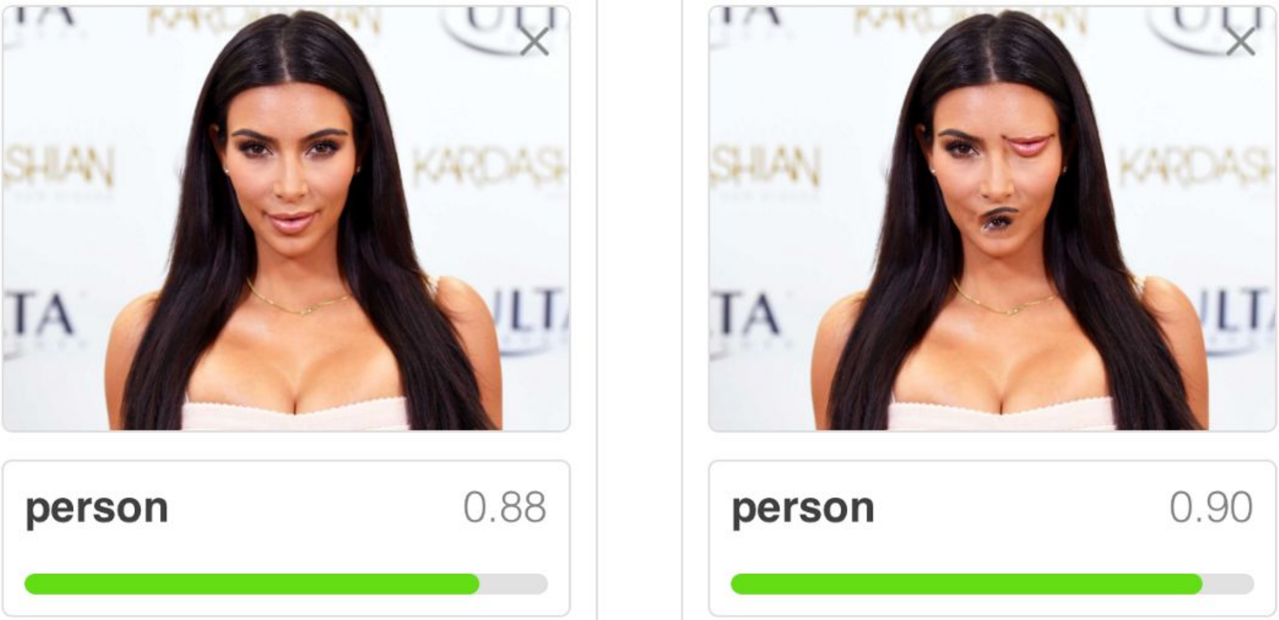
Comp541 Final Project
Gürsu Gülcü

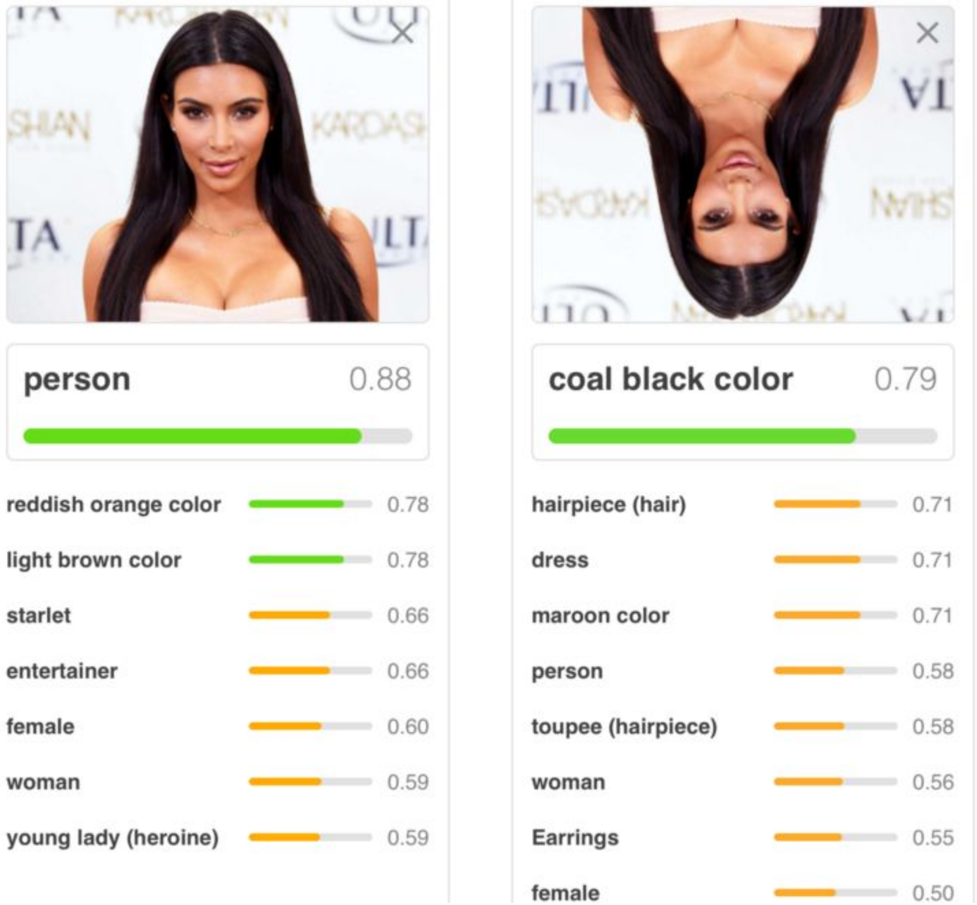
“Dynamic Routing Between Capsules”
by Sabour, Frosst, Hinton

Problem 1: A CNN may identify both as a “person”. Why?



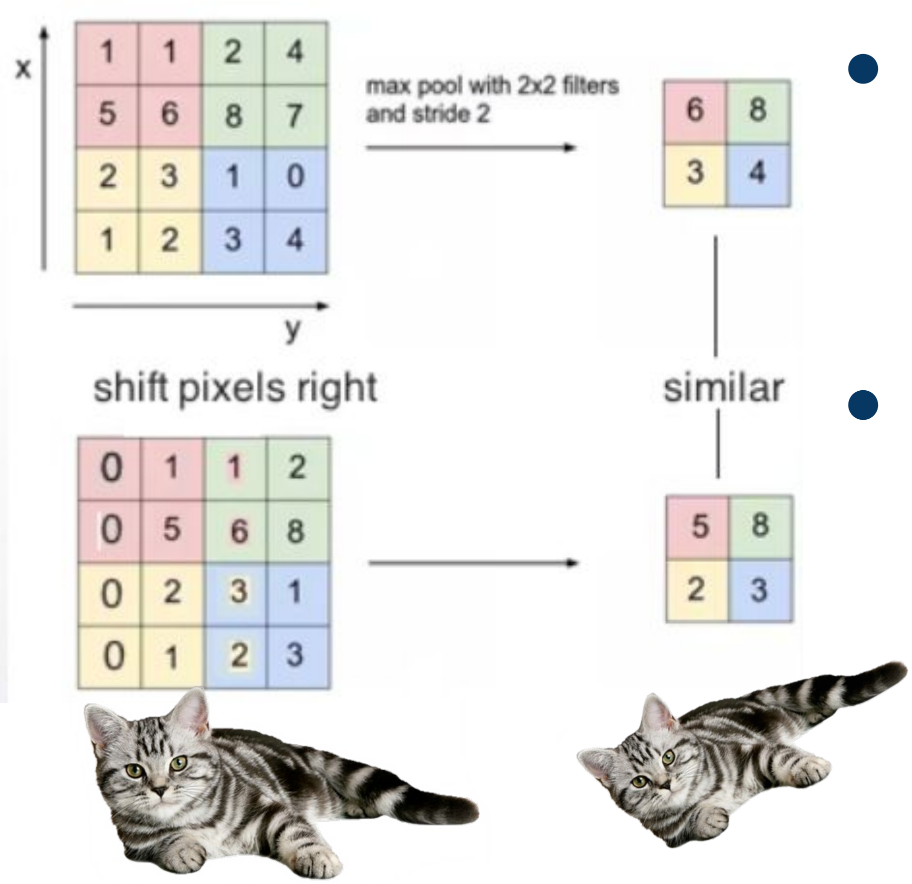
- It merely checks for the existence of some features it has learnt, not for the relationships among the features.

Problem 2: A CNN fails to identify the inverted image.



- Feature detection process is not as robust to transformations as us humans.
- Maybe our algorithms should try to mimic the human brain?

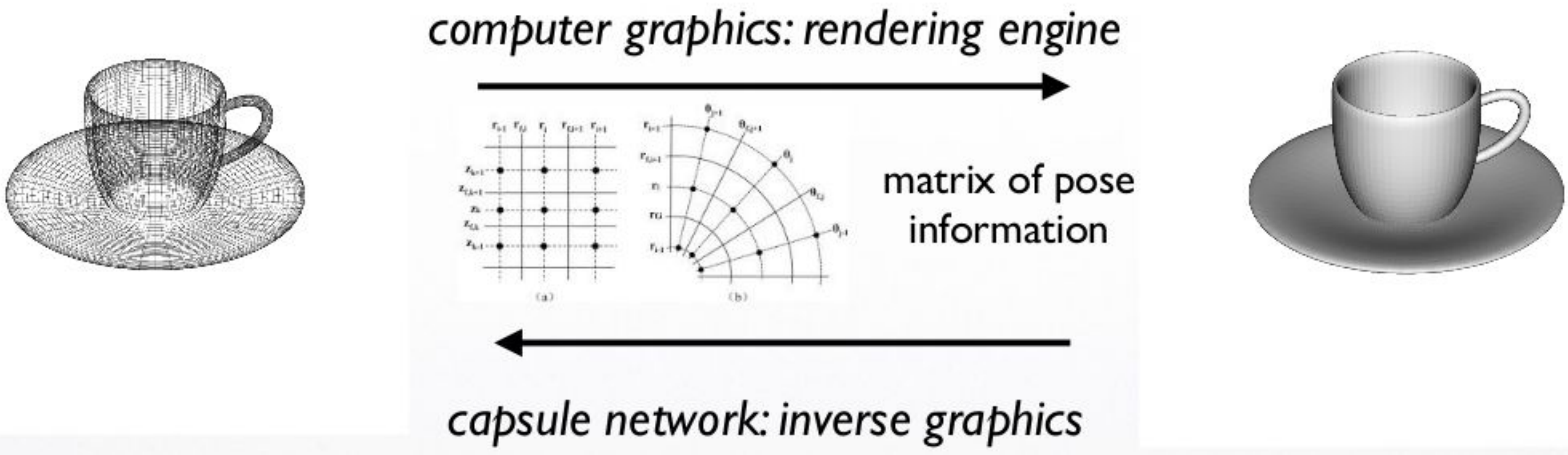
Problem 3: Pooling operation may be discarding useful information.



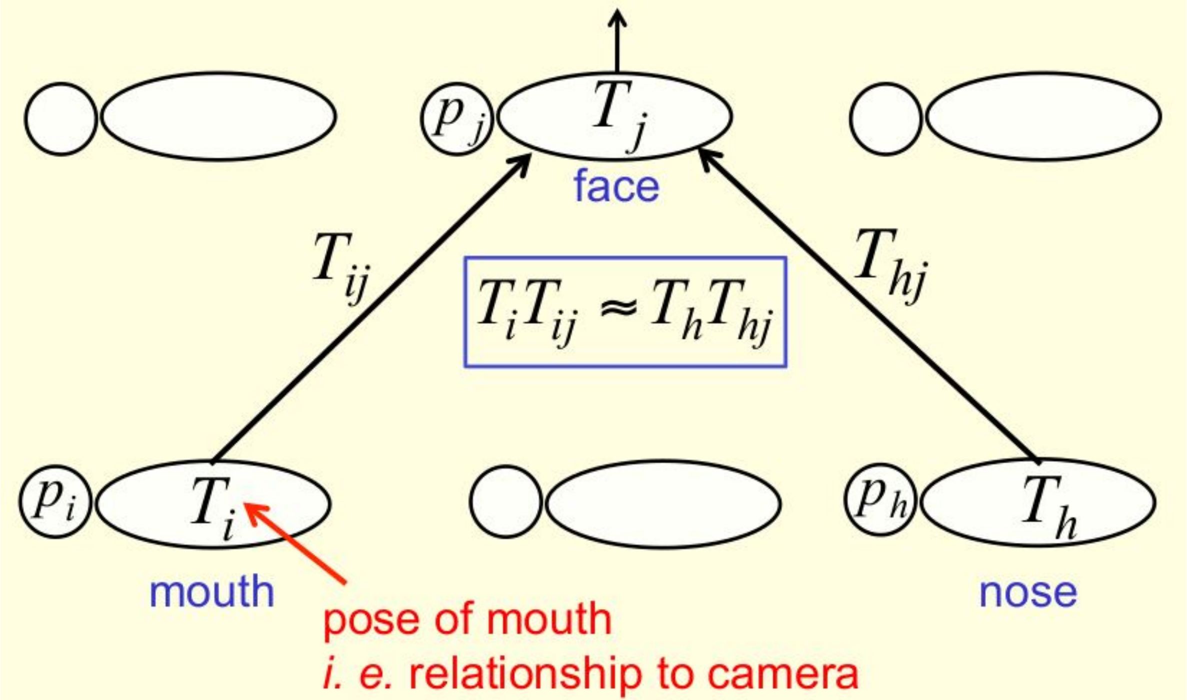
- Sub-sampling due to pooling obtains crude spatial **invariance** by ignoring details -> Neural activities constant.
- Better to aim for **equivariance**: Invariance under transformations only.
  - Changes in viewpoint should change neural activities
  - Constant weights should code the viewpoint-invariant knowledge

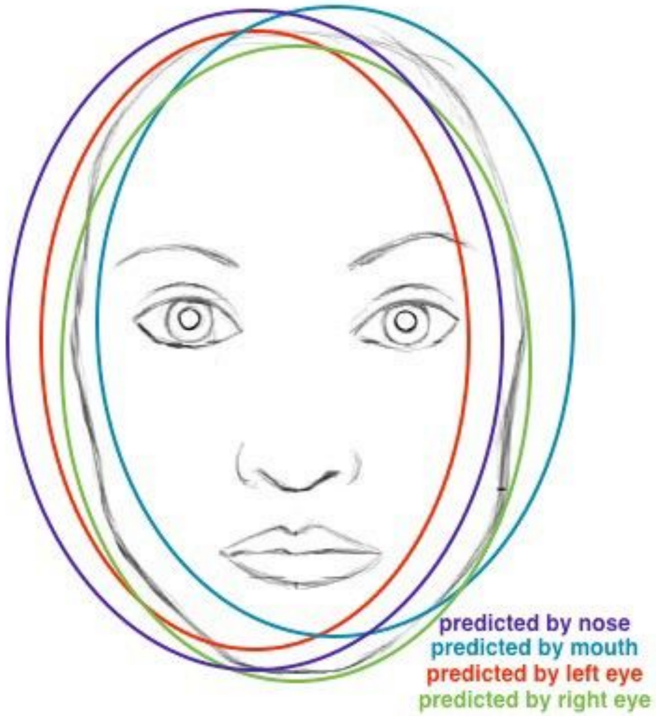
A Possible Solution: Inverse Computer Graphics

- Representation of objects in the brain does not depend on view angle
- Relationships between 3D objects can be represented by **pose**
  - **Translation** and **rotation**



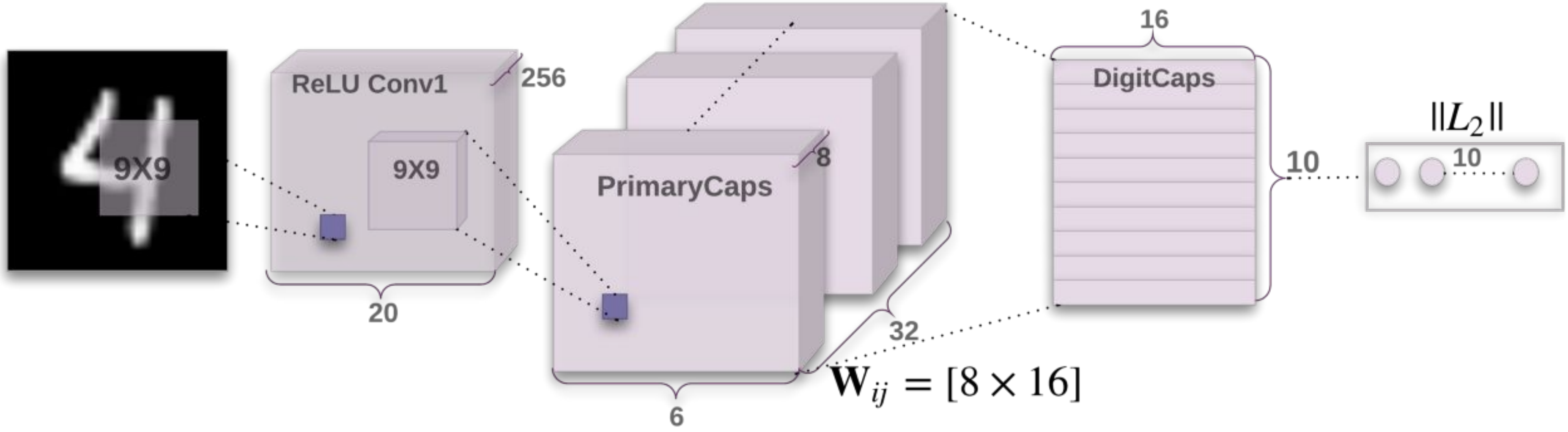
A higher level entity is present if lower level visual entities can agree on their predictions for its pose





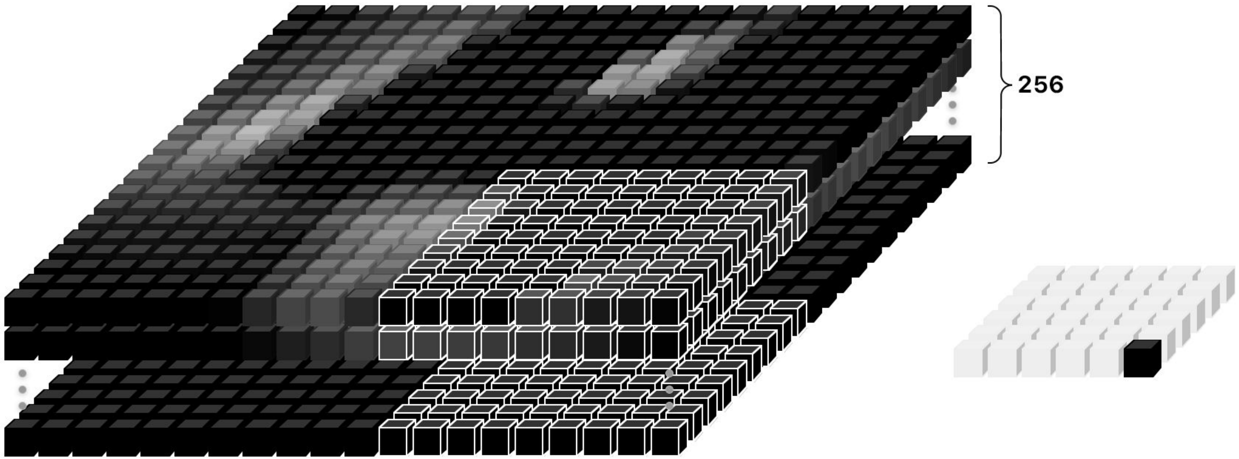
Predictions for face location from nose, mouth and eyes match -> there must be a face ***there***

Capsule Networks: A Solution to Our Problems?



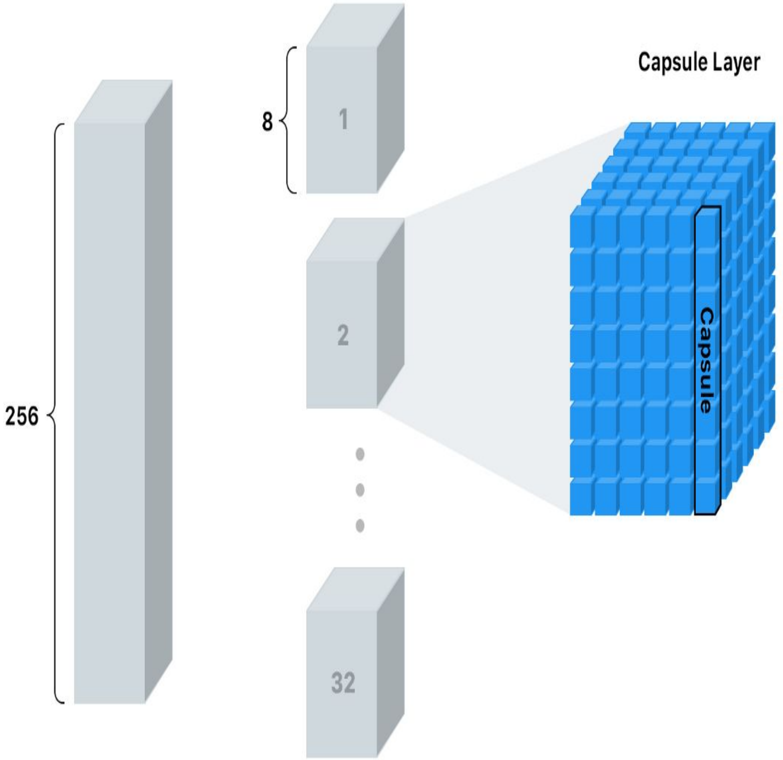
- **Input** is the familiar 28x28x1 MNIST images.
- **First layer** is convolutional with 256 9x9 kernels and stride=1
    - No pooling!
    - Output is 20x20x256 since (28-9+1)/1 = 20

Second Convolutional Layer -> Primary Capsules



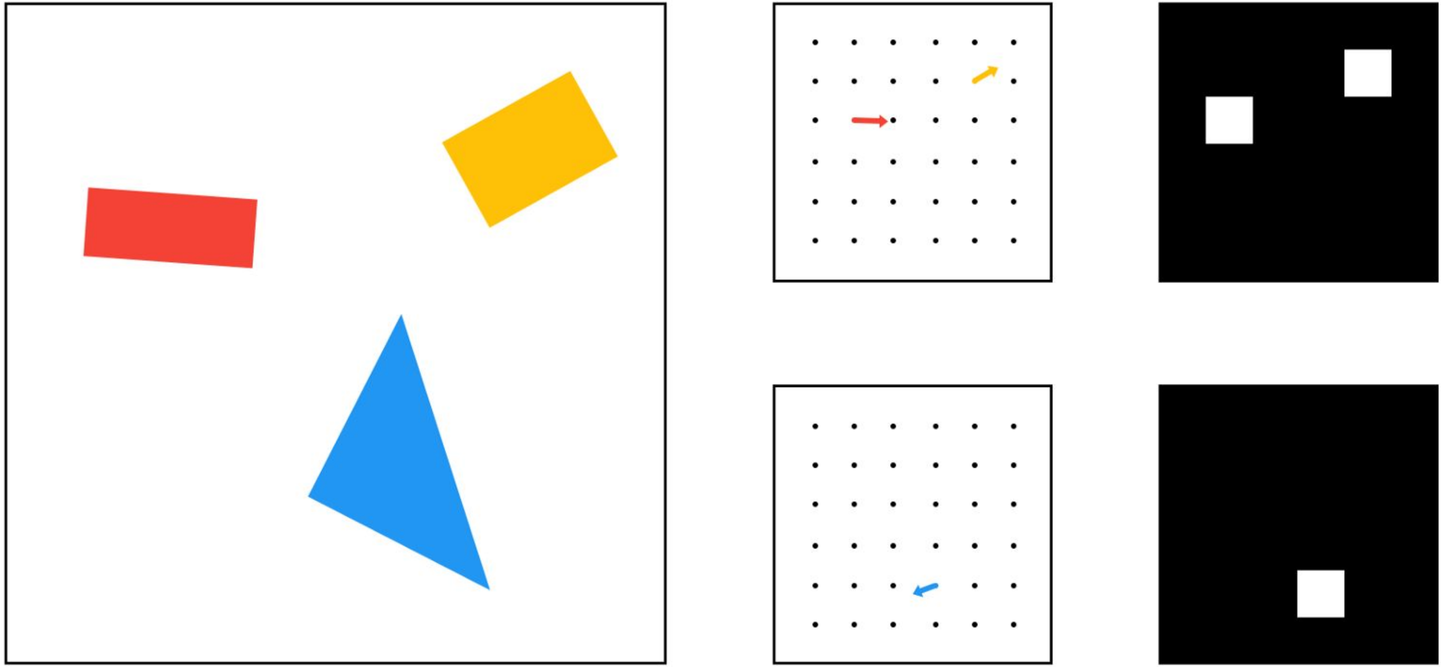
- **Second layer** is again convolutional with 256 9x9 kernels , stride=2
  - Output is 6x6x256 since (20-9+1)/2 = 6

Dissecting the Layer of Primary Capsules



- Group 256 channels into 32
    - Depth is 8
    - 6x6=36 localized "capsules", 32 groups, each having a depth of 8
- Interpretation:
    - Z-axis (channels): Capsule contents
        - 32 features holding 8 numbers
        - Thickness, orientation etc.
    - X,Y axis (6x6 grid): Spatial info
        - Looking at a specific part of the image

What are the Primary Capsules looking at?



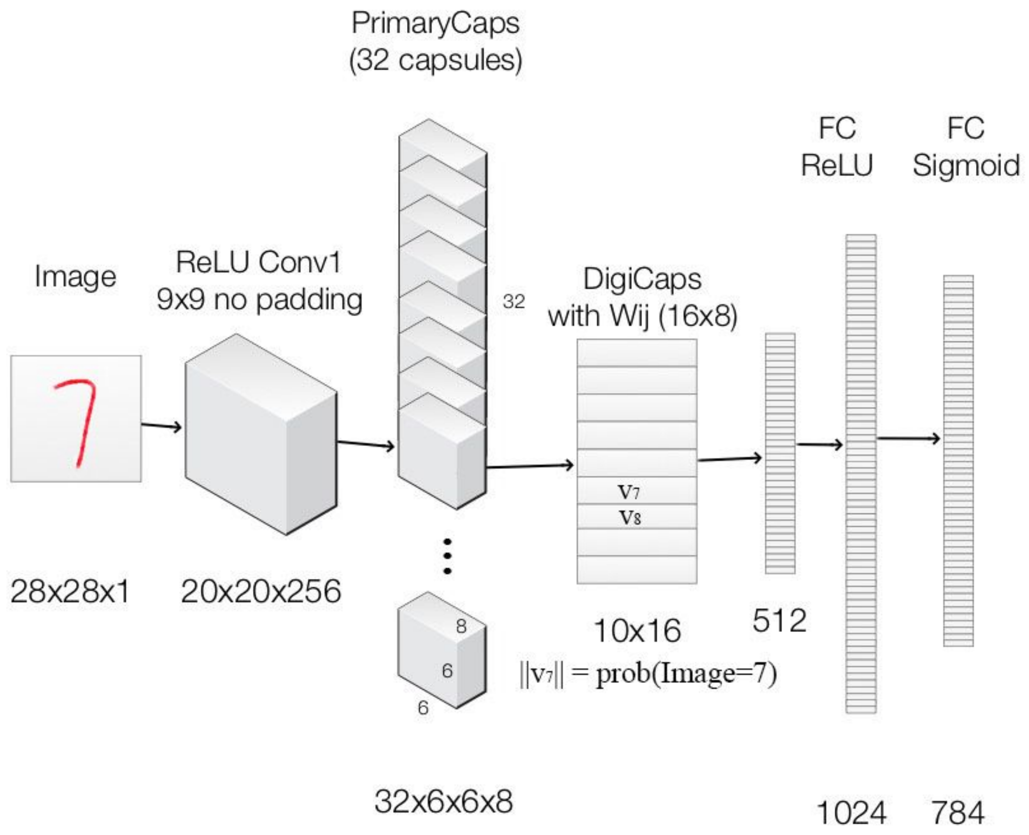
- “Squash” the output vectors before passing to the next layer

The Squashing Function

$$\mathbf{v}_j = \underbrace{\boxed{\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}}}_{\text{additional “squashing”}} \underbrace{\boxed{\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}}}_{\text{unit scaling}}$$

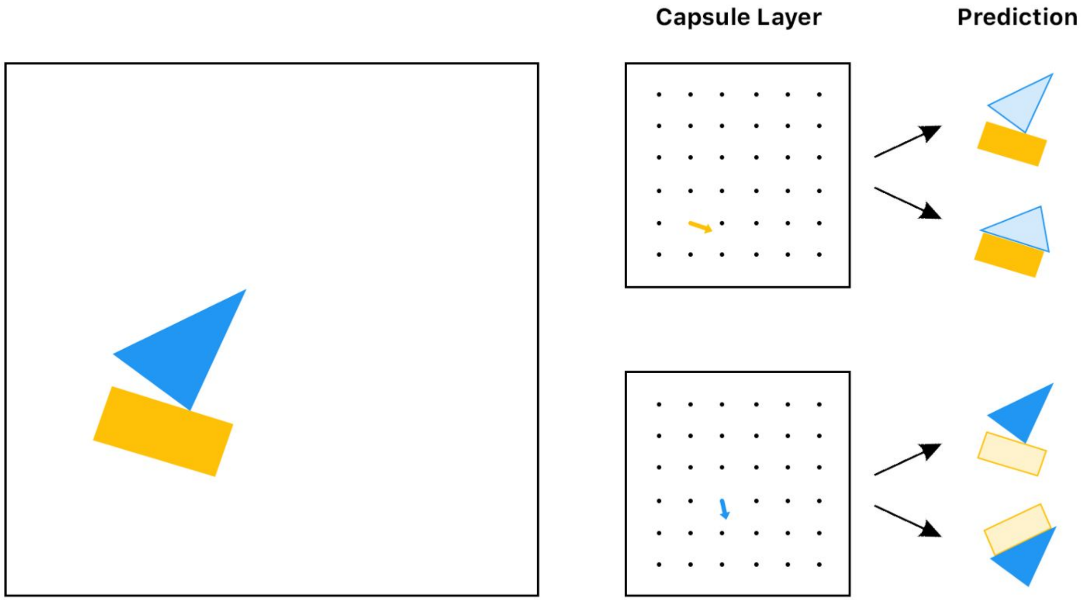
- Normalizing operation:
  - Unit vector for a long vector
  - Scale with its own small magnitude for further shrinkage for a short one

Next Capsule Layer: Digit Caps



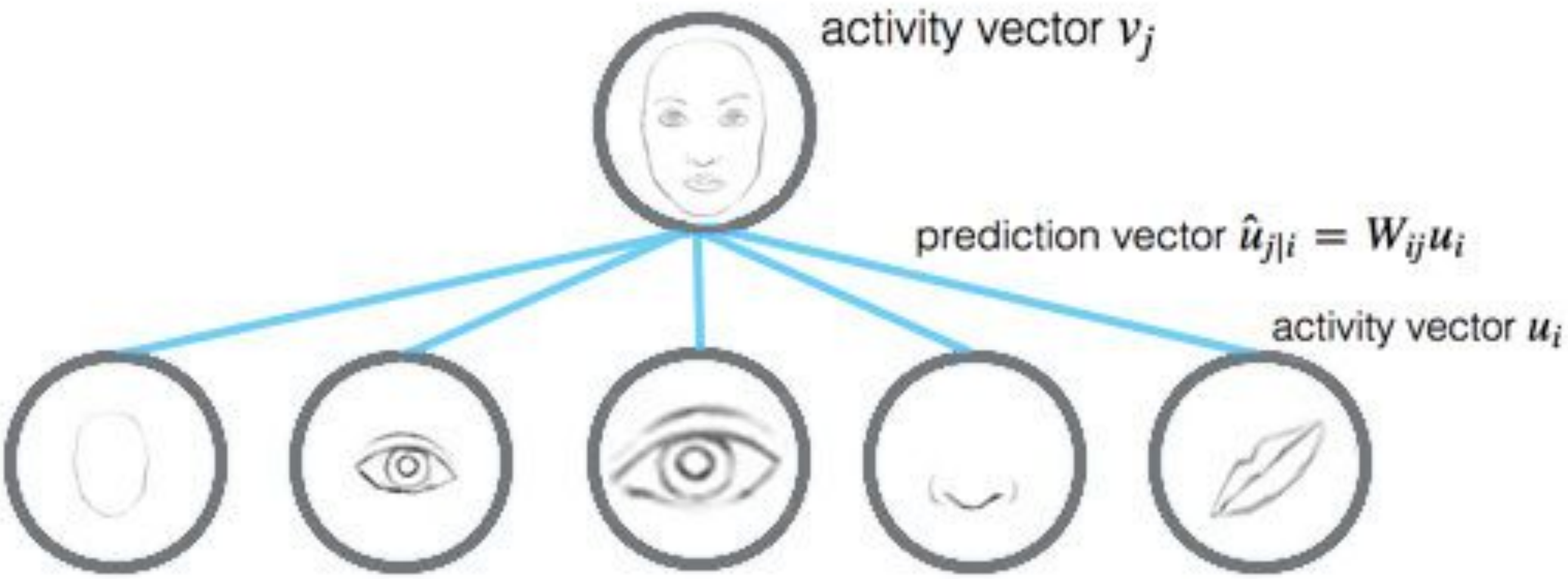
- Digit Caps are 16-D
  - Hold more information than Primary Caps (8-D) since they are looking at whole images
- An algorithm called **routing** is utilized to learn the weights of the transformation matrix Wij
  - Each Wij is 8(in)x16(out)

Role of Primary Capsule: “Which higher capsule am I a part of?”



- The square and the triangle will agree on the boat but disagree on the house due to the routing algorithm. How?

Calculation for Predicting the Output of the Next Layer



W_{ij} is 16x8, u_i is 8x1 -> $W_{ij}u_i = u_{j|i}$ is 1x16 -> Higher level capsules are 16-D

All are put in a big 1152 (Primary Capsules) x 10 (Digit Capsules) matrix

Calculating the Affine Transformation Matrices is a challenge

Batch size

$$
\begin{pmatrix}
\mathbf{W}_{1,1} & \mathbf{W}_{1,2} & \cdots & \mathbf{W}_{1,10} \\
\mathbf{W}_{2,1} & \mathbf{W}_{2,2} & \cdots & \mathbf{W}_{2,10} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{W}_{1152,1} & \mathbf{W}_{1152,2} & \cdots & \mathbf{W}_{1152,10}
\end{pmatrix}
\cdot
\begin{pmatrix}
\mathbf{u}_{1} & \mathbf{u}_{1} & \cdots & \mathbf{u}_{1} \\
\mathbf{u}_{2} & \mathbf{u}_{2} & \cdots & \mathbf{u}_{2} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{u}_{1152} & \mathbf{u}_{1152} & \cdots & \mathbf{u}_{1152}
\end{pmatrix}
=
\begin{pmatrix}
\hat{\mathbf{u}}_{1|1} & \hat{\mathbf{u}}_{2|1} & \cdots & \hat{\mathbf{u}}_{10|1} \\
\hat{\mathbf{u}}_{1|2} & \hat{\mathbf{u}}_{2|2} & \cdots & \hat{\mathbf{u}}_{10|2} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{\mathbf{u}}_{1|1152} & \hat{\mathbf{u}}_{2|1152} & \cdots & \hat{\mathbf{u}}_{10|1152}
\end{pmatrix}
$$

- Multiplication of 1152x10xBatch_Size pairs of matrices in each iter.
  - Note the dot, not a regular matrix multiplication of two huge matrices.
  - Must be done parallel on GPU, tile() & matmul() would come in handy.

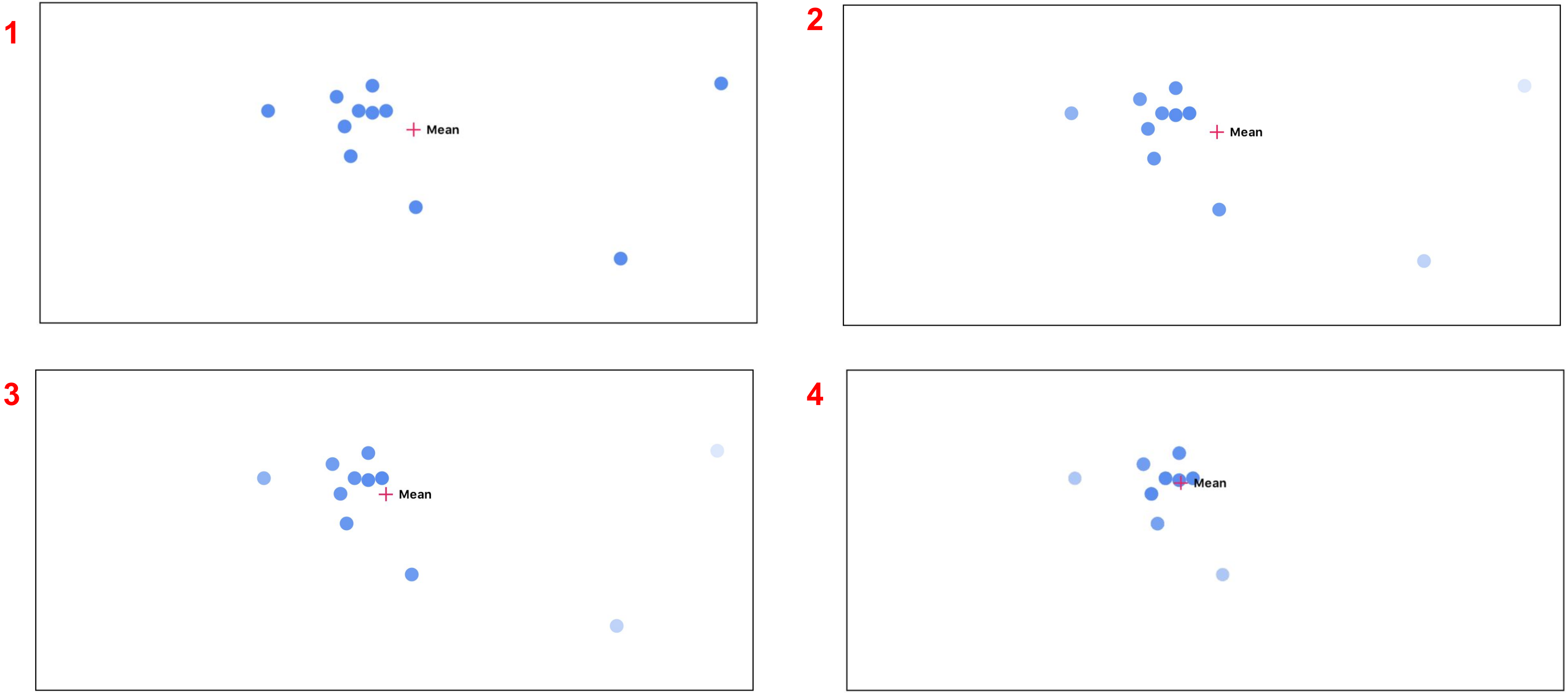
Details of the Routing Algorithm

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 $\mathbf{v}_j$

- The **r** iterations increase the weight of primary ui to estimate digit vj when the affine transformation of ui is aligned with vj (uhat([j|i].vj))
  - Repeatedly calculate the cluster center and adjust weights by cosine similarity: cos(0)=1 when aligned, cos(90)=0 when orthogonal
  - Would you agree that something is weird about **line 7**?

Routing Algorithm In Action
(Darker=Bigger Value)



Calculating the Loss

CapsNet Loss Function

calculated for correct DigitCap

calculated for incorrect DigitCaps

loss term for one DigitCap

L2 norm

L2 norm

$$\boxed{L_c} = \boxed{\boxed{T_c}\boxed{\max(0, m^+ - ||\mathbf{v}_c||)^{\boxed{2}}}} + \boxed{\boxed{\lambda}\boxed{(1 - T_c)}\boxed{\max(0, ||\mathbf{v}_c|| - m^-)^{\boxed{2}}}}$$

1 when correct DigitCap, 0 when incorrect

zero loss when correct prediction with probability greater than 0.9, non-zero otherwise

0.5 constant used for numerical stability

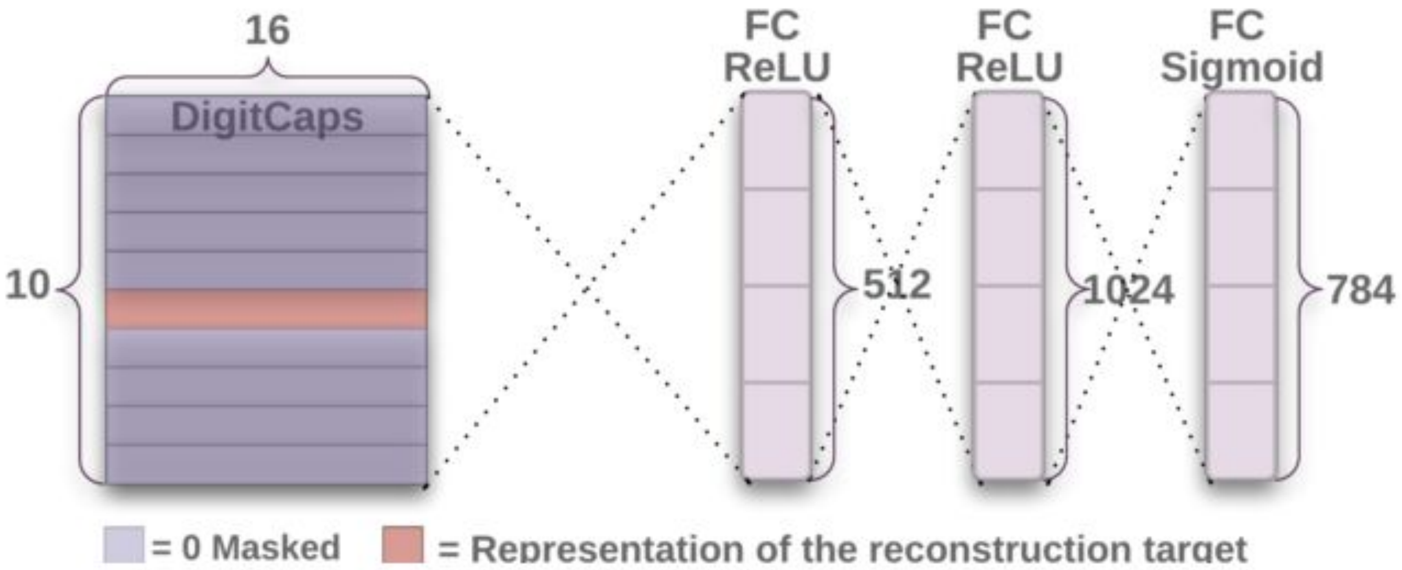
1 when incorrect DigitCap, 0 when correct

zero loss when incorrect prediction with probability less than 0.1, non-zero otherwise

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

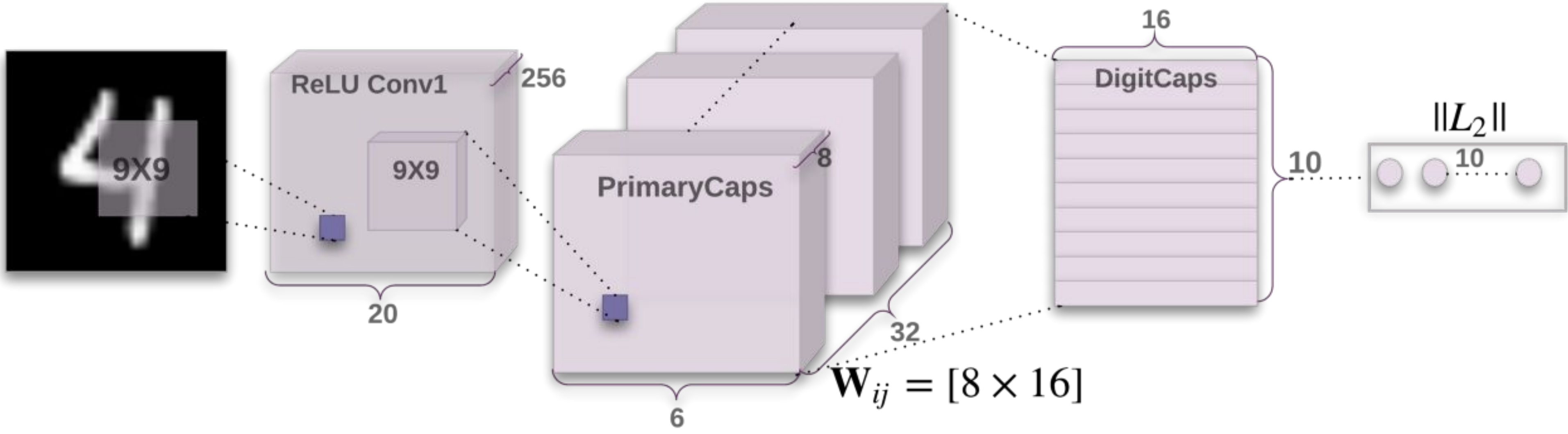
- Allows for detecting multiple images!
    - Can you detect a shortcoming?

Decoder Architecture



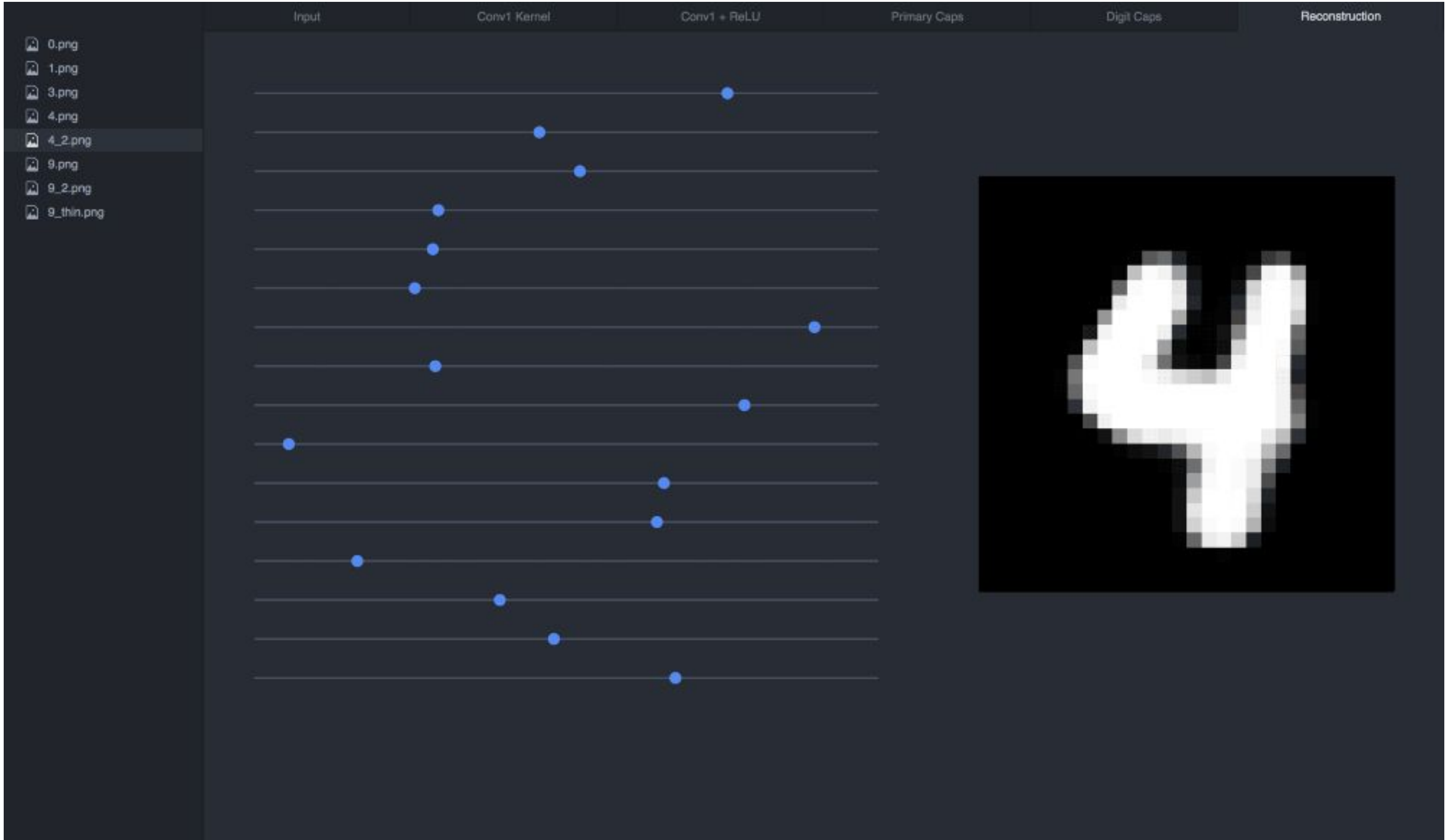
- Generates images from features to be compared against the input
  - The difference is an error, to be used as a regularizer

Summary



- Squashing the output of each DigitCaps gives the prob. of existence
  - Do they add up to 1?

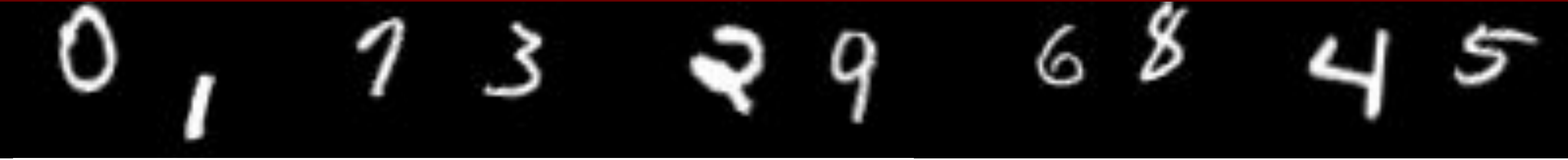
Reconstruction Visualization

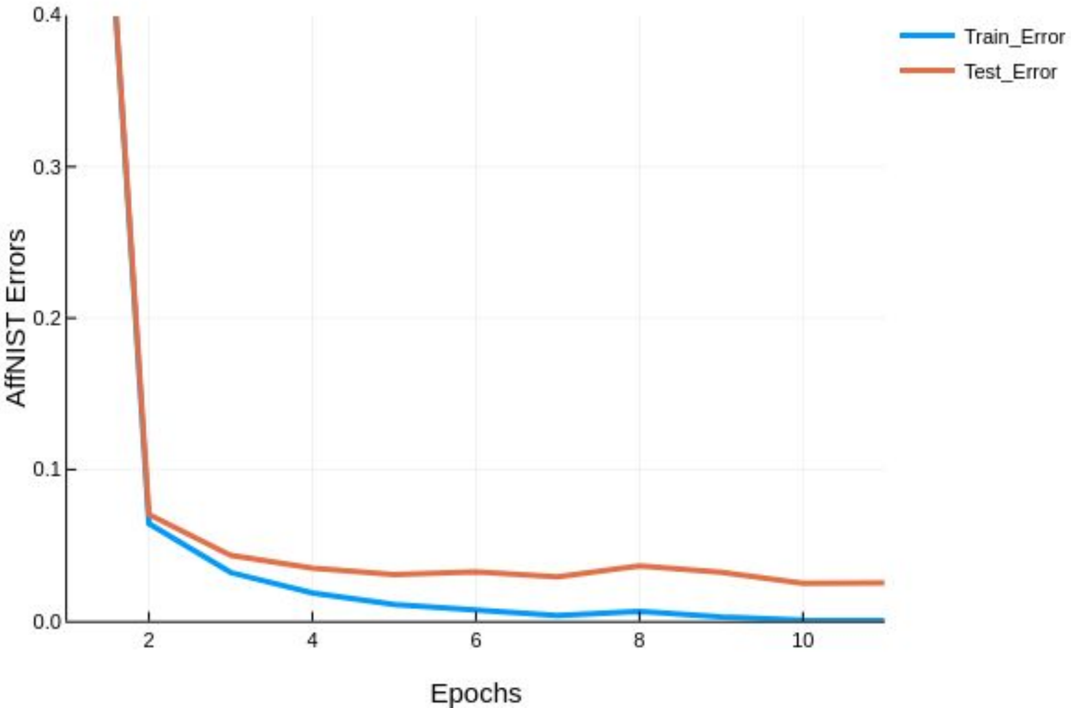


MNIST Error Results by Methods: Table1

METHOD	ROUTING r:	RECONSTRUCT	MNIST (%)	MY RESULT (Epoch)
Baseline	-	-	0.39	0.56 (40)
CapsNet	1	NO	0.34	0.74 (5)
CapsNet	1	YES	0.29	0.61 (16)
CapsNet	3	NO	0.35	0.86 (9)
CapsNet	3	YES	0.25	0.65 (12)

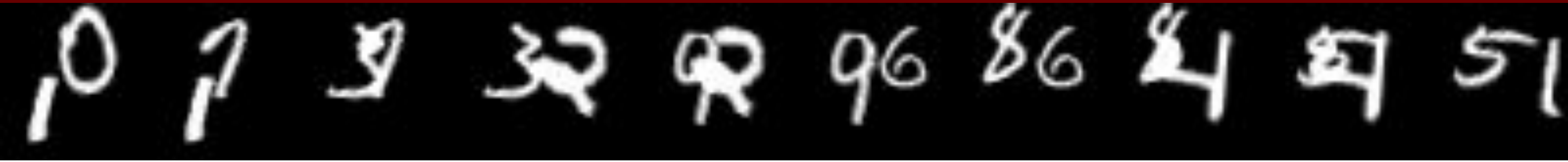
Promising Results for Viewpoint Invariance: The AffNIST Dataset





- In only 10 epochs (r=3):
  - Train Error < 0.03%
  - Test Error < 2.5%
- No previous training on transformed images
- Used 60K out of 2M affine-transformed MNIST images

Multi-MNIST Dataset: Not Publicly Available



- Tried to generate by superimposing AffNIST samples
- Remember sum of the output of Digit Caps != 1?
  - Probability of existence for each digit -> Can detect multiple digits!
- Remember the shortcoming of Loss Function?
  - Can not count the number of occurences of the same class
    - E.g. two 5’s or one 5 in the picture?
- My initial results (>50%) not comparable to reported figures (5-8%)
  - Dataset, training proc.,model parameters may be quite different

Q & A