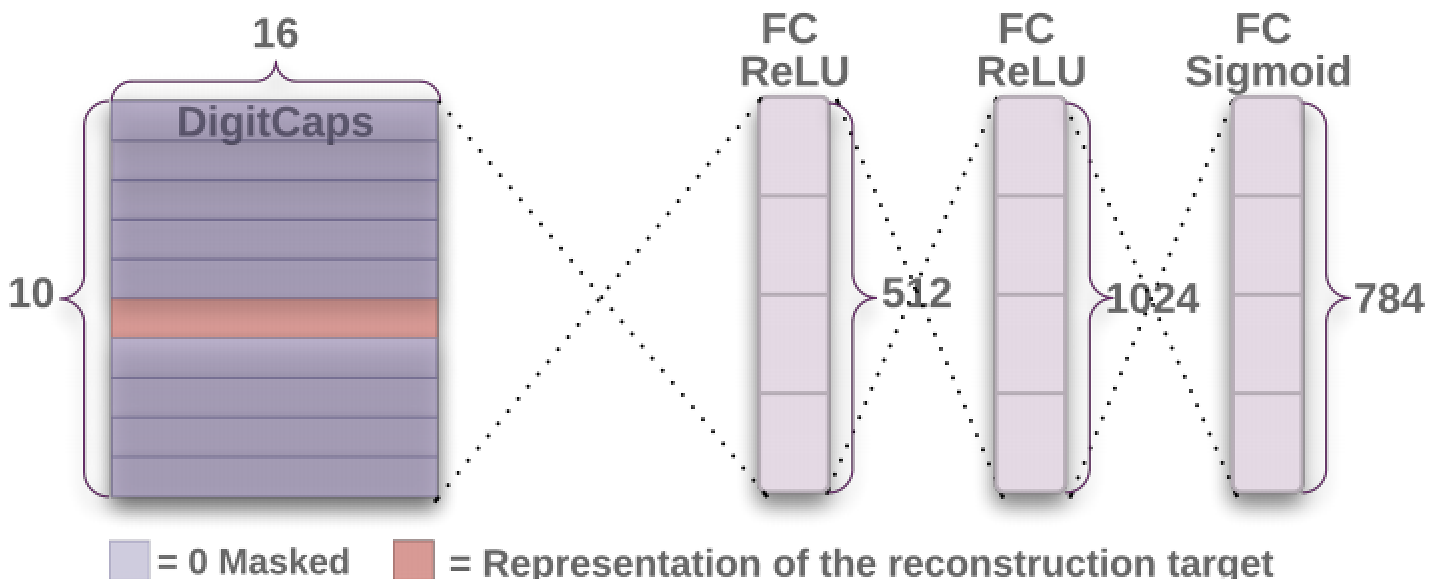
$$L_c = T_c \max(0, m^+ - \|v_c\|)^2 + \lambda(1 - T_c) \max(0, \|v_c\| - m^-)^2$$

- T_c = 1 if an object of class c is present
- $m^+ = 0.9$ and $m^- = 0.1$, $\lambda = 0.5$
- the total loss is just sum of the losses of all classes

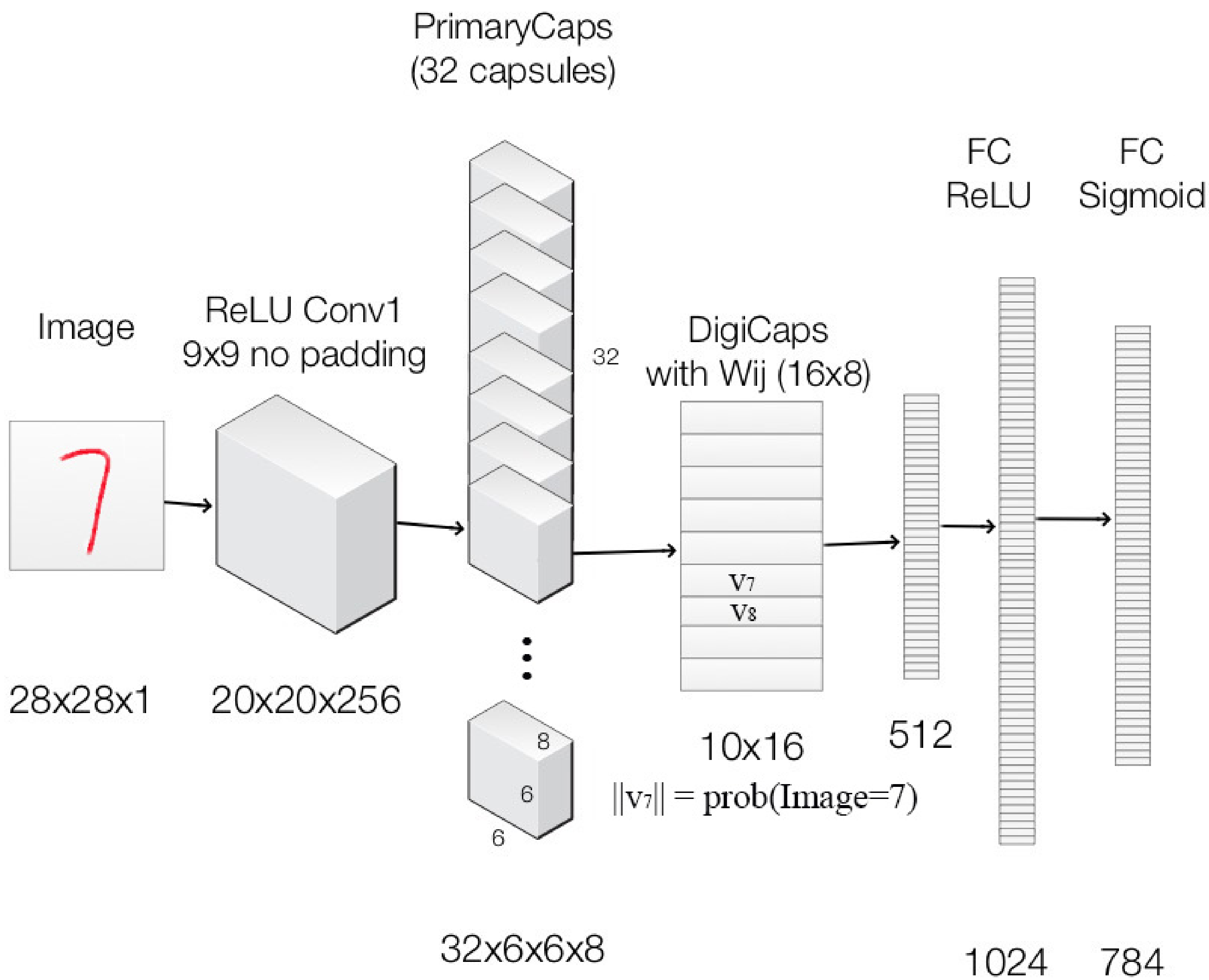
Experiments



- CapsNet for MNIST



- Reconstruction Regularization

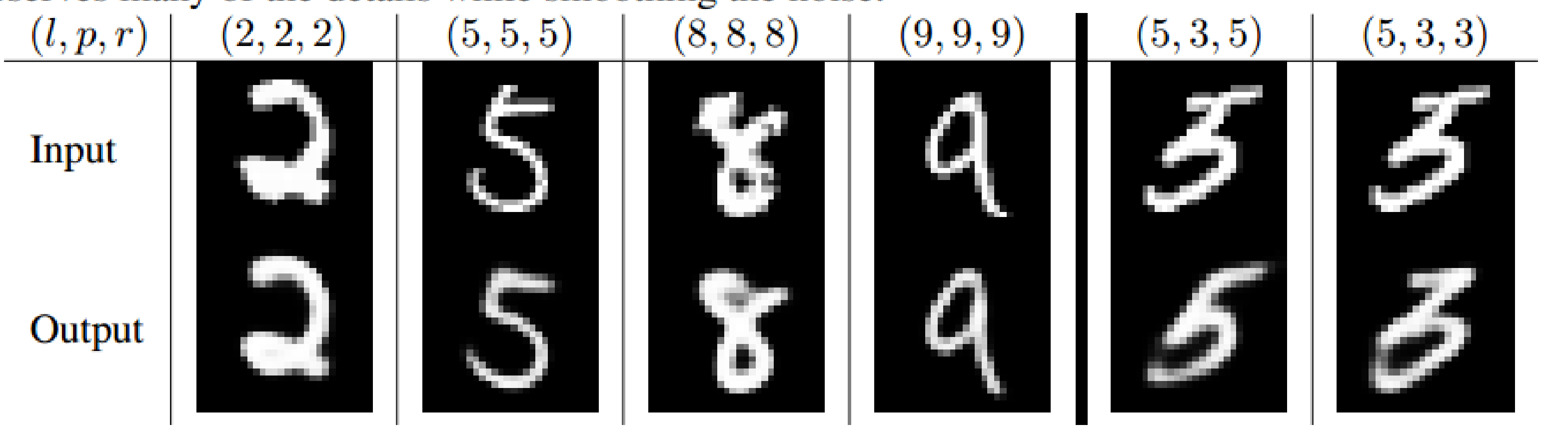

PrimaryCaps
(32 capsules)
FC
ReLU
FC
Sigmoid
Image
ReLU Conv1
9x9 no padding
32
DigiCaps
with Wij (16x8)
v7
v8
28x28x1
20x20x256
8
6
6
10x16
512
||v7|| = prob(Image=7)
32x6x6x8
1024
784

Layer Name	Apply	Output shape
Image	Raw image array	28x28x1
ReLU Conv1	Convolution layer with 9x9 kernels output 256 channels, stride 1, no padding with ReLU	20x20x256
PrimaryCapsules	Convolution capsule layer with 9x9 kernel output 32x6x6 8-D capsule, stride 2, no padding	6x6x32x8
DigiCaps	Capsule output computed from a W_{ij} (16x8 matrix) between u_i and v_j (i from 1 to 32x6x6 and j from 1 to 10).	10x16
FC1	Fully connected with ReLU	512
FC2	Fully connected with ReLU	1024
Output image	Fully connected with sigmoid	784 (28x28)

Table 1: CapsNet classification test accuracy. The MNIST average and standard deviation results are reported from 3 trials.

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	$\mathbf{0.25}_{\pm 0.005}$	$\mathbf{5.2}$

Figure 3: Sample MNIST test reconstructions of a CapsNet with 3 routing iterations. (l, p, r) represents the label, the prediction and the reconstruction target respectively. The two rightmost columns show two reconstructions of a failure example and it explains how the model confuses a 5 and a 3 in this image. The other columns are from correct classifications and shows that model preserves many of the details while smoothing the noise.



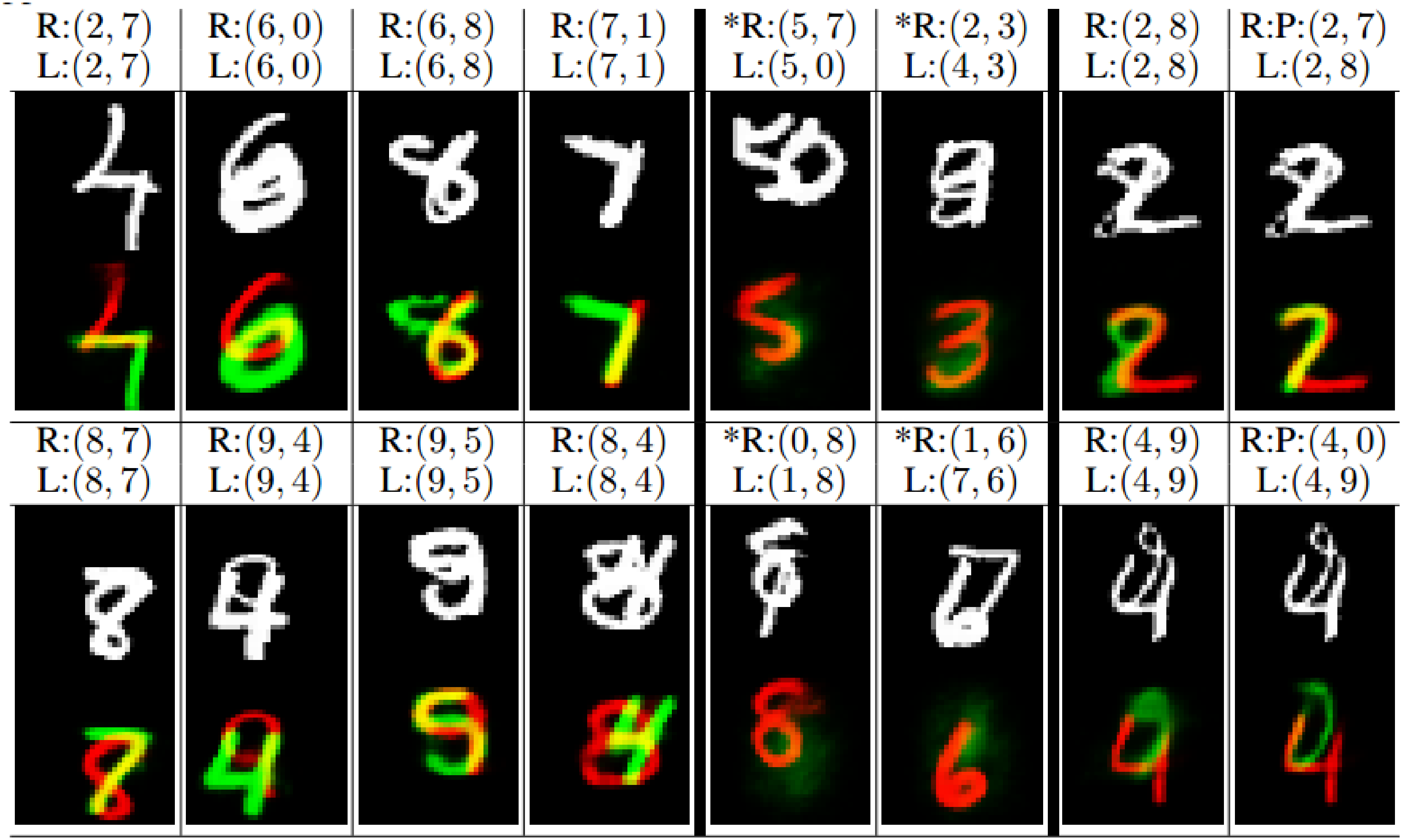
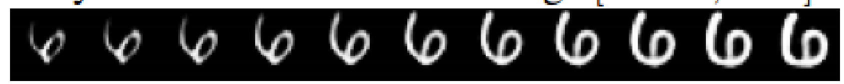
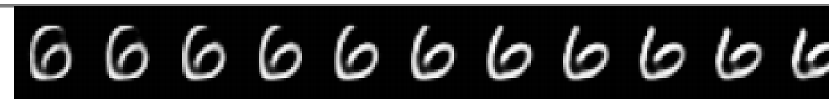

R:(2, 7)
L:(2, 7)
R:(6, 0)
L:(6, 0)
R:(6, 8)
L:(6, 8)
R:(7, 1)
L:(7, 1)
*R:(5, 7)
L:(5, 0)
*R:(2, 3)
L:(4, 3)
R:(2, 8)
L:(2, 8)
R:P:(2, 7)
L:(2, 8)
R:(8, 7)
L:(8, 7)
R:(9, 4)
L:(9, 4)
R:(9, 5)
L:(9, 5)
R:(8, 4)
L:(8, 4)
*R:(0, 8)
L:(1, 8)
*R:(1, 6)
L:(7, 6)
R:(4, 9)
L:(4, 9)
R:P:(4, 0)
L:(4, 9)

Figure 4: Dimension perturbations. Each row shows the reconstruction when one of the 16 dimensions in the DigitCaps representation is tweaked by intervals of 0.05 in the range $[-0.25, 0.25]$.

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

Conclusions

- scalar to vector
- routing-by-agreement
- slower than CNN
- not yet proven its effectiveness in large-scale data

Thanks Q&A

https://jhui.github.io/2017/11/03/Dynamic-Routing-Between-Capsules/