Capsule Networks

--intuitions, models, and applications

LI, Jian
2018-10-09

Dynamic Routing Between Capsules

Sara Sabour **Nicholas Frosst**

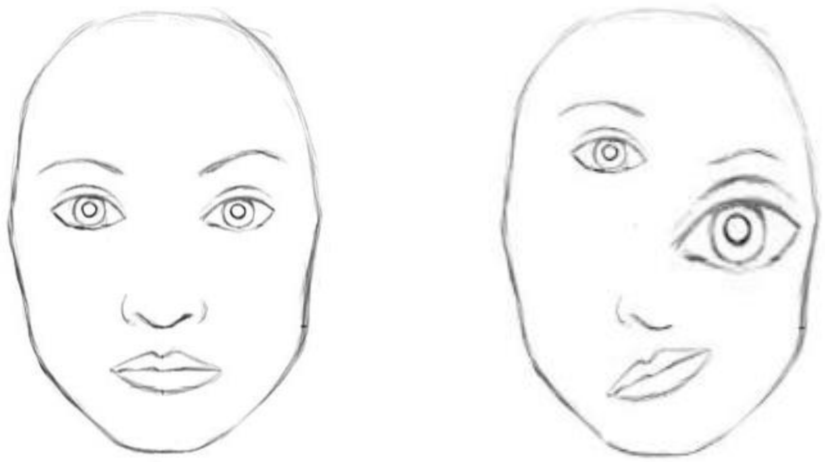
Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com

In NIPS 2017

Drawbacks of CNN

- Computer Vision today: CNN + (Max) Pooling.

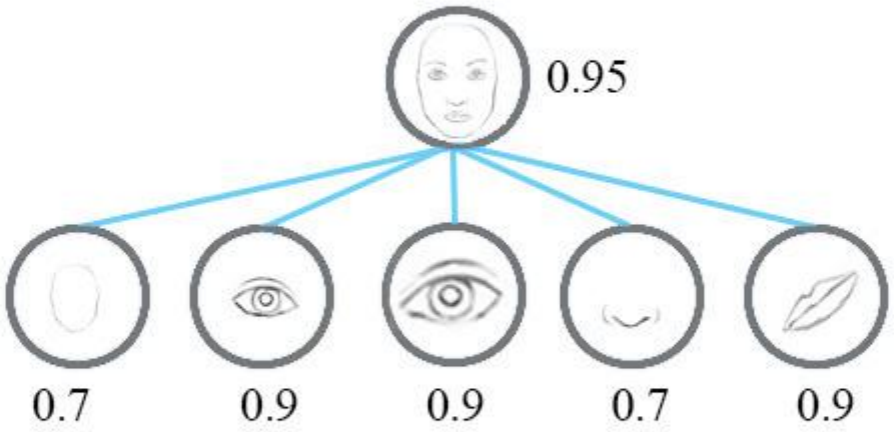
- CNN is good at detecting features but less effective at exploring the spatial relationships (perspective, size, orientation).



- For a simple CNN, there are two human faces!

Drawbacks of CNN

- Higher-level features combine lower-level features as a weighted sum.



- Need pose relationships in lower-level features: translation and rotation.

- Current solution: max-pooling (translation invariant), data augmentation (viewpoint invariant).

Problems of Max-Pooling

- Max-pooling works well, but loses valuable information and does not encode important spatial hierarchies between features.

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

- Bad information aggregation.

- Invariance V.S. Equivariance

Capsules

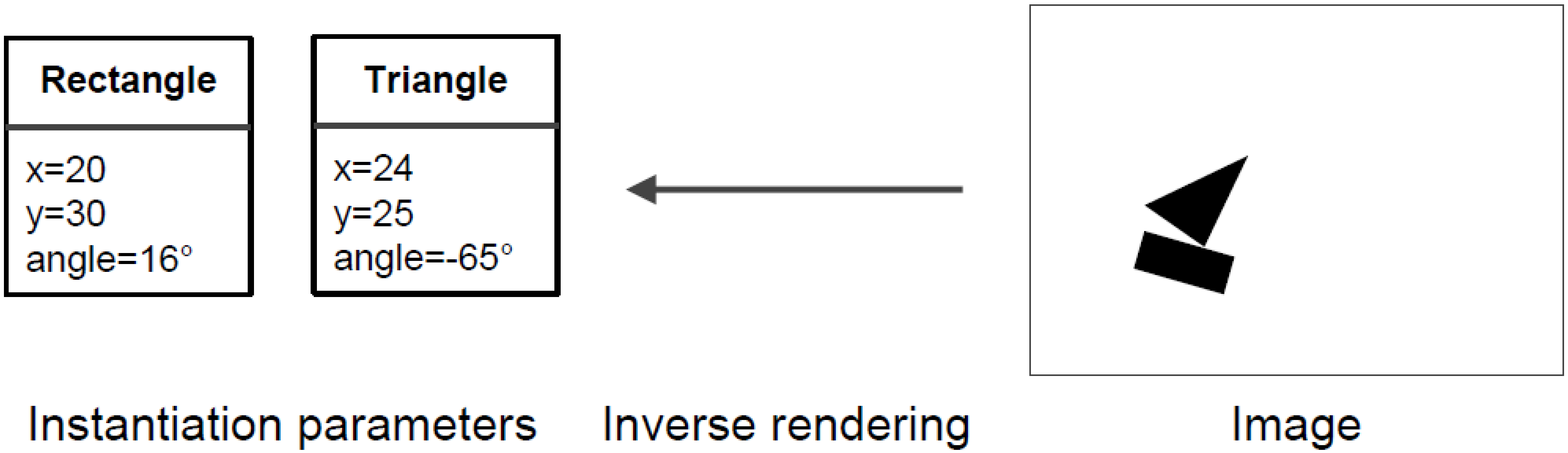
- Capsules encapsulate all important information about the state of the feature they are detecting in vector form (V.S. neuron outputs a scalar).

- Length of the vector: probability of a feature

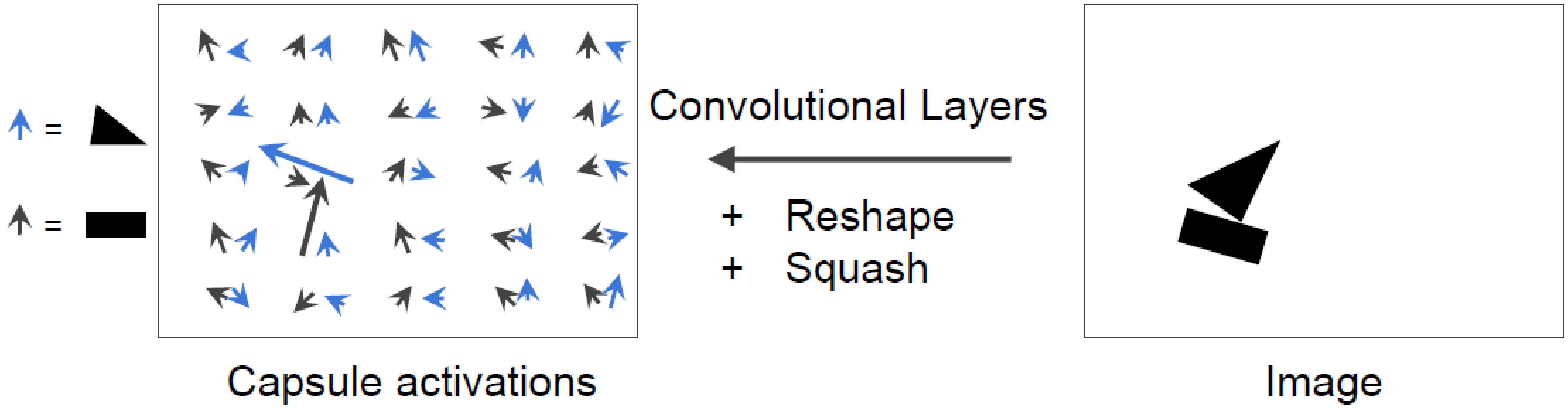
- Direction of the vector: state of the detected feature (“instantiation parameters”).

- When entity moves over the manifold of possible appearances: length remains constant but direction changes. (Equivariance)

Capsules Examples

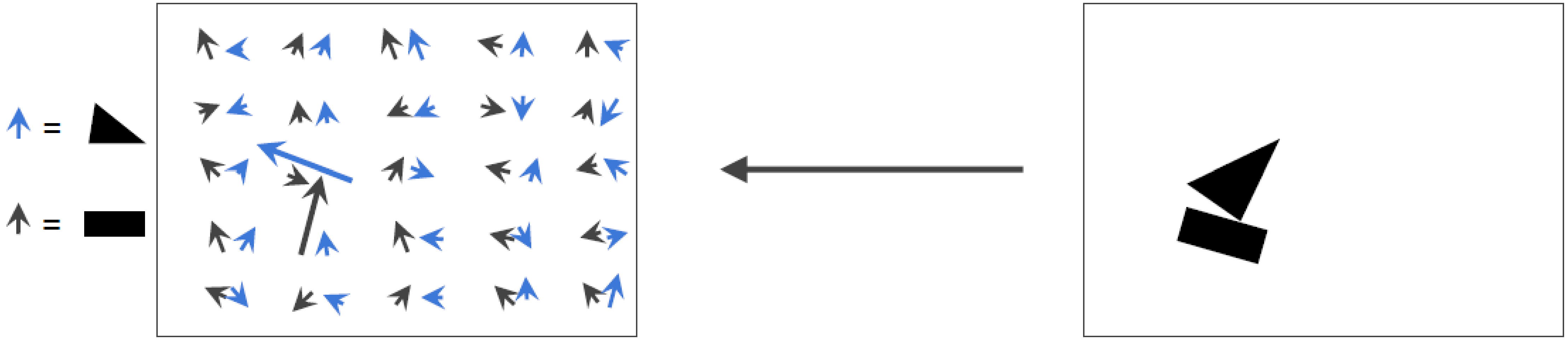


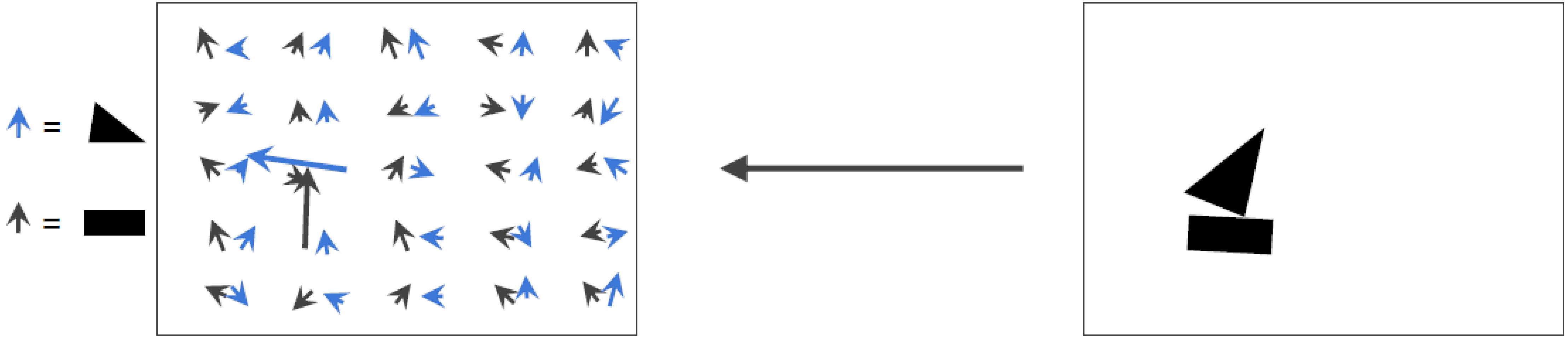
Capsules Examples



Activation vector:	**Length** = estimated probability of presence **Orientation** = object's estimated pose parameters

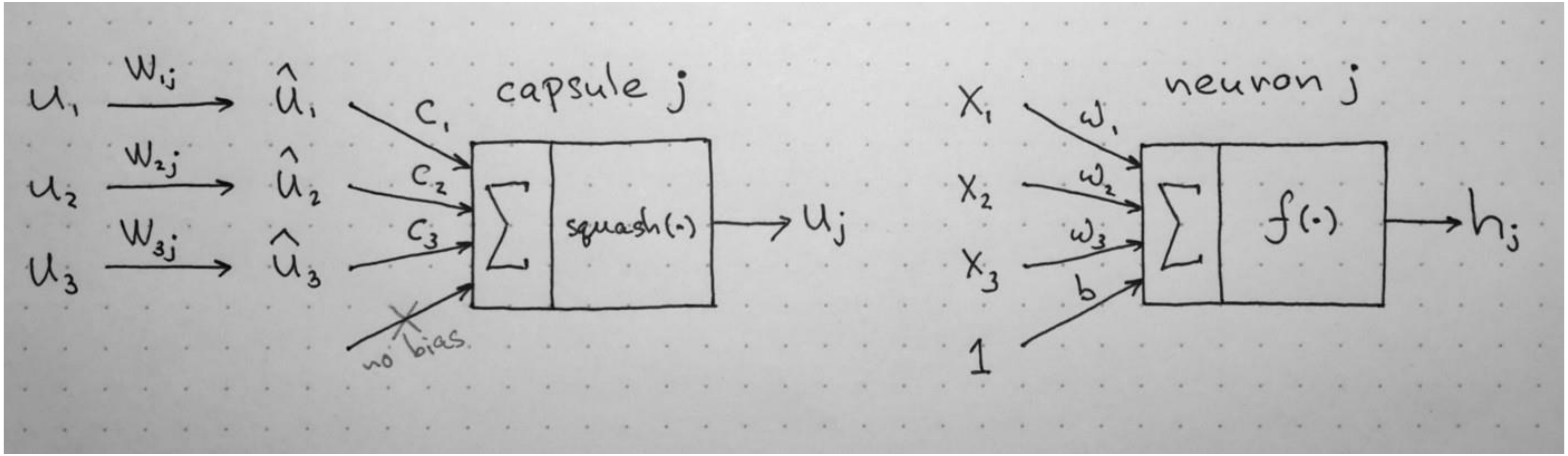
Equivariance



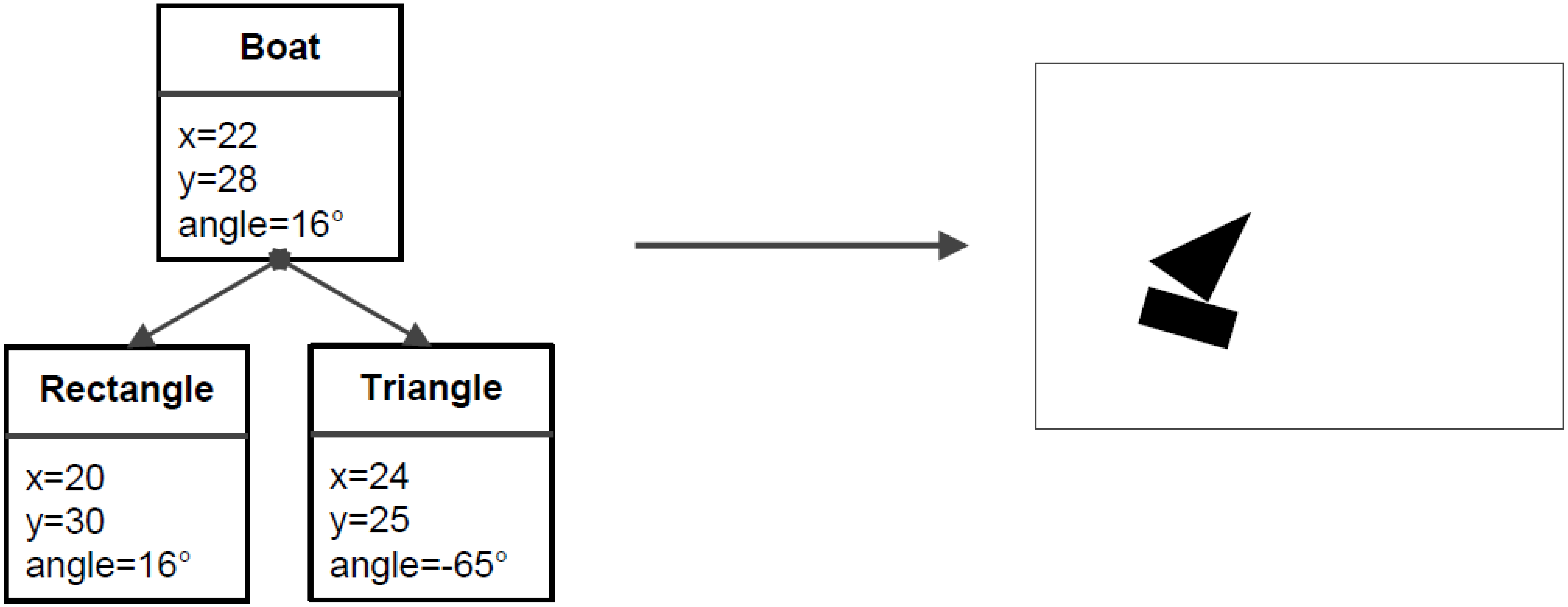


Capsules

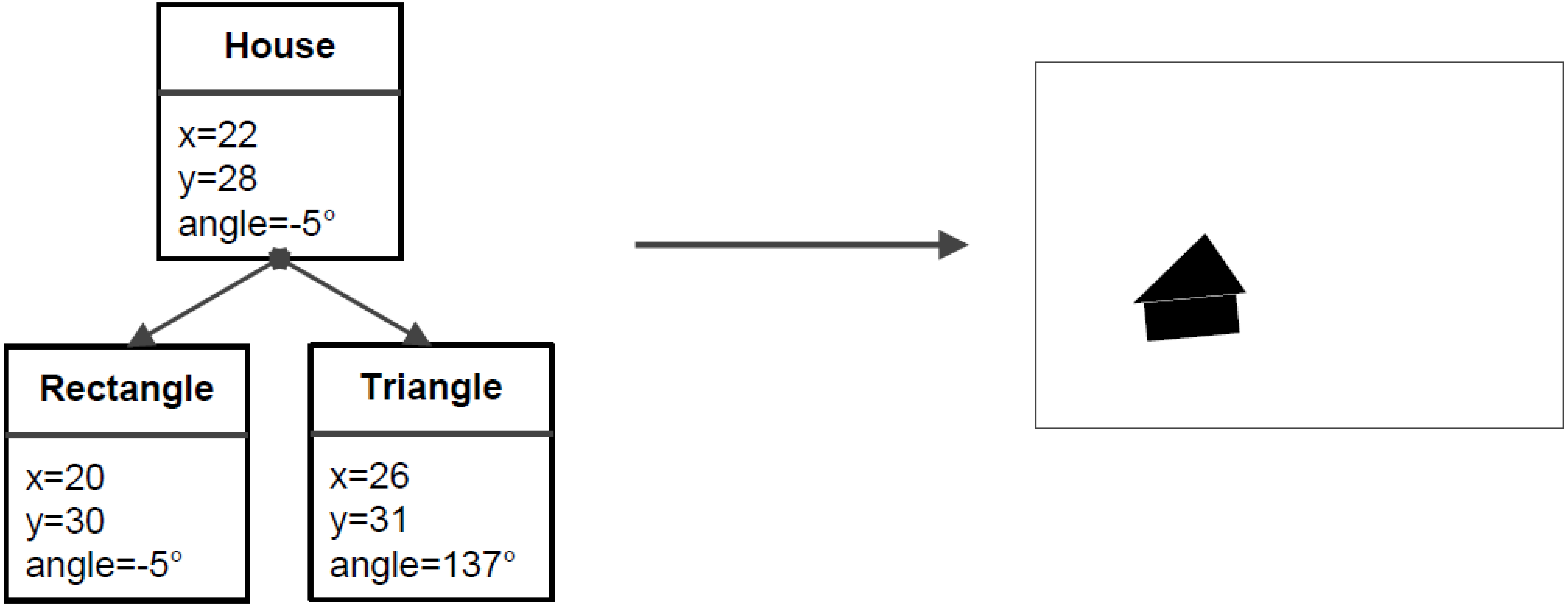
Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\hat{\mathbf{u}}_{j\vert i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\vert i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear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



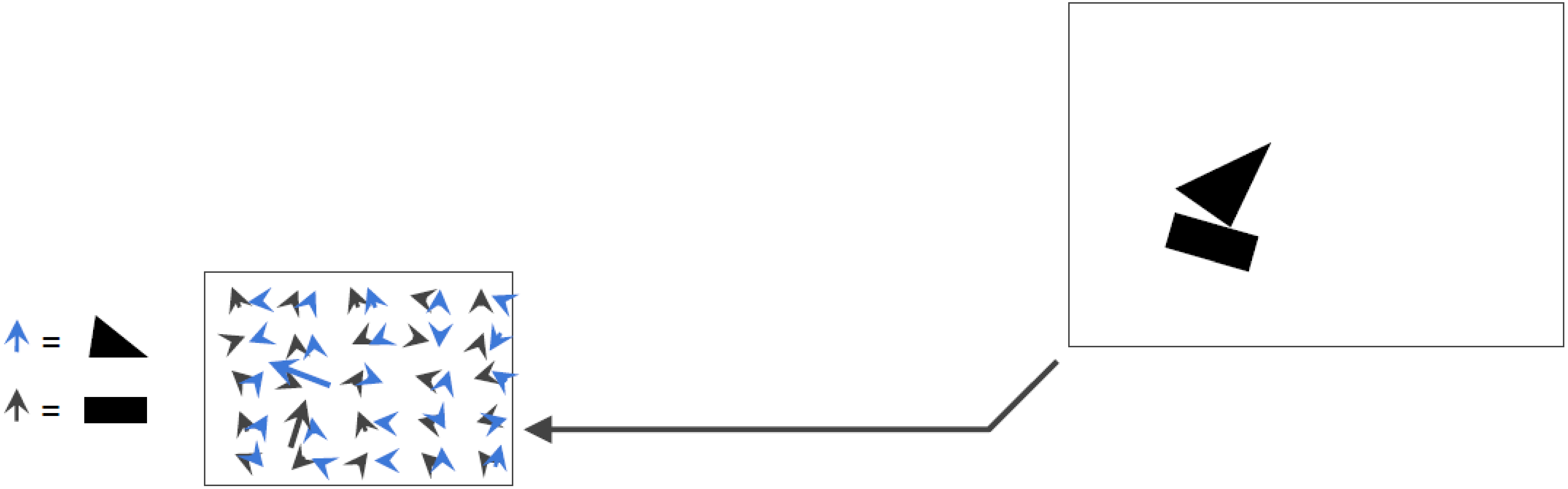
Capsules Examples



Capsules Examples

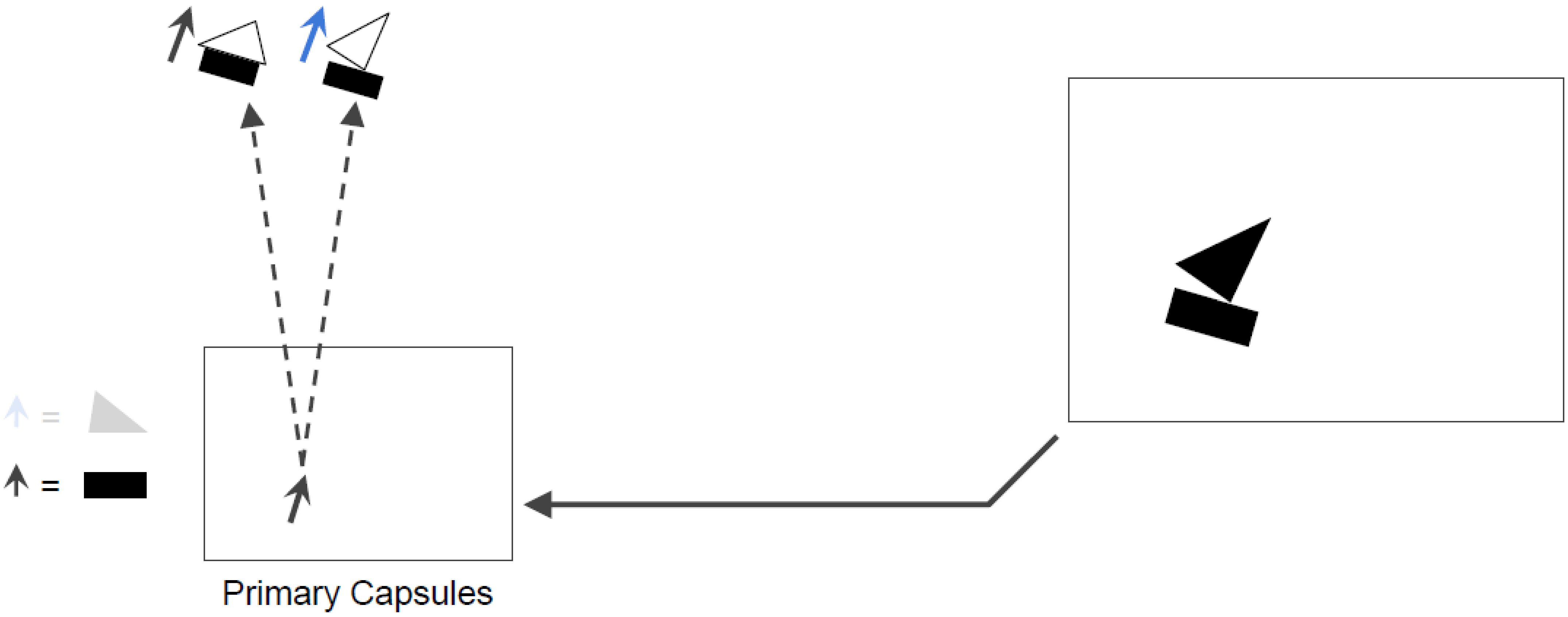


Primary Capsules

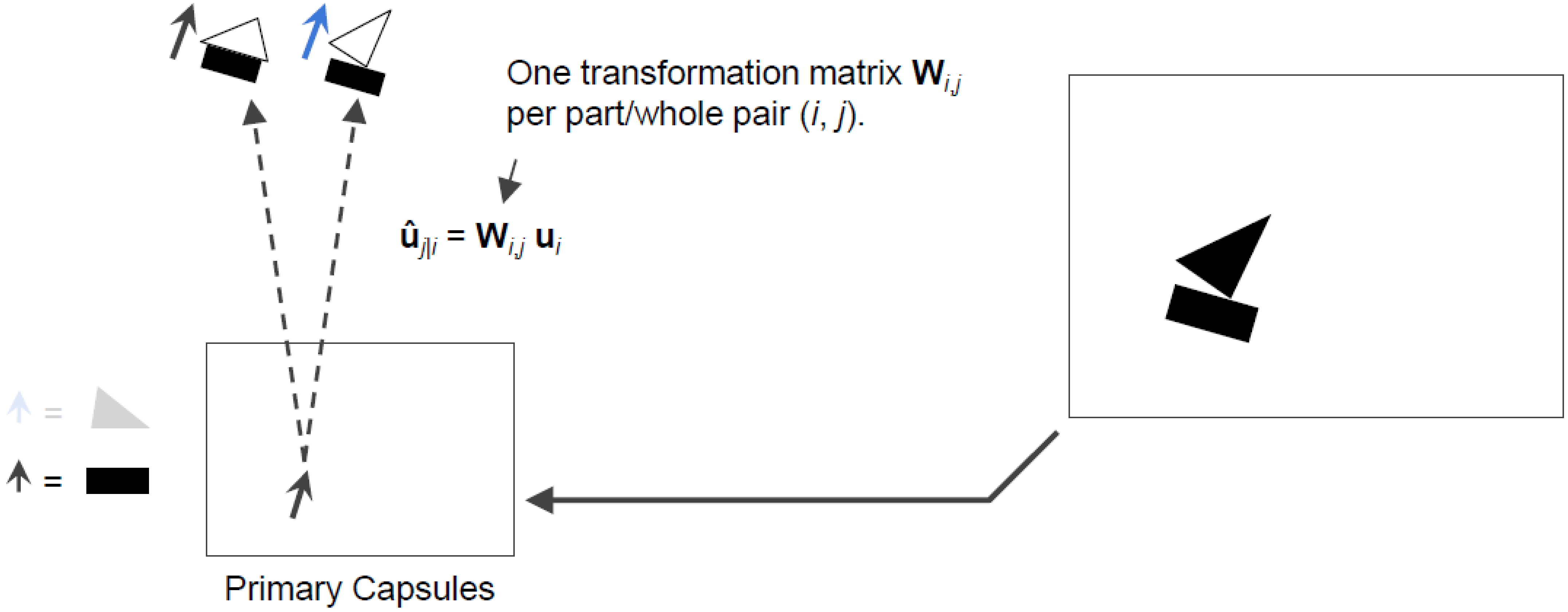


Primary Capsules

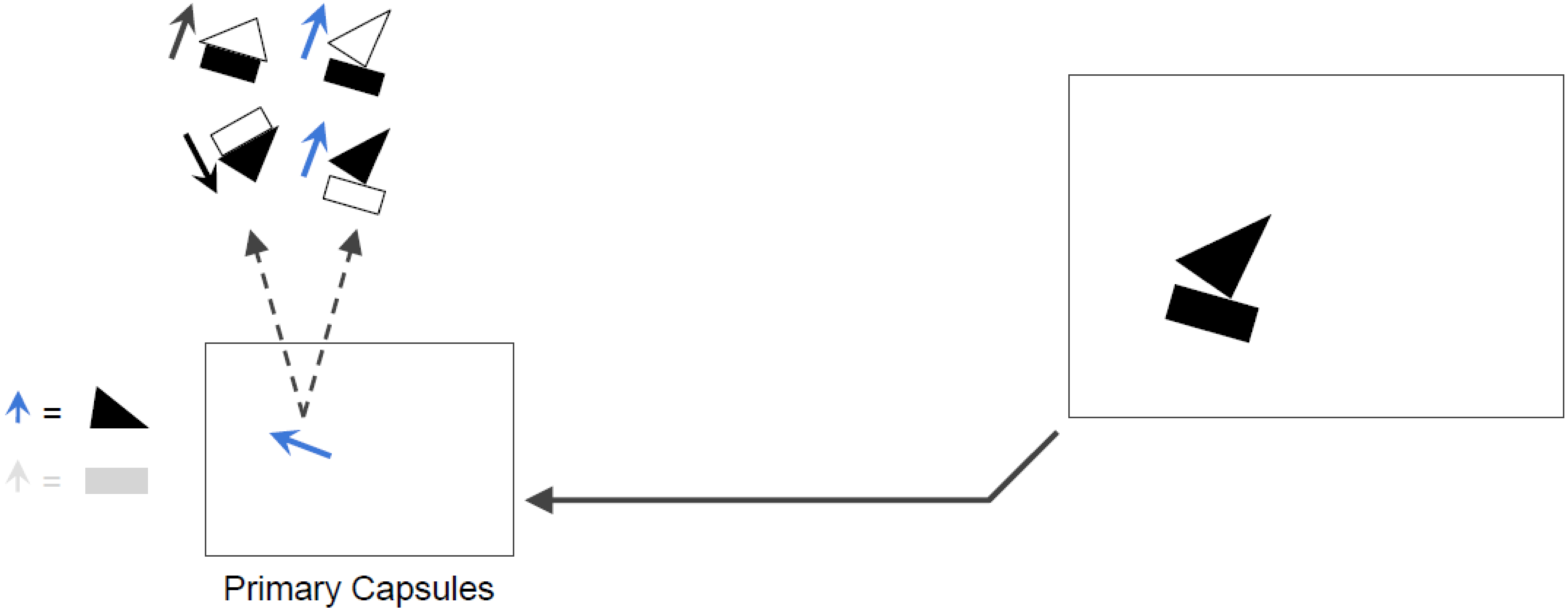
Predict Next Layer's Output



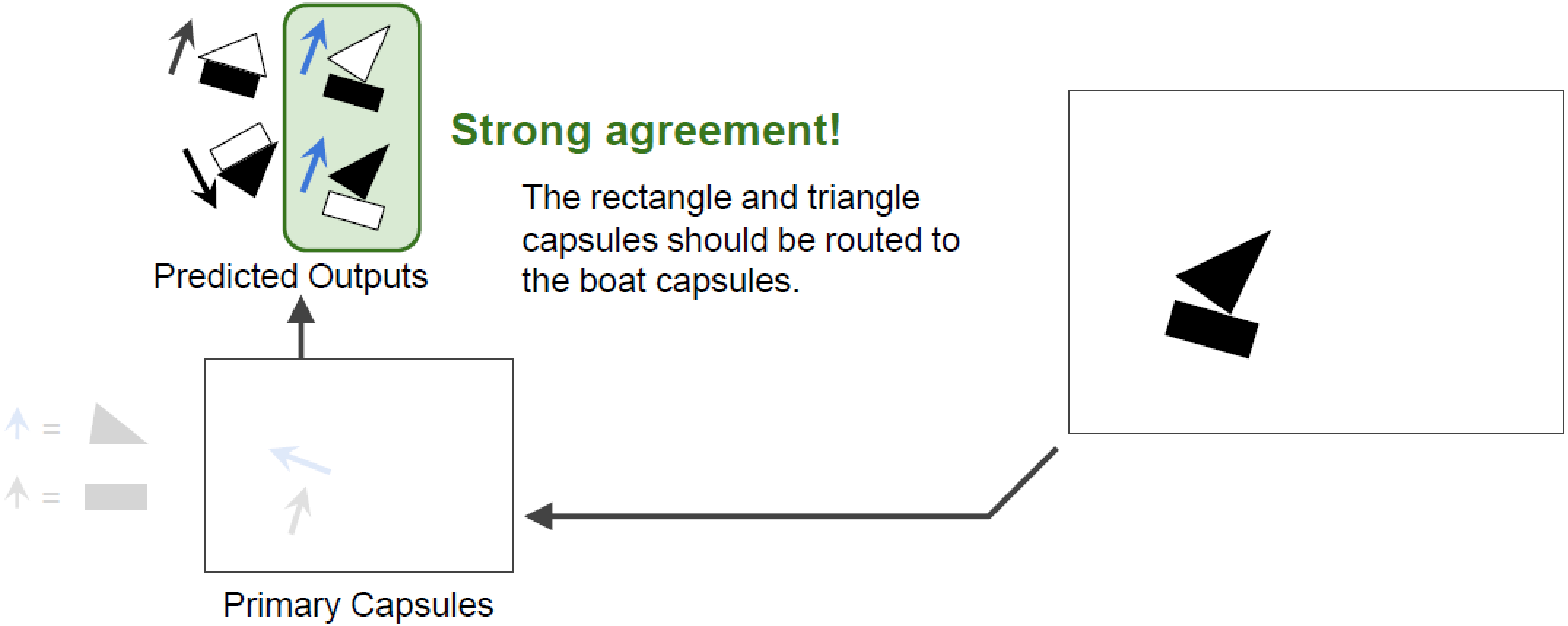
Predict Next Layer's Output



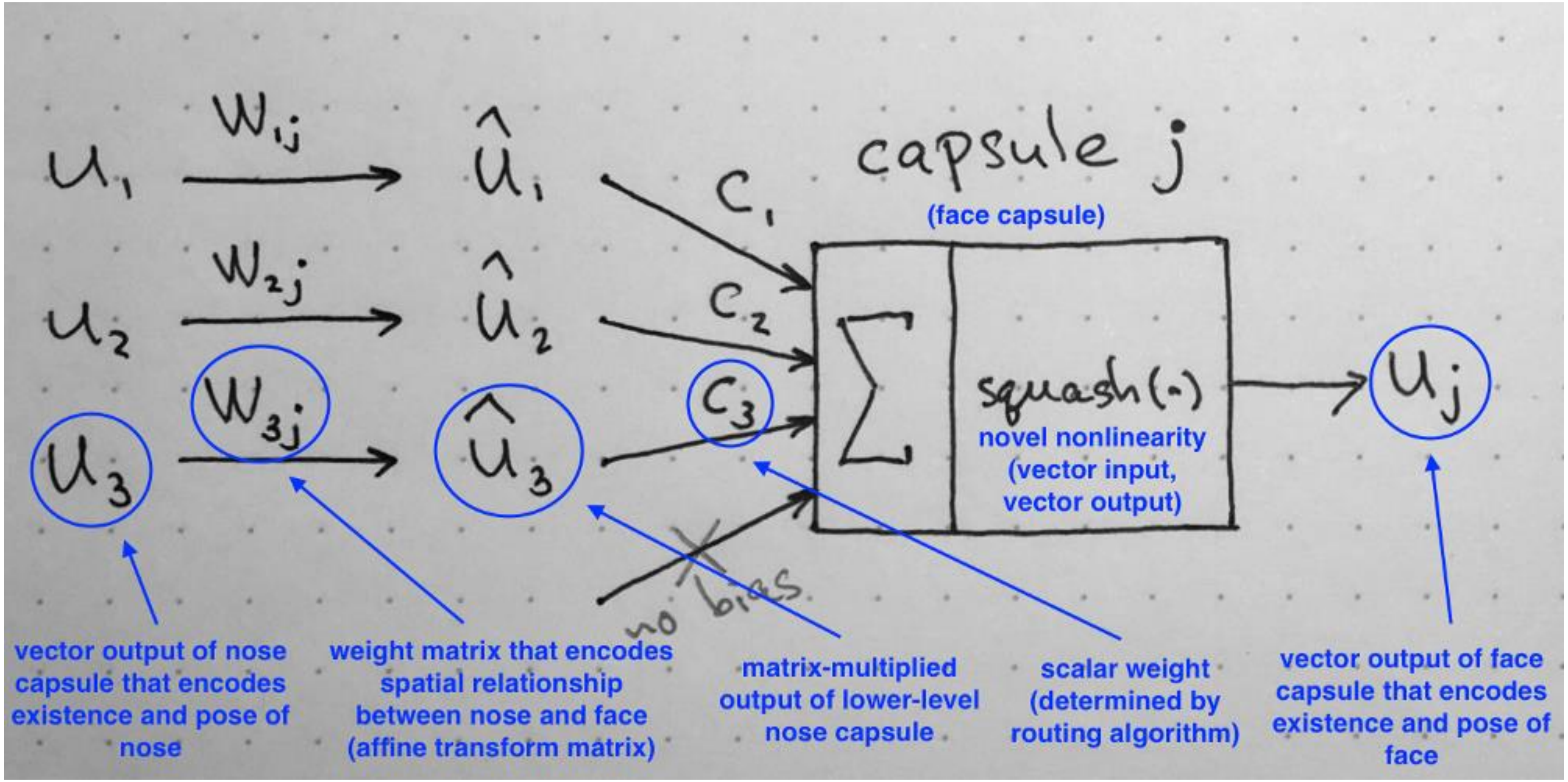
Predict Next Layer's Output



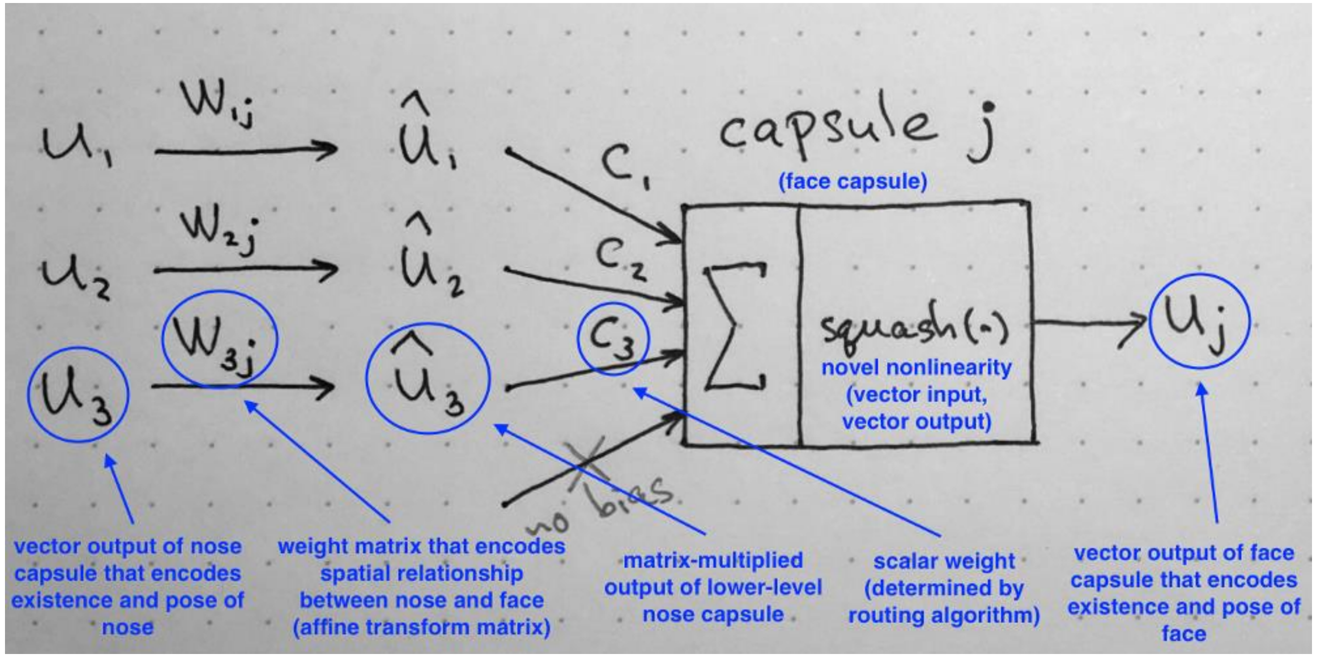
Routing by Agreement



Capsule Computation



How to decide the scalar weights *C* ?



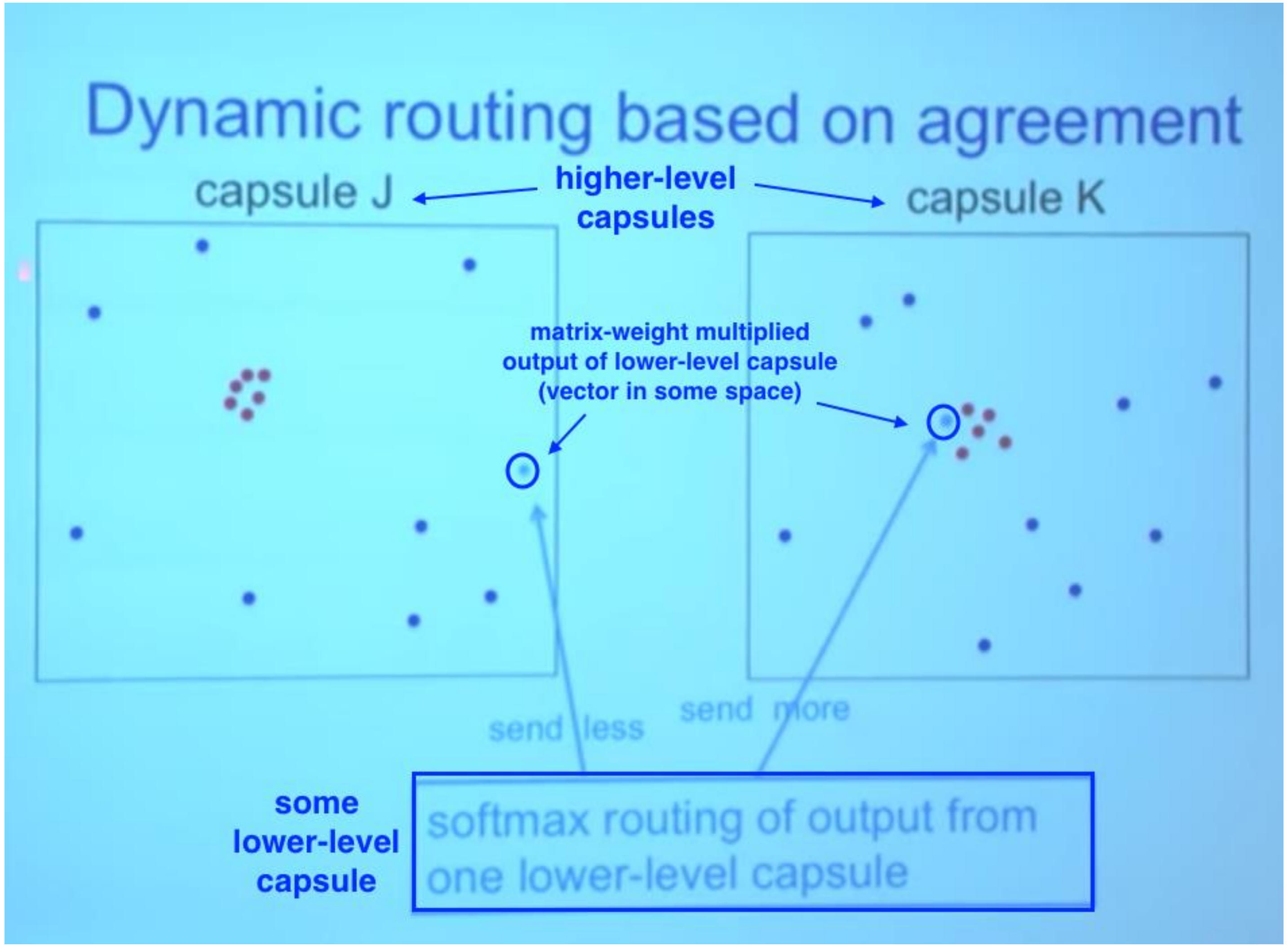
Scalar Weights

- In the neuron case, the weights are learned during backpropagation.

- Re-calculate the weights for every datapoint including the testing data.

Dynamic Routing

- Lower level capsule will send its output to the higher level capsule that "agrees" with its output.



Iterative 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 $\mathbf{v}_j$

Prediction vector (vote): $\hat{u}_{j|i} = W_{ij} u_i$

Iterative Dynamic Routing

Procedure 1 Routing algorithm.

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

Similarity score: b_{ij}

Coefficient: $c_{ij} = \dfrac{\exp b_{ij}}{\sum_k \exp b_{ik}}$

Iterative Dynamic Routing

Procedure 1 Routing algorithm.

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

$$s_j = \sum_i c_{ij}\hat{u}_{j|i}$$

$$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2}\frac{s_j}{\|s_j\|}$$ (output capsule)

Iterative Dynamic Routing

Procedure 1 Routing algorithm.

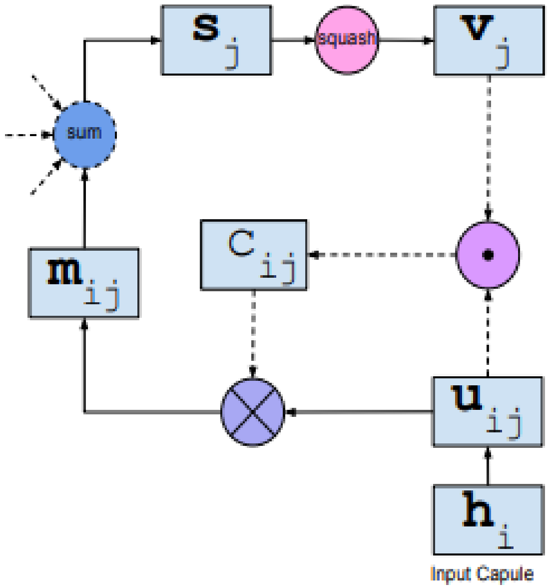
procedure ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \mathtt{softmax}(\mathbf{b}_i)$ ▷ $\mathtt{softmax}$ computes Eq. 3
    for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

Dot product captures similarity.

Iterative Dynamic Routing

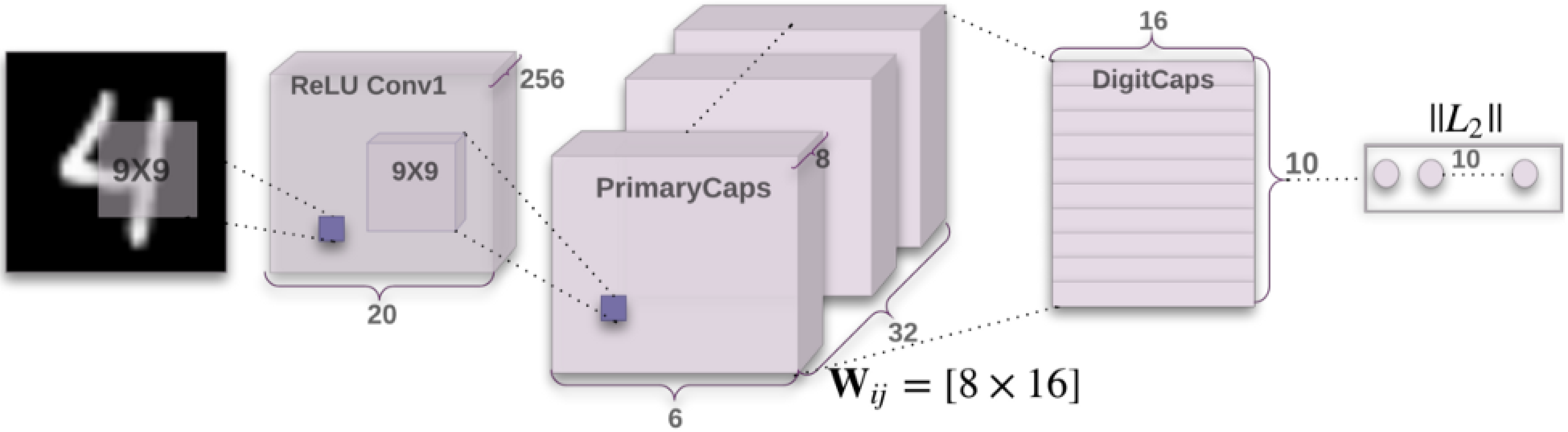
Procedure 1 Routing algorithm.

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



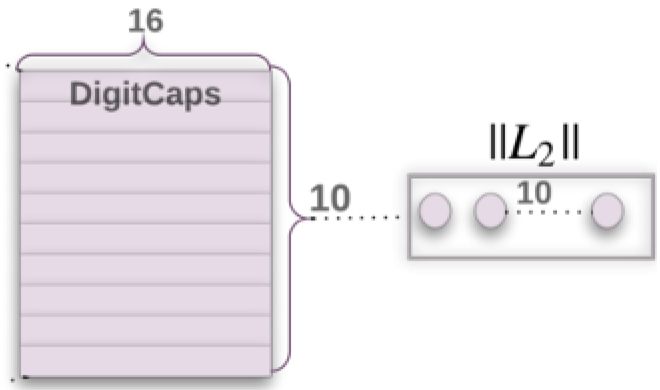
r = 3

CapsNet



Input capsules: 6x6x32 8-dimensional vectors
Output capsules: 10 16-dimensional vectors

Margin Loss



CapsNet Loss Function

calculated for correct DigitCap

calculated for incorrect DigitCaps

loss term for one DigitCap

L2 norm

L2 norm

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

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

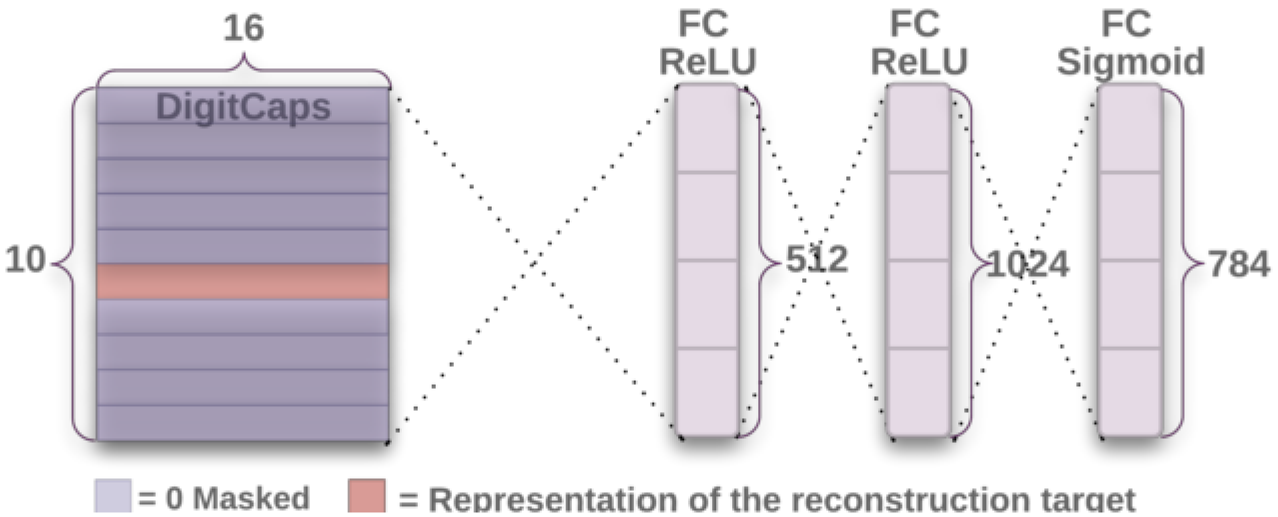
1 when incorrect DigitCap, 0 when correct

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

CapsNet

Reconstruction as regularization:



$$Loss = margin\_loss + \alpha * reconstruction\_loss$$

MNIST Classification

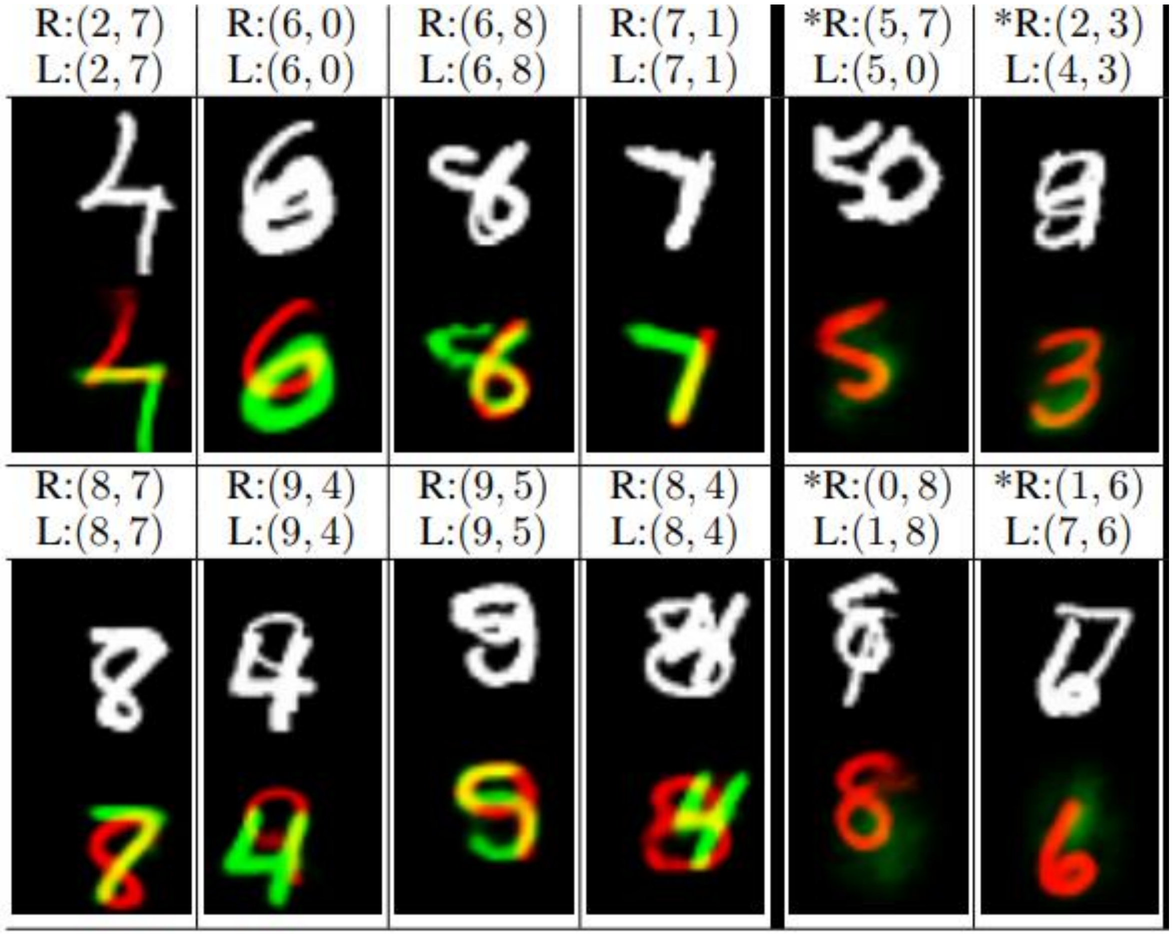
Table 1: CapsNet classification test accuracy. The MNIST average and standard deviation results are reported from 3 trials.

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

Parameters: baseline has 35.4M while CapsNet has 8.2M parameters and 6.8M parameters without the reconstruction subnetwork.

Segmenting Highly Overlapping Digits

L: labels
R: reconstruction



Limitation of Dynamic Routing

- The squash non-linearity prevents sensible objective function.

- Cosine of the angle saturates at 1, insensitive to good agreement and very good agreement.

MATRIX CAPSULES WITH EM ROUTING

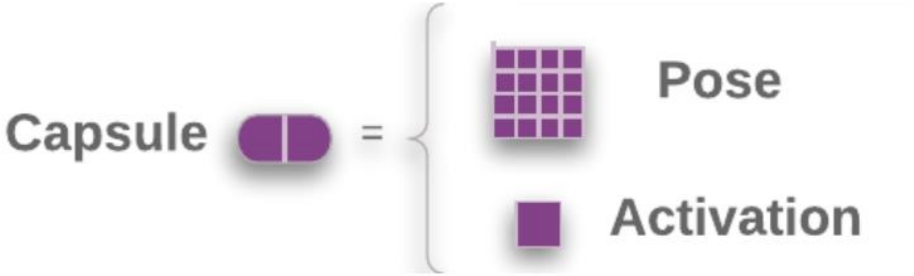
Geoffrey Hinton, Sara Sabour, Nicholas Frosst
Google Brain
Toronto, Canada
{geoffhinton, sasabour, frosst}@google.com

In ICLR 2018

vector capsule -> matrix capsule
dynamic routing -> em routing

Matrix Capsule

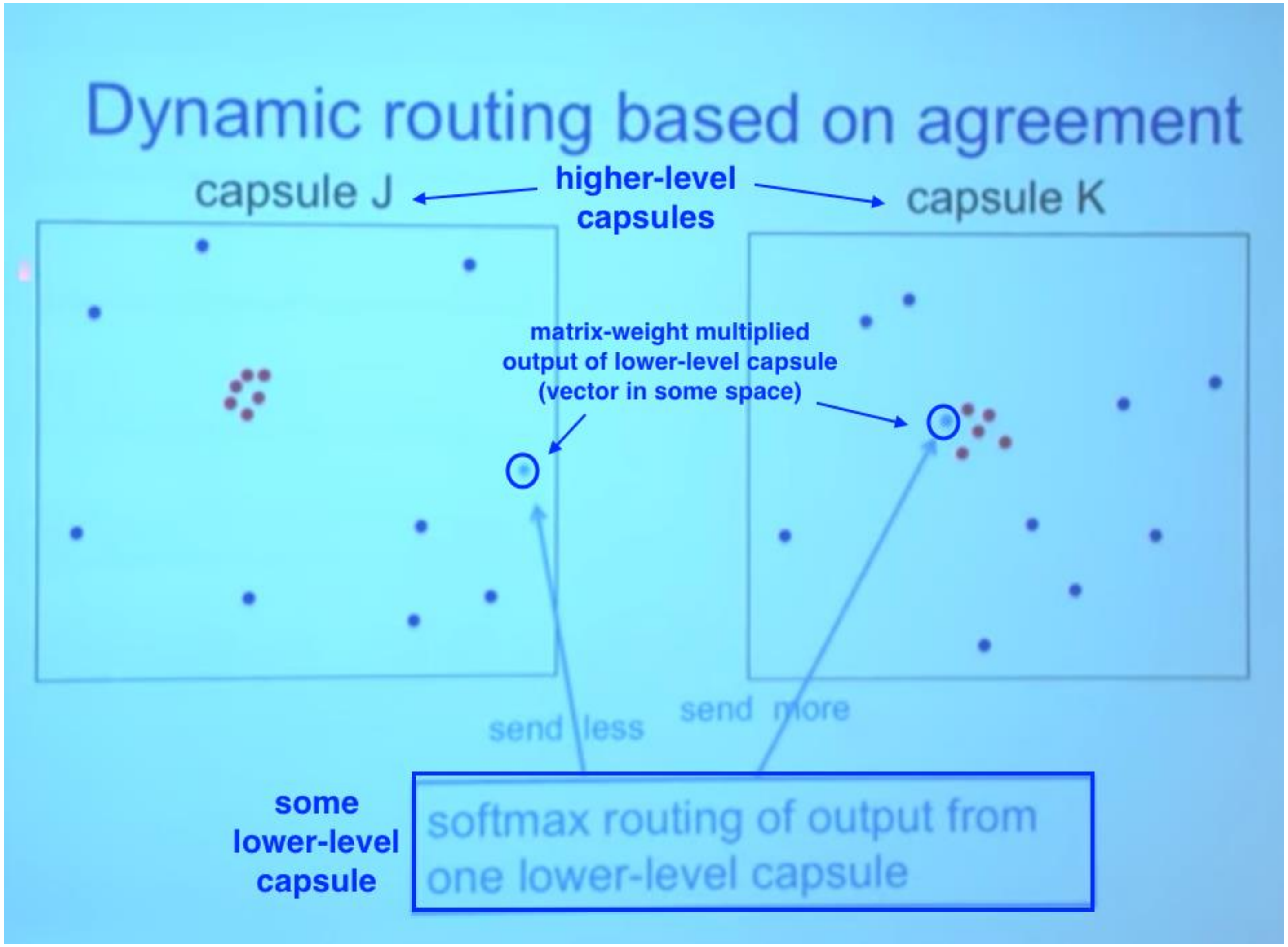
- Activation (likeliness) + 4x4 pose matrix (viewpoint).



- In computer graphics, a pose matrix defines the translation and the rotation of an object.

Gaussian Mixture Clustering

- Each higher-level capsule acts as a Gaussian, with 16 μ and 16 σ.



Gaussian Mixture Clustering

- Let V_{ij} be the vote from child capsule i for the parent capsule j, and v_{ij}^h be its h-th component.

- The probability of v_{ij}^h belonging to Gaussian j:

$$p_{i|j}^h = \frac{1}{\sqrt{2\pi(\sigma_j^h)^2}} \exp\left(-\frac{(v_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}\right)$$

$$\ln(p_{i|j}^h) = \ln \frac{1}{\sqrt{2\pi(\sigma_j^h)^2}} \exp\left(-\frac{(v_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}\right)$$

$$= -\ln(\sigma_j^h) - \frac{\ln(2\pi)}{2} - \frac{(v_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}$$

Cost and Activation

- Let $cost$ be the negative log likelihood:

$$cost_{ij}^{h} = -\ln(P_{i|j}^{h})$$

- Summing over all lower-level capsules:

$$\begin{aligned} cost_j^h &= \sum_i -r_{ij} ln(P_{i|j}^h) \\ &= \frac{\sum_i r_{ij}(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} + \left(ln(\sigma_j^h) + \frac{ln(2\pi)}{2}\right)\sum_i r_{ij} \\ &= \left(ln(\sigma_j^h) + \frac{1}{2} + \frac{ln(2\pi)}{2}\right)\sum_i r_{ij} \end{aligned}$$

r_{ij} is the assignment probability, initially $1/|\Omega_{L+1}$

- Activation of parent capsule j:

$$a_j = logistic\left(\lambda\left(\beta_a - \beta_u \sum_i r_{ij} - \sum_h cost_j^h\right)\right)$$

Cost and Activation

- Let $cost$ be the negative log likelihood:

$$cost_{ij}^h = -\ln(P_{i|j}^h)$$

- Summing over all lower-level capsules:

$$\begin{aligned} cost_j^h &= \sum_i -r_{ij} ln(P_{i|j}^h) \\ &= \frac{\sum_i r_{ij}(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} + \left(ln(\sigma_j^h) + \frac{ln(2\pi)}{2} \right) \sum_i r_{ij} \\ &= \left(ln(\sigma_j^h) + \frac{1}{2} + \frac{ln(2\pi)}{2} \right) \sum_i r_{ij} \end{aligned}$$

- Activation of parent capsule j:

$$a_j = logistic\left(\lambda \left(\beta_a - \beta_u \sum_i r_{ij} - \sum_h cost_j^h \right) \right)$$

Parameters to estimate:

$r_{ij}, \mu_j, \sigma_j, a_j$

r_{ij} is the assignment probability, initially $1/|\Omega_{L+1}$

EM Algorithm

- Parameters to estimate: $r_{ij}, \mu_j, \sigma_j, a_j$

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
高斯混合成分个数 k.

过程:

初始化高斯混合分布的模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$
repeat
  for $j = 1, 2, \ldots, m$ **do**
    根据式(9.30)计算 $\boldsymbol{x}_j$ 由各混合成分生成的后验概率, 即 $\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \boldsymbol{x}_j)$ $(1 \leqslant i \leqslant k)$
  end for

E step

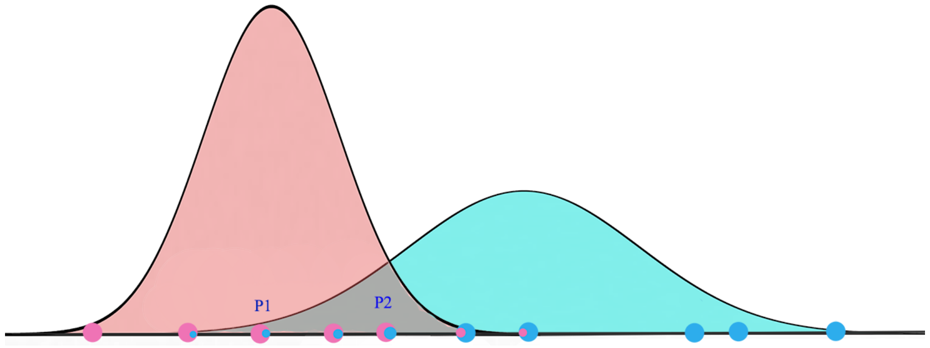
  for $i = 1, 2, \ldots, k$ **do**
    计算新均值向量: $\boldsymbol{\mu}'_i = \frac{\sum_{j=1}^m \gamma_{ji} \boldsymbol{x}_j}{\sum_{j=1}^m \gamma_{ji}}$;
    计算新协方差矩阵: $\boldsymbol{\Sigma}'_i = \frac{\sum_{j=1}^m \gamma_{ji} (\boldsymbol{x}_j - \boldsymbol{\mu}'_i)(\boldsymbol{x}_j - \boldsymbol{\mu}'_i)^{\mathrm{T}}}{\sum_{j=1}^m \gamma_{ji}}$;
    计算新混合系数: $\alpha'_i = \frac{\sum_{j=1}^m \gamma_{ji}}{m}$;
  end for

M step

  将模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$ 更新为 $\{(\alpha'_i, \boldsymbol{\mu}'_i, \boldsymbol{\Sigma}'_i) \mid 1 \leqslant i \leqslant k\}$
until 满足停止条件
$C_i = \varnothing$ $(1 \leqslant i \leqslant k)$
for $j = 1, 2, \ldots, m$ **do**
  根据式(9.31)确定 $\boldsymbol{x}_j$ 的簇标记 λ_j;
  将 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \bigcup \{\boldsymbol{x}_j\}$
end for

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

图 9.6 高斯混合聚类算法



EM Algorithm

procedure EM ROUTING($\boldsymbol{a}$, V)
  $\forall i \in \Omega_L, j \in \Omega_{L+1}$: $R_{ij} \leftarrow 1/|\Omega_{L+1}|$
  for t iterations **do**
    $\forall j \in \Omega_{L+1}$: M-STEP($\boldsymbol{a}$, R, V, j)
    $\forall i \in \Omega_L$: E-STEP(μ, σ, $\boldsymbol{a}$, V, i)
  return $\boldsymbol{a}$, M

- Inputs: activation and votes from the child capsules.
- Outputs: activation and pose of the parent capsules.
- Pose matrix: reshape 16 μ to 4x4.
- Parameters to estimate: R_{ij}, μ_j, σ_j

EM Algorithm

procedure M-STEP($\boldsymbol{a}$, R, V, j) ▷ for one higher-level capsule

$$\forall i \in \Omega_L\text{: } R_{ij} \leftarrow R_{ij} * \boldsymbol{a}_i$$

$$\forall h\text{: } \mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$$

$$\forall h\text{: } (\sigma_j^h)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$$

$$cost^h \leftarrow (\beta_v + log(\sigma_j^h)) \sum_i R_{ij}$$

$$a_j \leftarrow sigmoid(\lambda(\beta_a - \sum_h cost^h))$$

Hold R_{ij} constant, adjust (μ_j, σ_j, a_j) for parent capsules

EM Algorithm

procedure E-STEP(μ, σ, $\boldsymbol{a}$, V, i) ▷ for one lower-level capsule

$$\forall j \in \Omega_{L+1}\colon \boldsymbol{p}_j \leftarrow \frac{1}{\sqrt{\prod_h^H 2\pi(\sigma_j^h)^2}} e^{-\sum_h^H \frac{(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}}$$

$$\forall j \in \Omega_{L+1}\colon \boldsymbol{R}_{ij} \leftarrow \frac{\boldsymbol{a}_j p_j}{\sum_{u \in \Omega_{L+1}} \boldsymbol{a}_u p_u}$$

Hold (μ_j, σ_j, a_j) constant, adjust R_{ij} for child capsules

CapsNet

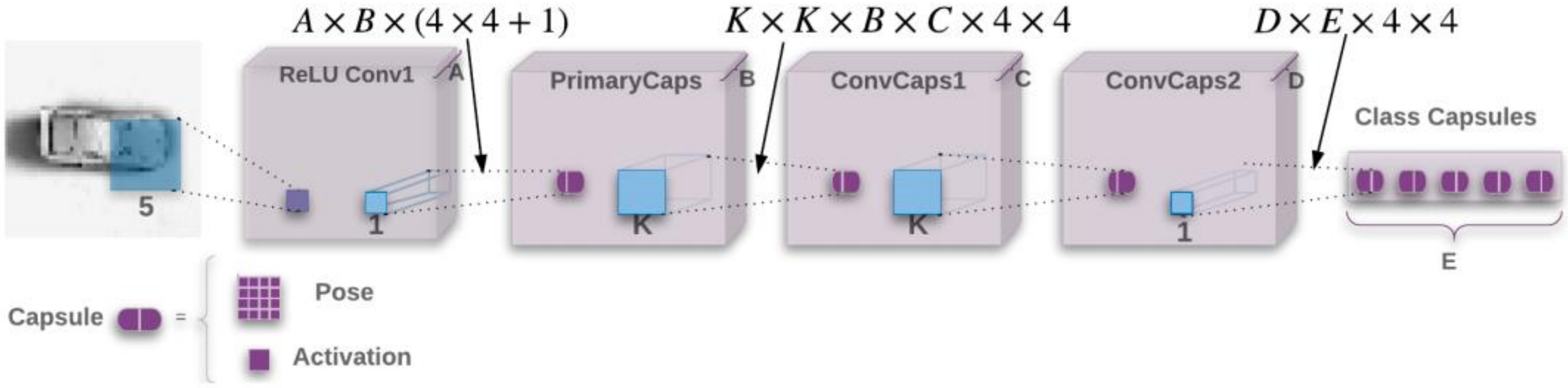


Figure 1: A network with one ReLU convolutional layer followed by a primary convolutional capsule layer and two more convolutional capsule layers.

Spread Loss:

$$L_i = (max(0, m - (a_t - a_i))^2, \quad L = \sum_{i \neq t} L_i$$

smallNORB classification

Table 1: The effect of varying different components of our capsules architecture on smallNORB.

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[1]	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%**

4.2M parameters
2.4M parameters
310K parameters

Investigating Capsule Networks with Dynamic Routing for Text Classification

Wei Zhao[1,2], Jianbo Ye[3], Min Yang[1*], Zeyang Lei[4], Soufei Zhang[5], Zhou Zhao[6]
[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[2] Tencent [3] Pennsylvania State University
[4] Graduate School at Shenzhen, Tsinghua University
[5] Nanjing University of Posts and Telecommunications [6] Zhejiang University

In EMNLP 2018

Text Classification

- To encode rich structures and spatial patterns in a sequence.

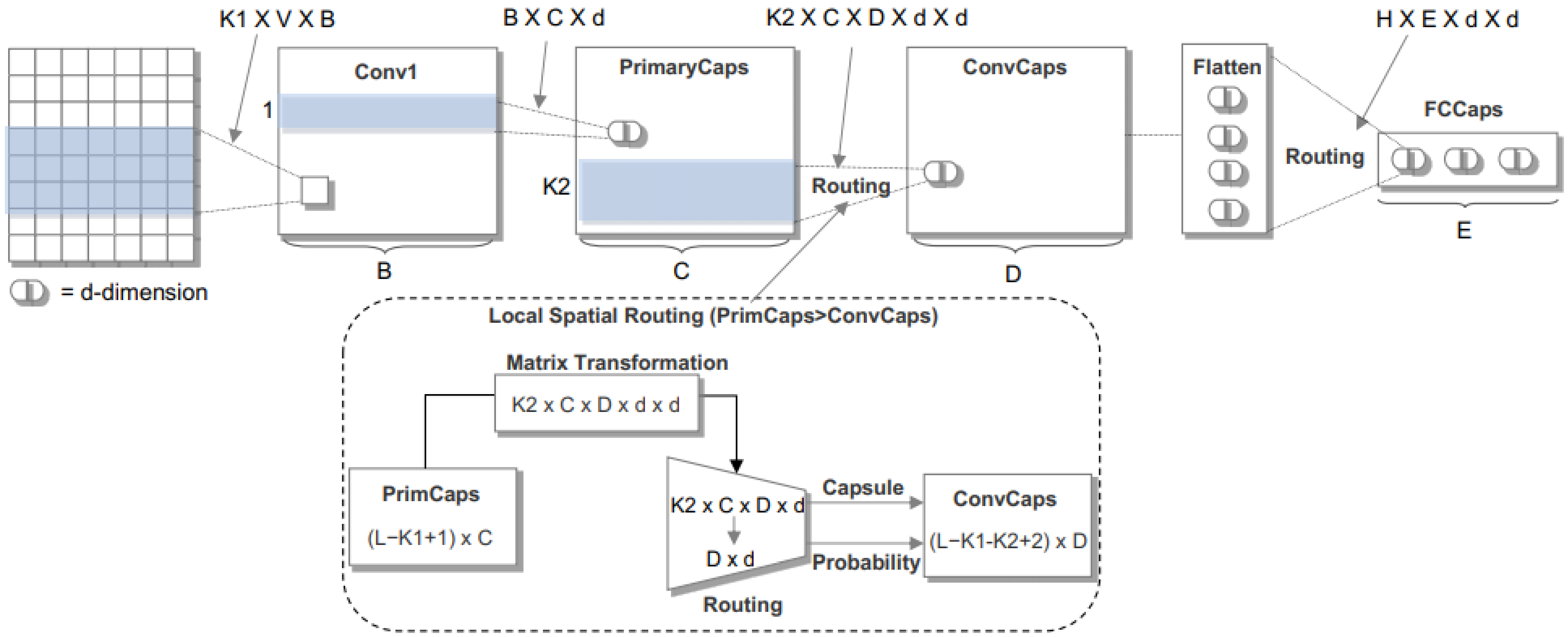


Figure 1: The Architecture of Capsule network for text classification. The processes of dynamic routing between consecutive layers are shown in the bottom.

Dynamic Routing

- Stabilize strategies:
  - An "orphan" category: background information such as stop words.
  - Replace Softmax as Leaky-Softmax.
  - Coefficients amendment.

Algorithm 1: Dynamic Routing Algorithm

1 **procedure** ROUTING($\hat{u}_{j|i}$, $\hat{a}_{j|i}$, r, l)
2 Initialize the logits of coupling coefficients $b_{j|i} = 0$
3 **for** r iterations **do**
4   for all capsule i in layer l and capsule j in layer $l + 1$: $c_{j|i} = \hat{a}_{j|i} \cdot \text{leaky-softmax}(b_{j|i})$
5   for all capsule j in layer $l + 1$: $v_j = g(\sum_i c_{j|i} \hat{u}_{j|i})$, $a_j = |v_j|$
6   for all capsule i in layer l and capsule j in layer $l + 1$: $b_{j|i} = b_{j|i} + \hat{u}_{j|i} \cdot v_j$
7 **return** v_j,a_j

Parallel Architectures

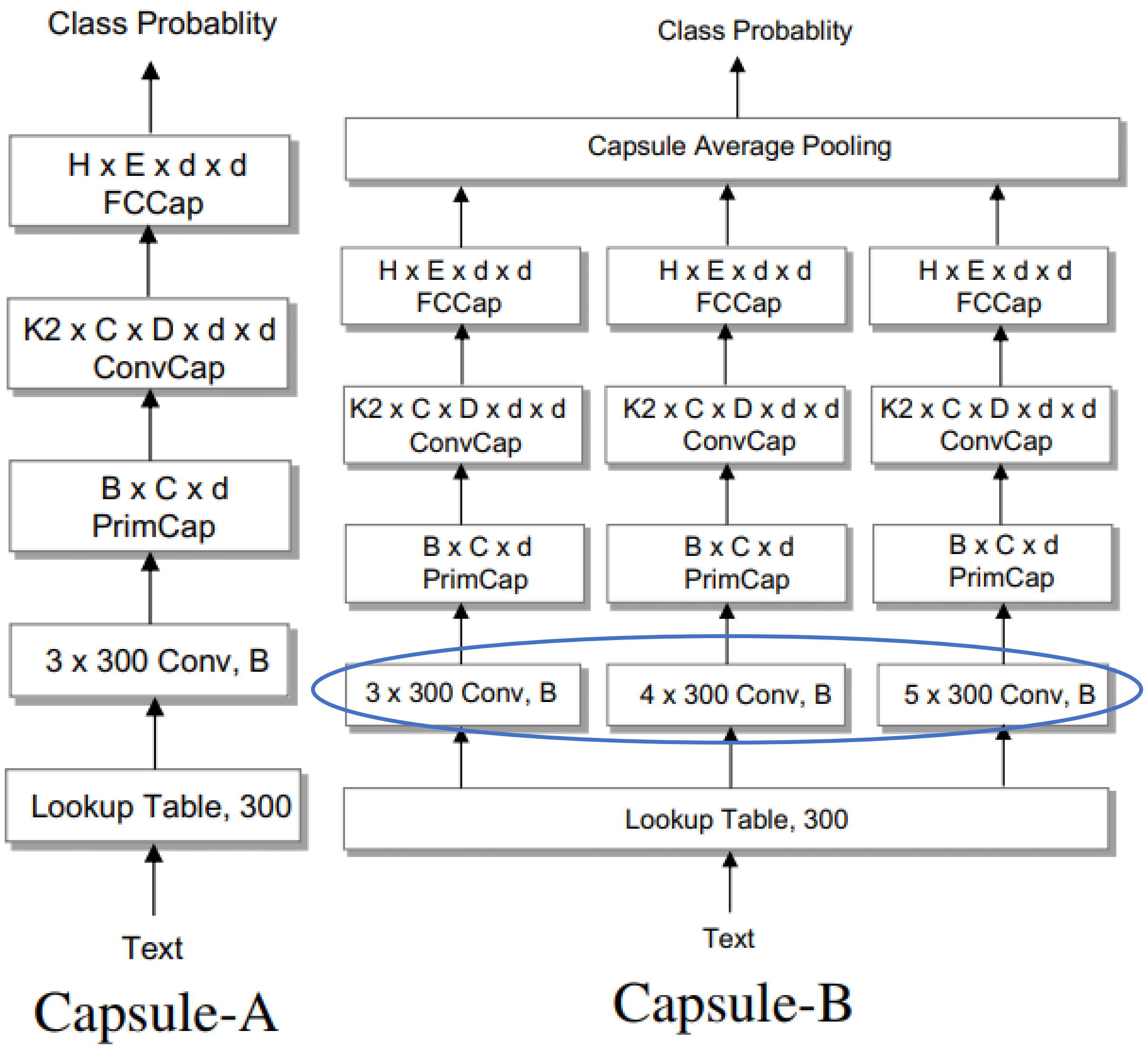


Figure 2: Two architectures of capsule networks.

Text Classification

	MR	SST2	Subj	TREC	CR	AG's
LSTM	75.9	80.6	89.3	86.8	78.4	86.1
BiLSTM	79.3	83.2	90.5	89.6	82.1	88.2
Tree-LSTM	80.7	85.7	91.3	91.8	83.2	90.1
LR-LSTM	81.5	**87.5**	89.9	-	82.5	-
CNN-rand	76.1	82.7	89.6	91.2	79.8	92.2
CNN-static	81.0	86.8	93.0	92.8	84.7	91.4
CNN-non-static	81.5	87.2	93.4	**93.6**	84.3	92.3
CL-CNN	-	-	88.4	85.7	-	92.3
VD-CNN	-	-	88.2	85.4	-	91.3
Capsule-A	81.3	86.4	93.3	91.8	83.8	92.1
Capsule-B	**82.3**	86.8	**93.8**	92.8	**85.1**	**92.6**

Table 2: Comparisons of our capsule networks and baselines on six text classification benchmarks.

Information Aggregation via Dynamic Routing for Sequence Encoding

Jingjing Gong, Xipeng Qiu*, Shaojing Wang, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
{jjgong15, xpqiu, sjwang17, xjhuang}@fudan.edu.cn

In COLING 2018

Sequence Encoding

- Embedding Layer

$$X = [\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L].$$

- Encoding Layer

$$\mathbf{h}_t^f = \mathrm{LSTM}(\mathbf{h}_{t-1}^f, \mathbf{x}_t),$$
$$\mathbf{h}_t^b = \mathrm{LSTM}(\mathbf{h}_{t+1}^b, \mathbf{x}_t),$$
$$\mathbf{h}_t = [\mathbf{h}_t^f; \mathbf{h}_t^b].$$
$$H = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L].$$

- Aggregation Layer

$$\mathbf{e}^{max} = \max([\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]),$$
$$\mathbf{e}^{avg} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{h}_i,$$

- Prediction Layer

$$\mathbf{p}(\cdot|\mathbf{e}) = \mathrm{softmax}(\mathrm{MLP}(\mathbf{e}))$$

Aggregation via Dynamic Routing

$$H = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L].$$

$$\Big\downarrow \text{dynamic routing}$$

$$V = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_M].$$

$$\Big\downarrow \text{concatenation}$$

$$\mathbf{e} = [\mathbf{v}_1; \ldots; \mathbf{v}_M].$$

$$\Big\downarrow$$

$$\mathbf{p}(\cdot|\mathbf{e}) = \mathrm{softmax}(\mathrm{MLP}(\mathbf{e}))$$

Text Classification

	Yelp-2013	Yelp-2014	IMDB	SST-1	SST-2
RNTN+Recurrent (Socher et al., 2013)	57.4	58.2	40.0	-	-
CNN-non-static (Kim, 2014)	-	-	-	48.0	87.2
Paragraph-Vec (Le and Mikolov, 2014)	-	-	-	48.7	**87.8**
MT-LSTM (F2S) (Liu et al., 2015)	-	-	-	49.1	87.2
UPNN(np UP) (Tang et al., 2015)	57.7	58.5	40.5	-	-
UPNN(full) (Tang et al., 2015)	59.6	60.8	43.5	-	-
Cached LSTM (Xu et al., 2016)	59.4	59.2	42.1	-	-
Max pooling	61.1	61.2	41.1	48.0	87.0
Average pooling	60.7	60.6	39.1	46.2	85.2
Self-attention	61.0	61.5	43.3	48.2	86.4
Standard DR-AGG	**62.1**	**63.0**	**45.1**	**50.5**	87.6
Reverse DR-AGG	61.6	62.5	44.5	49.3	87.2