DL Seminar Season #2

Paper review

한양대학교 AI Lab 석사 1기 유재창

Contents

논문의 목적 및 동기

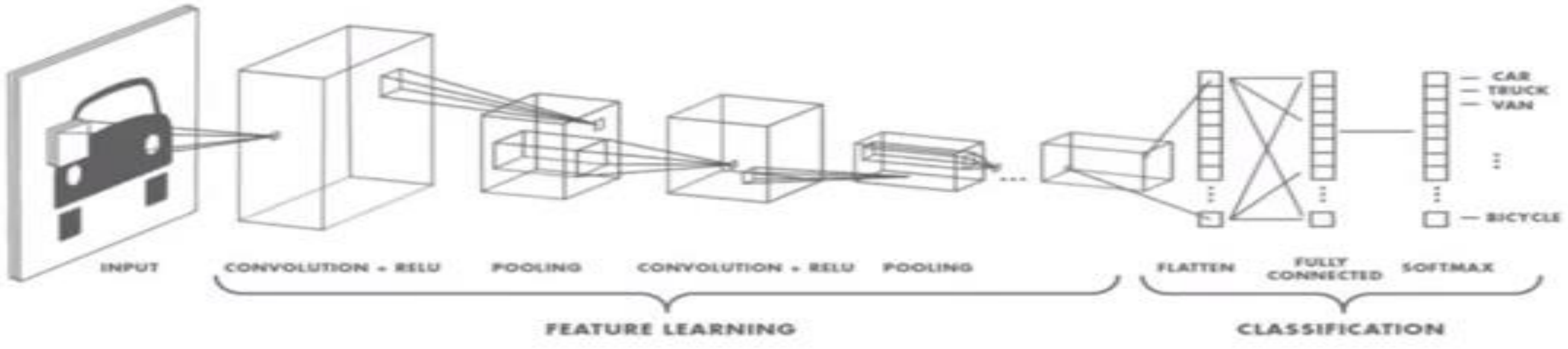
NIPS 2017 Paper

Dynamic Routing Between Capsules

by Sara Sabour, Nicholas Frosst, Geoffrey E. Hinton

Oct. 2017: https://arxiv.org/abs/1710.09829

논문의 목적 및 동기



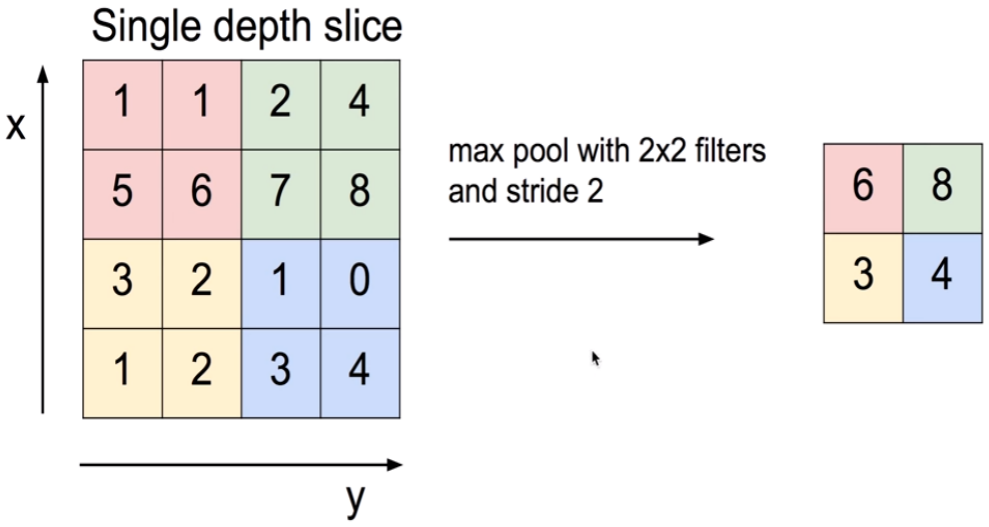
CNN의 문제점

- Feature를 학습할 때 Feature를 Detect 하는 것은 능숙
  하지만 Feature들 간의 위치 관계와 같은 Special relationship을 학습 하는데 문제.
- Image가 rotation, tilt등이 되면 성능이 저하.
- 이미지 픽셀 하나하나를 mapping out 시킨다는건 계산 복잡도 측면에서 매우 비 효율적.

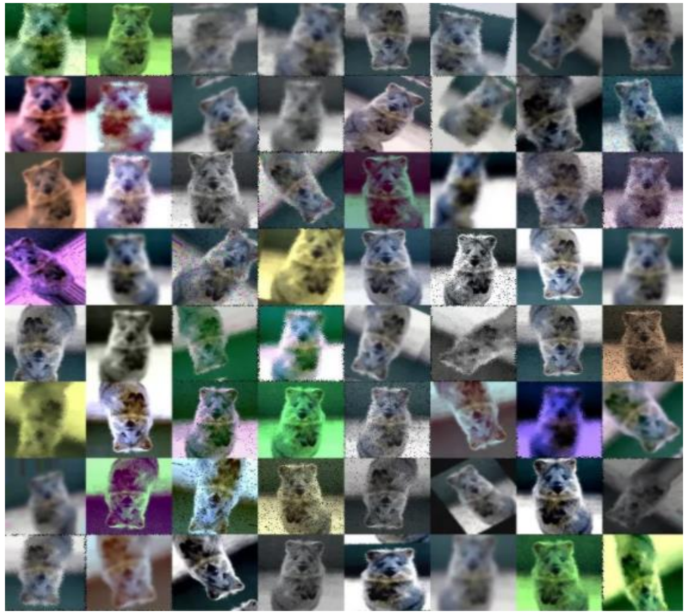
Dynamic routing between Capsules

논문의 목적 및 동기

1. Max Pooling



2. Data Augmentation



- Max Pooling과 Data Augmentation을 통해 어느정도 해결

논문의 목적 및 동기

Capsule의 개념이 등장하게 된 배경

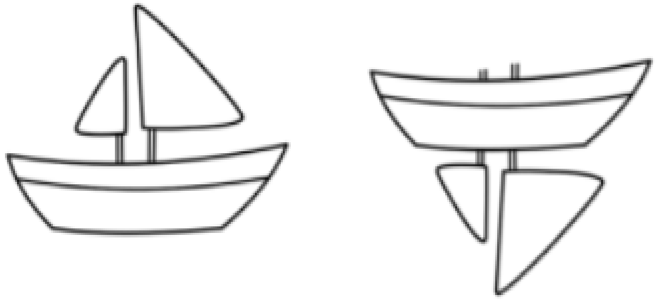
Pooling Layer의 문제점

- translation Invariance라는 특성을 부여하면서 추출된 변수의 차원을 줄이는 방법으로 CNN의 성능이 다른 구조에 비해 높은데 있어 크게 기여했지만 이 방법은 가장 큰 Activation만 선택하기 때문에 정보 손실이 큼.
- 이미지의 부분을 요약하여 표현하는 단계 하지만 이 과정이 반복될수록 기존에 픽셀이 가지고 있는 위치정보를 잃게 된다.
- Object detection이나 Segmentation과 같이 세밀한 정보가 필요한 경우 문제가 됨.

Dynamic routing between Capsules

논문의 목적 및 동기

CNN의 문제점



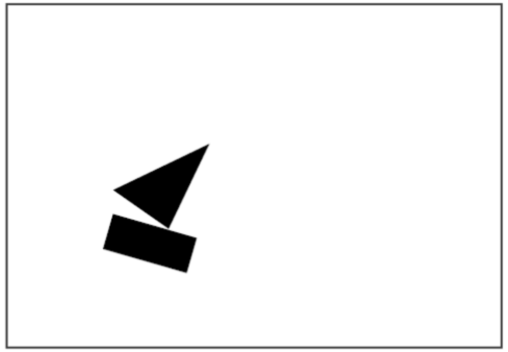
CNN의 경우 둘다 올바른 사람, 배로 분류 함.

Capsule Networks

Capsule이란?

Rectangle	**Triangle**
x=20 y=30 angle=16°	x=24 y=25 angle=-65°

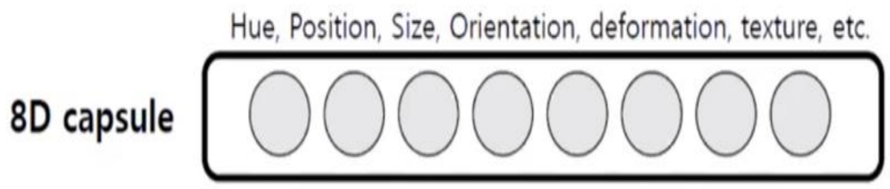
entity

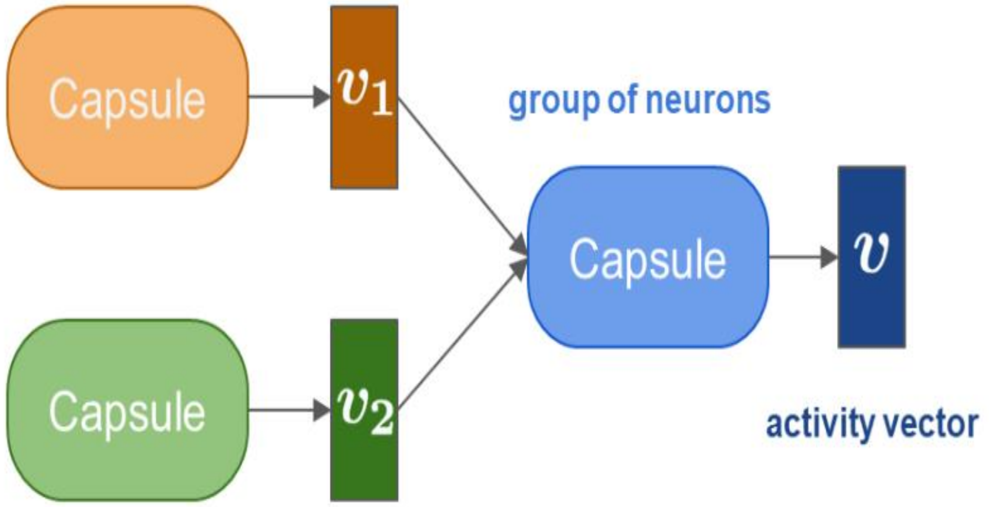


Image

Capsule Networks

Capsule이란?



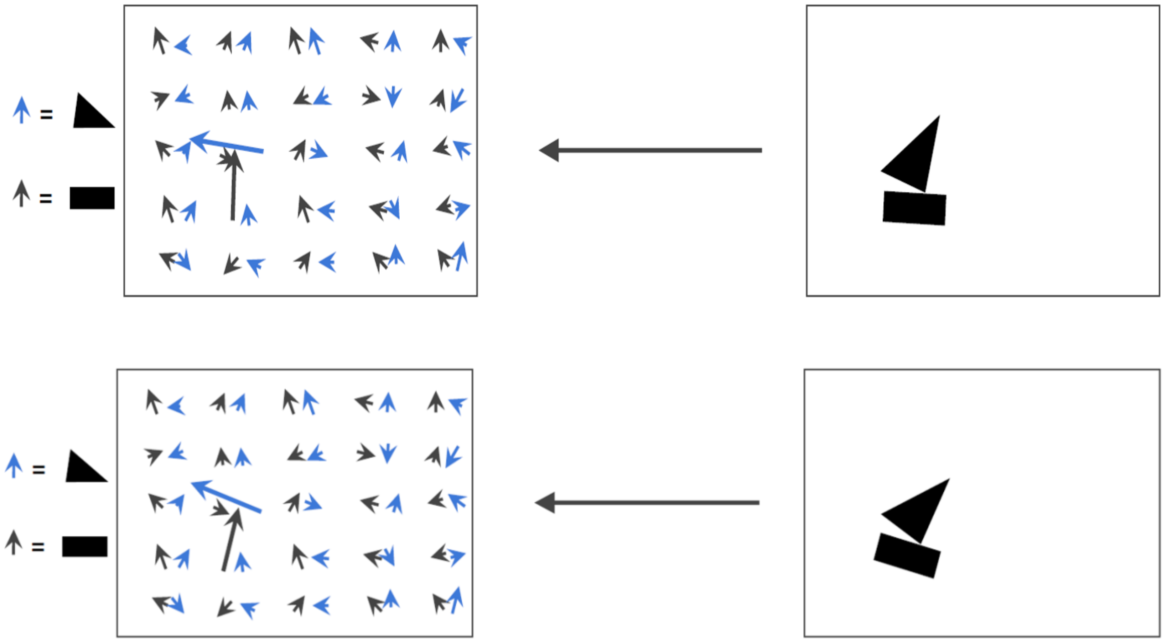


- 기존의 CNN의 기본 단위가 뉴런이라고 한다면 Capsule은 여러 개의 뉴런을 묶어 하나의 벡터를 입 출력 단위로 함.

Dynamic routing between Capsules

Capsule Networks

Capsule이란?



- 캡슐은 activity vector로 표현되고 그 vector의 길이로 The probability of entity exists를 나타낼 수 있음.

Capsule Networks

Routing Algorithm

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

$\mathbf{u}_i$ 는 현재 레이어의 i번째 캡슐의 output으로
prediction vector라고 함.
$\mathbf{W}_{ij}$는 weight matrix로 공간적인 relationship을 나타냄.

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}$$

c_{ij}는 routing algorithm에 의해 결정 되는
weight라고 생각 cij는 총합이 1.
Capsule과 Capsule 사이의 가중치.

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

$\hat{\mathbf{u}}_{j|i}$ 와 c_{ij} 를 weighted sum한 것으로
다음 layer의 캡슐에 input이 됨.

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

비선형성을 갖게 해주는 Squashing function.

Capsule Networks

Routing Algorithm

CapsNet에서 학습과정에 필요한 알고리즘

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 $\mathbf{v}_j$

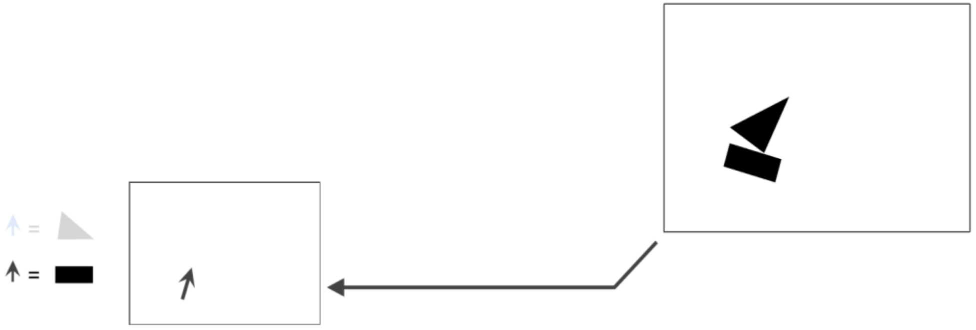
r = routing알고리즘 시행 횟수.
l = primary capsules의 layer.

Capsule Networks

Routing Algorithm

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: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ 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 $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ 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 $\mathbf{v}_j$



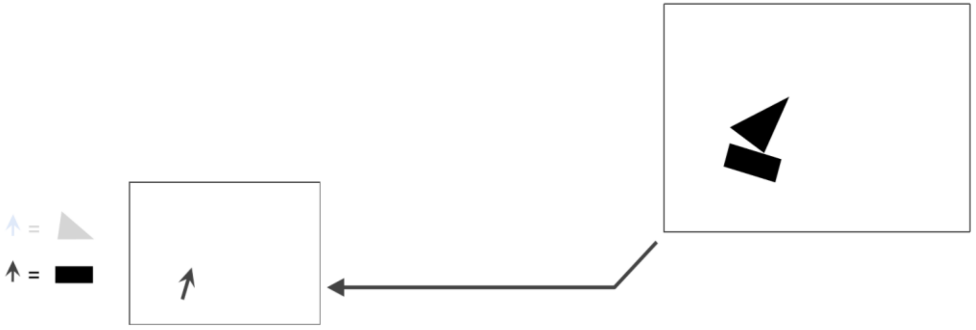
Dynamic routing between Capsules

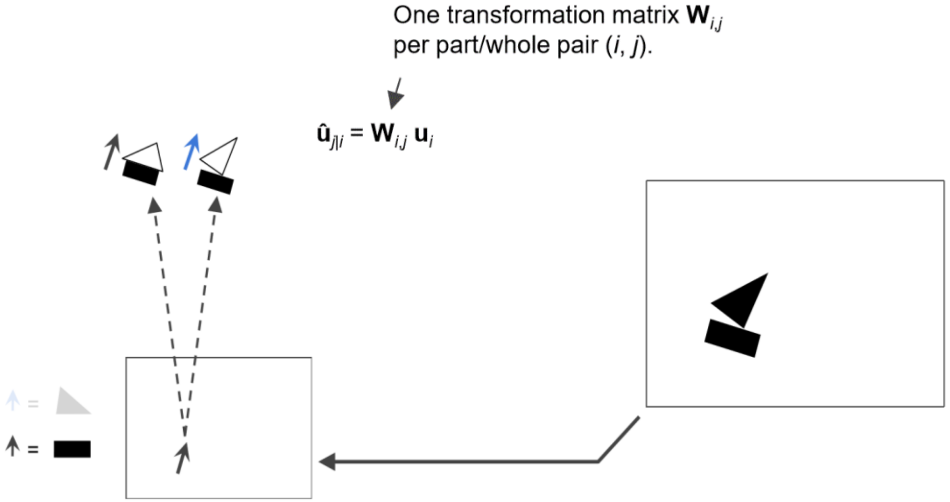
Capsule Networks

Routing Algorithm

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 $\mathbf{v}_j$





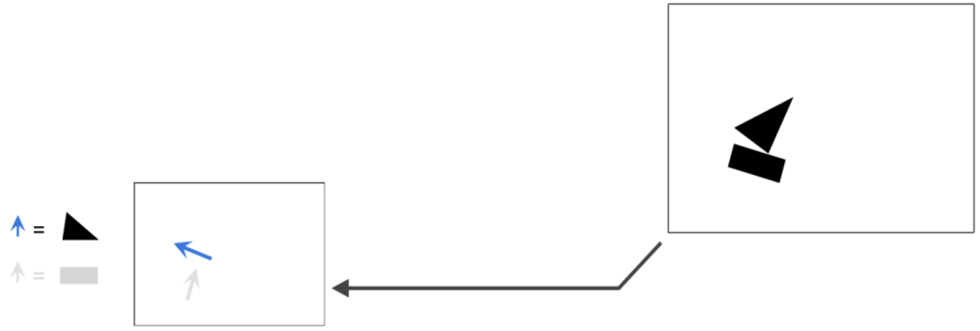
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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 $\mathbf{v}_j$



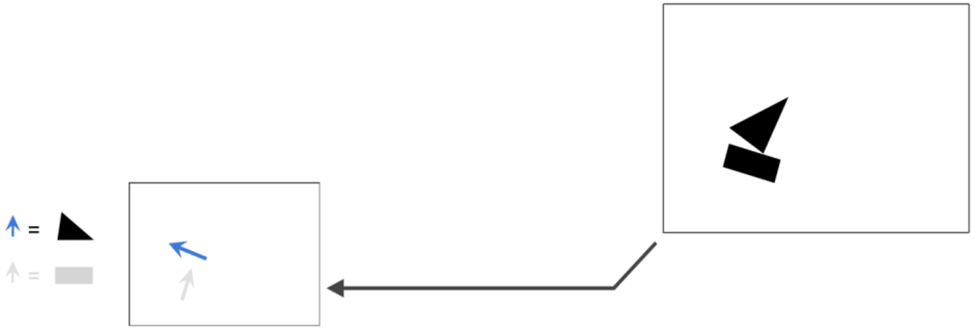
Dynamic routing between Capsules

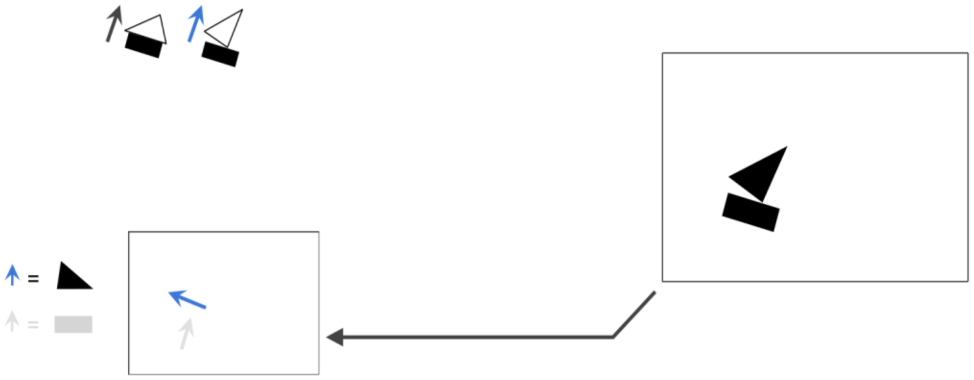
Capsule Networks

Routing Algorithm

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 $\mathbf{v}_j$





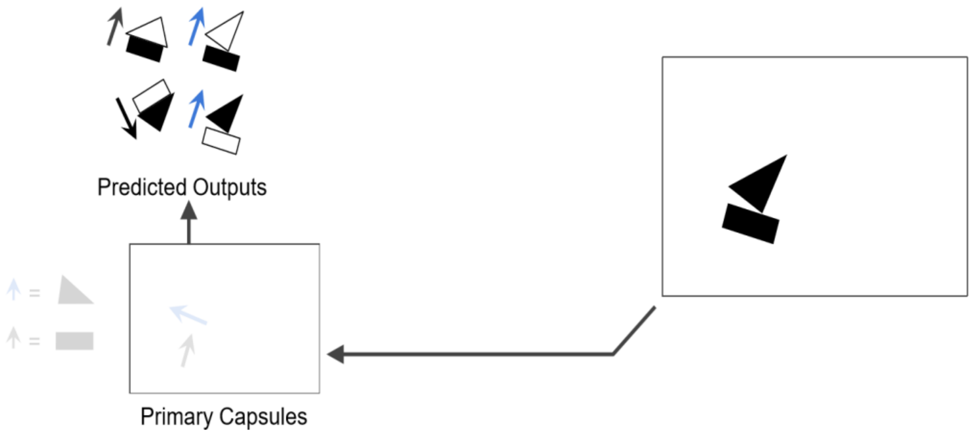
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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 $\mathbf{v}_j$



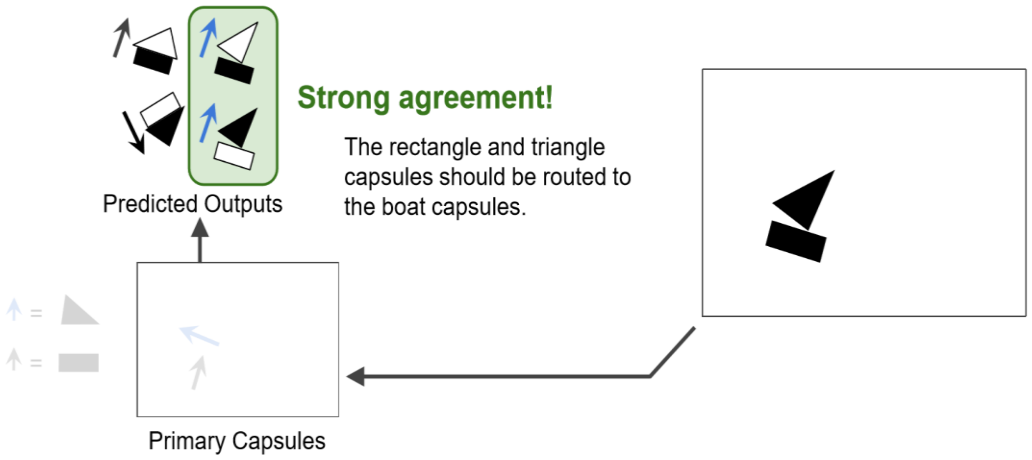
- l 레이어에서 각 캡슐들의 prediction vector들이 나옴.

Capsule Networks

Routing Algorithm

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)                    ▷ softmax computes Eq. 3
        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)               ▷ squash computes Eq. 1
        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
```



- 두 가지의 prediction vector가 비슷하므로 strong한 agreement를 갖음.

Dynamic routing between Capsules

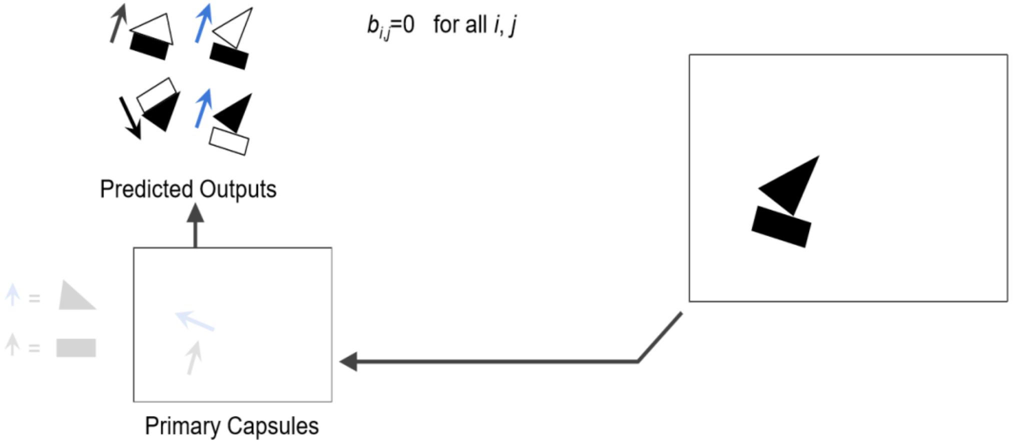
Capsule Networks

Routing Algorithm

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 $\mathbf{v}_j$



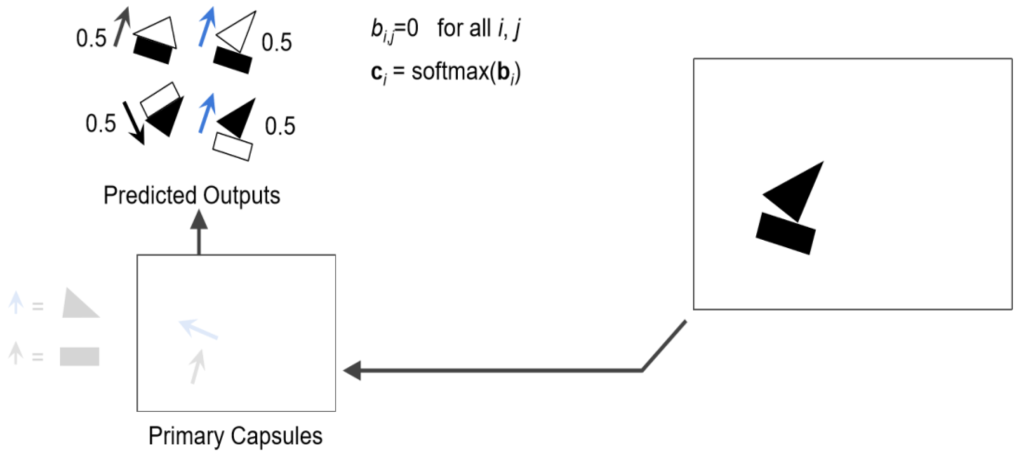
- 레이어 간 Capsule들의 초기 가중치를 구하는 단계.

Capsule Networks

Routing Algorithm

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: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ 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 $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ 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 $\mathbf{v}_j$



- 초기 bij는 0 으로 초기화 했으므로 Softmax를 취하면 초기 캡슐들 끼리 연결된 weight값은 0.5로 동일.

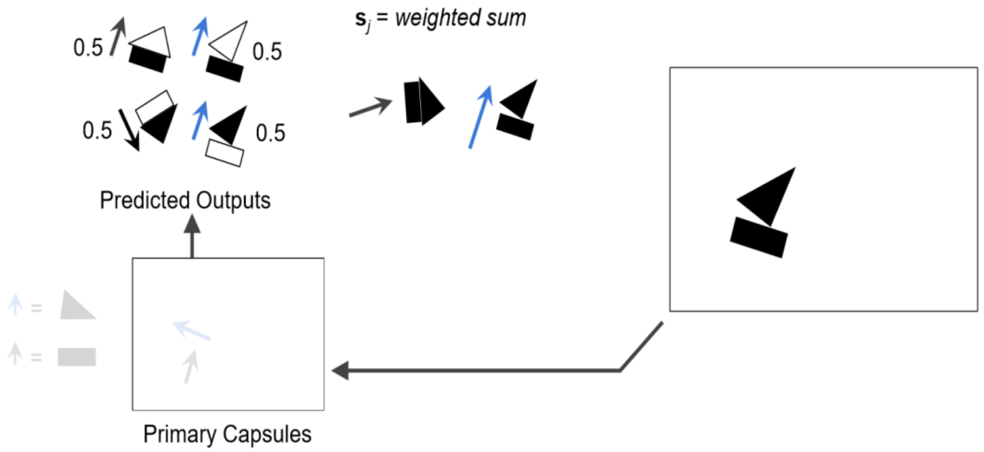
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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)            ▷ softmax computes Eq. 3
        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)        ▷ squash computes Eq. 1
        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
```



- 다음 레이어의 각 Capsule들의 input인 s1, s2을 구한 가중치와 prediction vector사이의 weighted sum을 통해 구함.

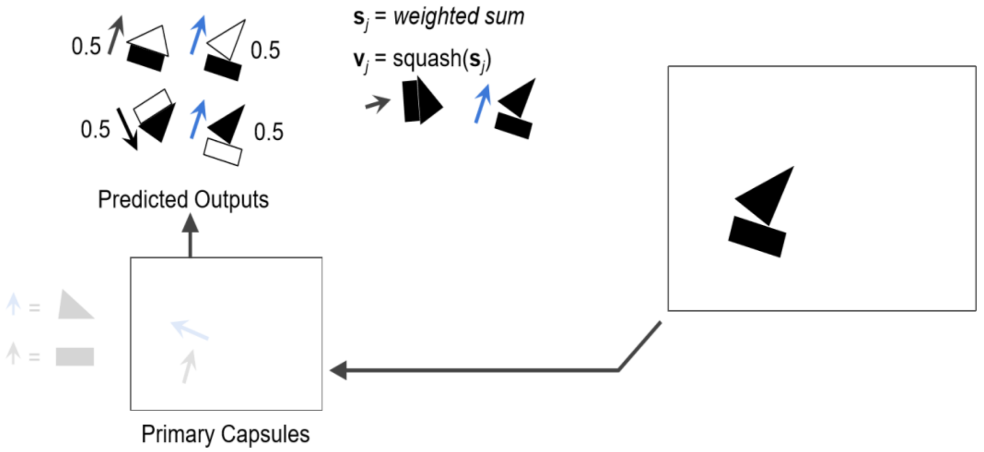
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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 $\mathbf{v}_j$



- 비선형성을 더하기 위해 squashing function을 이용해줌.
- 이전과 비교해보면 벡터의 크기가 줄어든 것을 확인 할 수 있음 0~1사이.

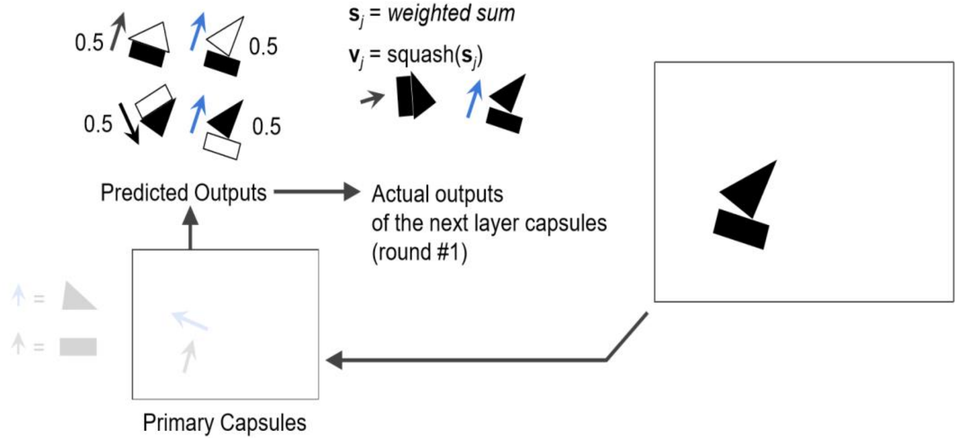
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ 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 $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ 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 $\mathbf{v}_j$



- 다음 레이어의 Capsule들의 output activation vector들이 구해짐.

Dynamic routing between Capsules

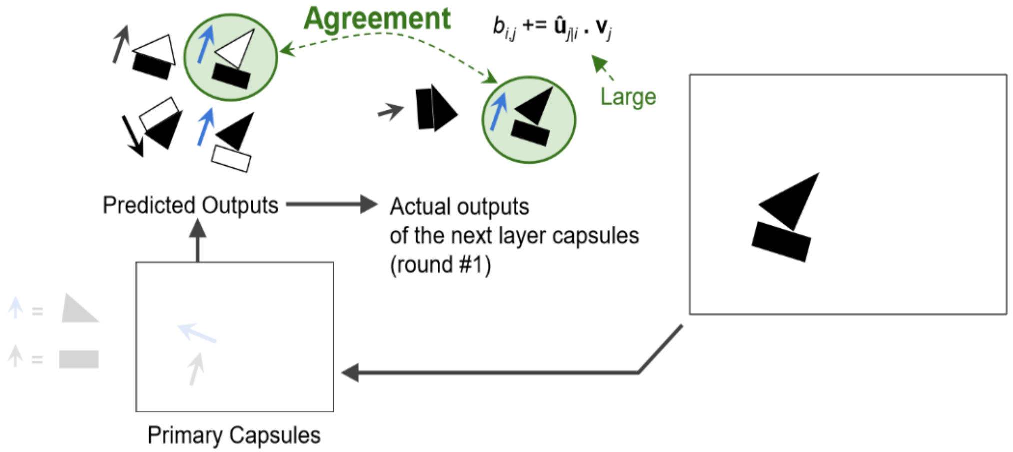
Capsule Networks

Routing Algorithm

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 $\mathbf{v}_j$



- 1번의 routing 알고리즘이 마무리 되는 단계에서 가중치 bij를 업데이트 해주는 부분.

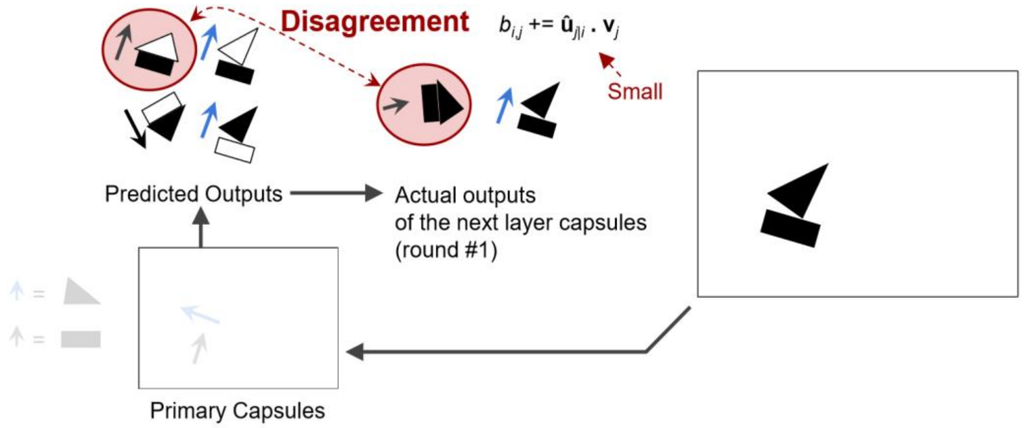
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ 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 $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ 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 $\mathbf{v}_j$



skip

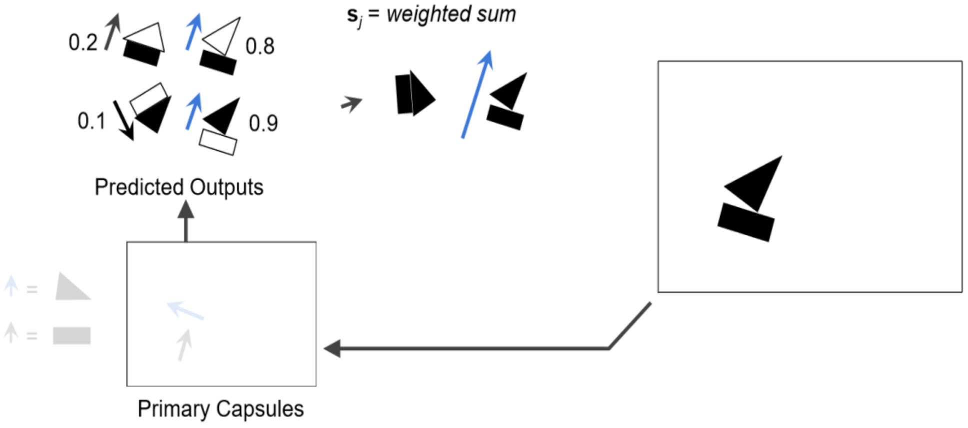
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ 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 $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ 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 $\mathbf{v}_j$



skip

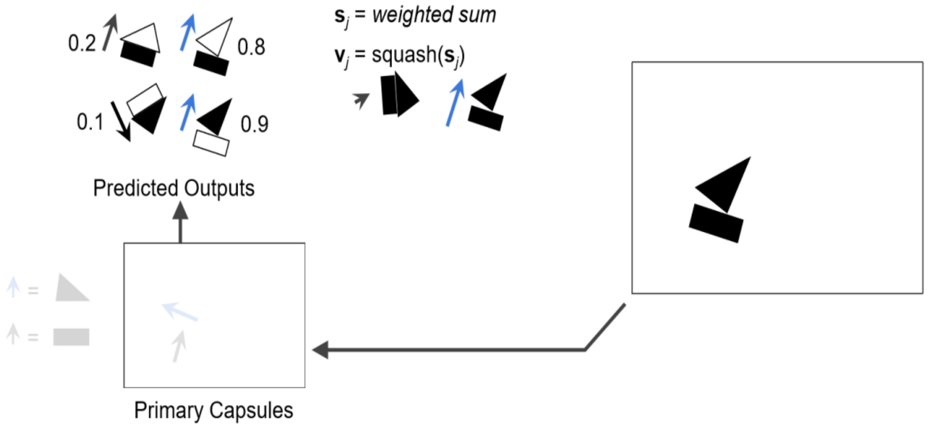
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ 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 $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ 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 $\mathbf{v}_j$



skip

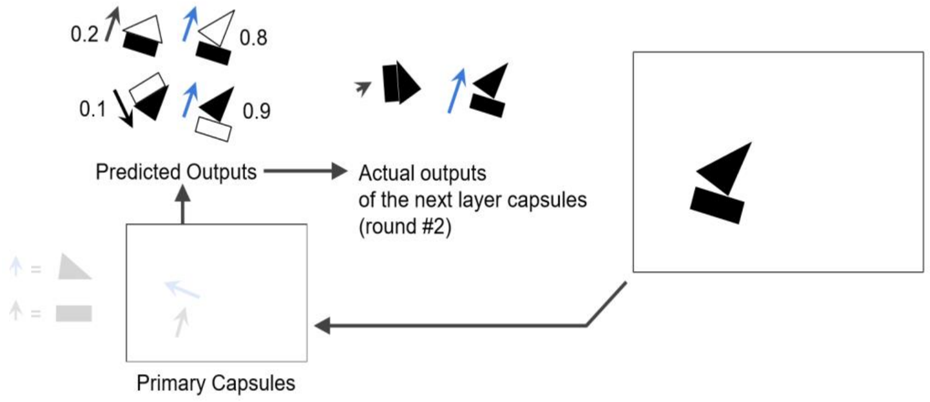
Dynamic routing between Capsules

Capsule Networks

Routing Algorithm

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 $\mathbf{v}_j$

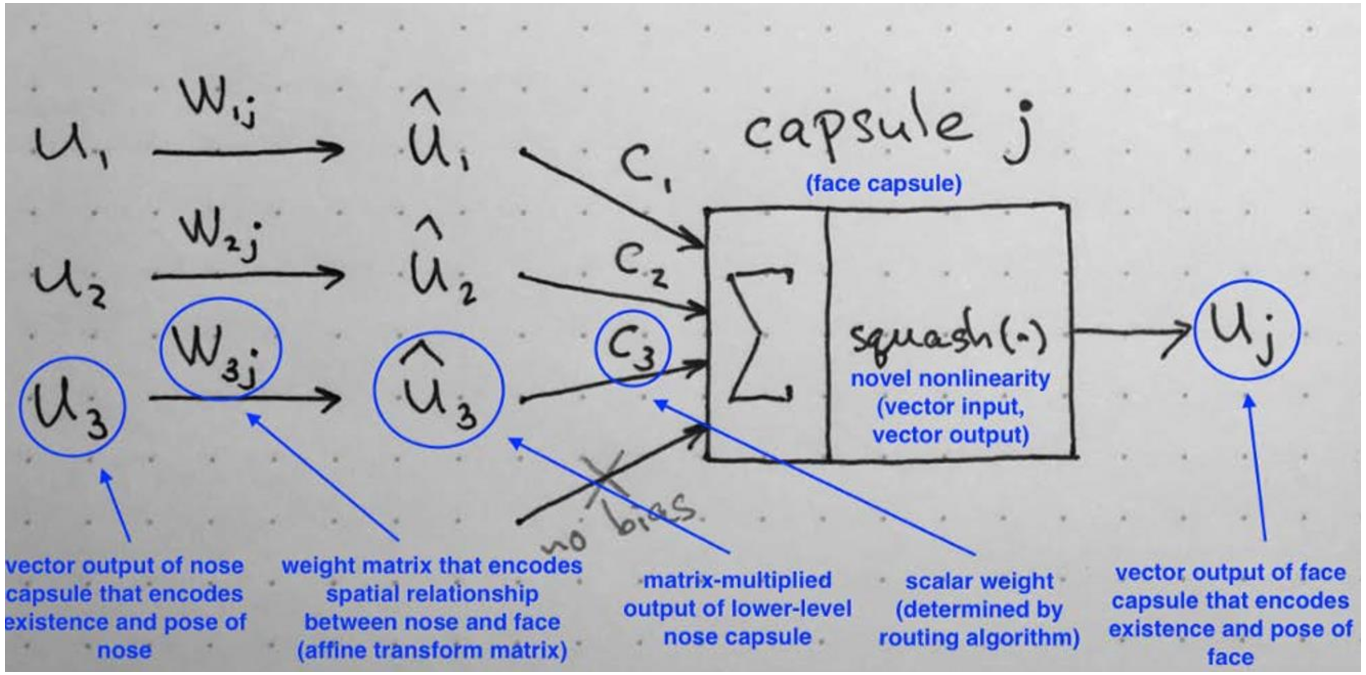


skip

Dynamic routing between Capsules

Capsule Networks

Ex) 사람 얼굴



Dynamic routing between Capsules

Capsule Networks

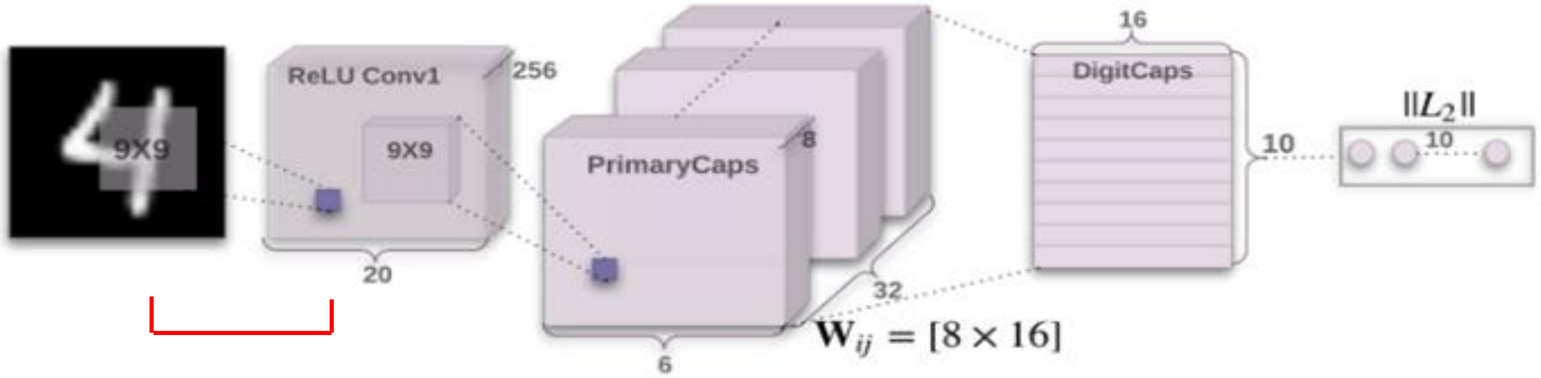
기존의 신경망과 CapsNet의 비교

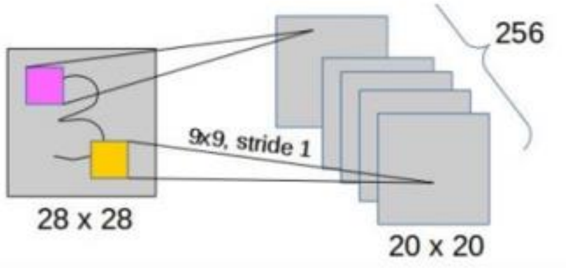
Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\hat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear 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)

Dynamic routing between Capsules

Capsule Networks

Caps Net 구조



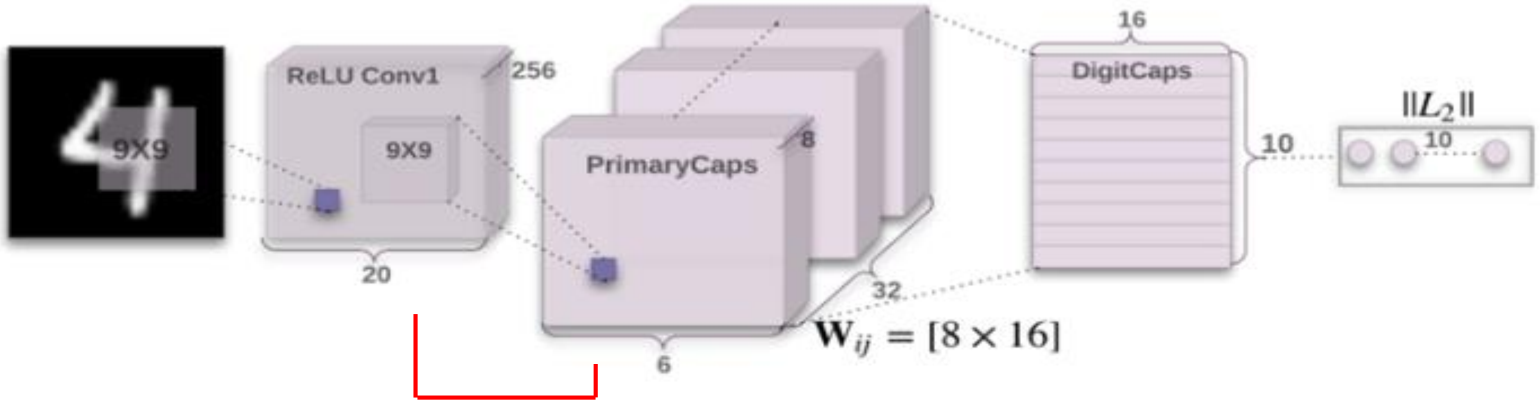


1. 첫 번째 Layer는 usual한 convolutional layer

- 28x28의 image를 9x9의 256개의 filter와 stride 1로 feature map을 만듦.
- 20x20x256 형태의 feature map이 나옴.

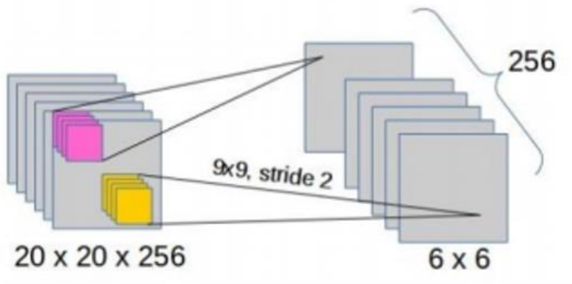
Capsule Networks

Caps Net 구조



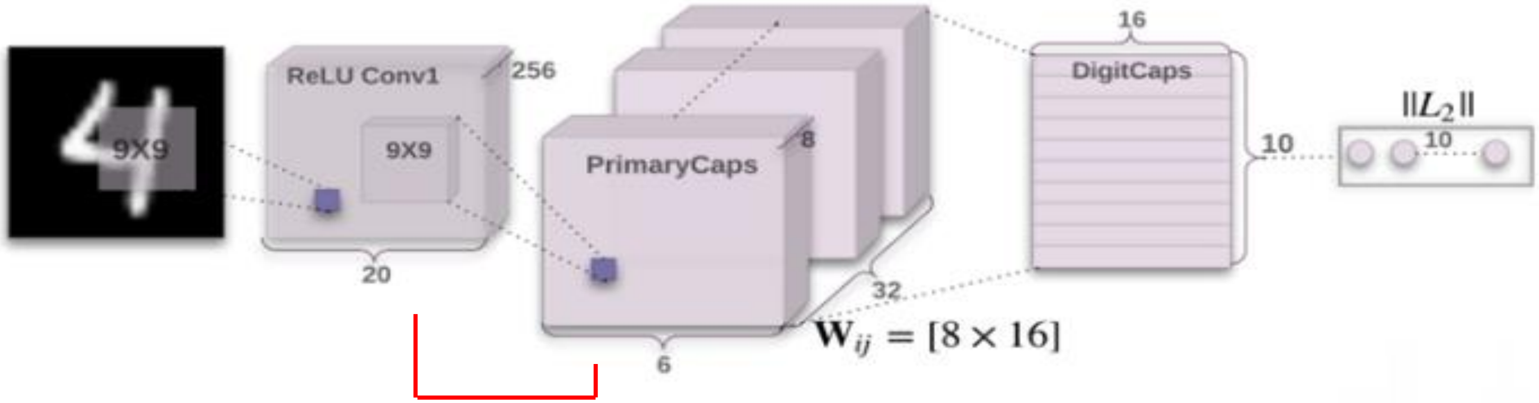
2. 두 번째 Layer는 PrimaryCapsule layer

- Capsule을 만드는 layer로 9x9의 filter 1개와 stride는 2로 6x6x8x32의 feature map을 만듦.



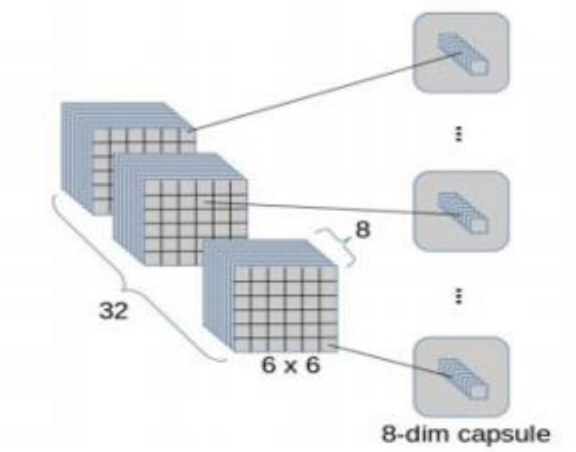
Capsule Networks

Caps Net 구조



2. 두 번째 Layer는 PrimaryCapsule layer

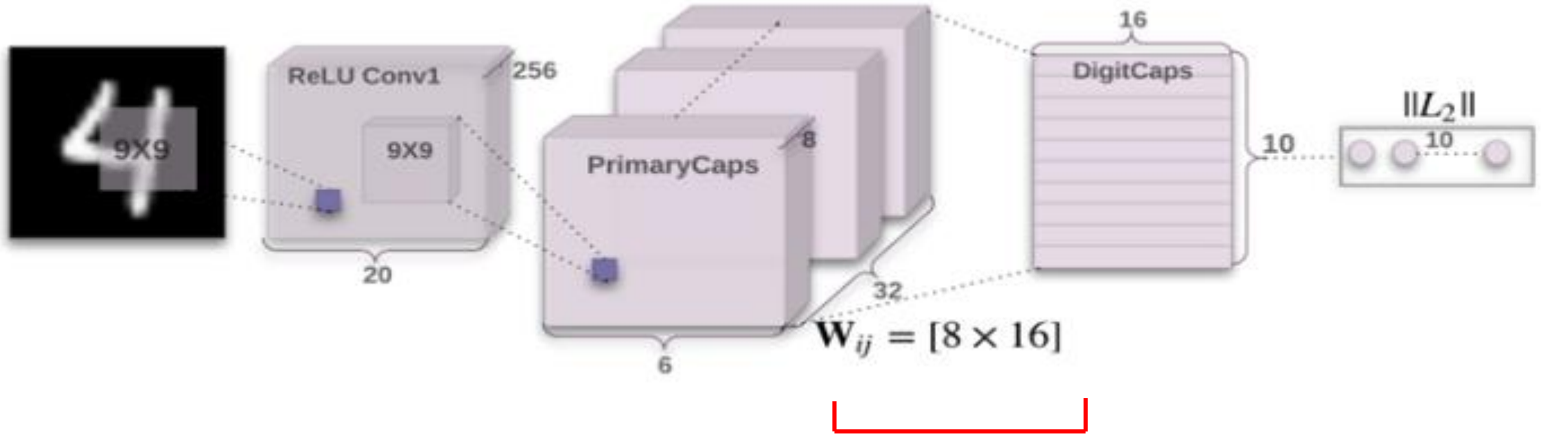
- 6x6x8x32의 feature map을 통해 1x1x8의 8D Capsule이 6x6x32 = 1152개가 만들어짐.



Dynamic routing between Capsules

Capsule Networks

Caps Net 구조



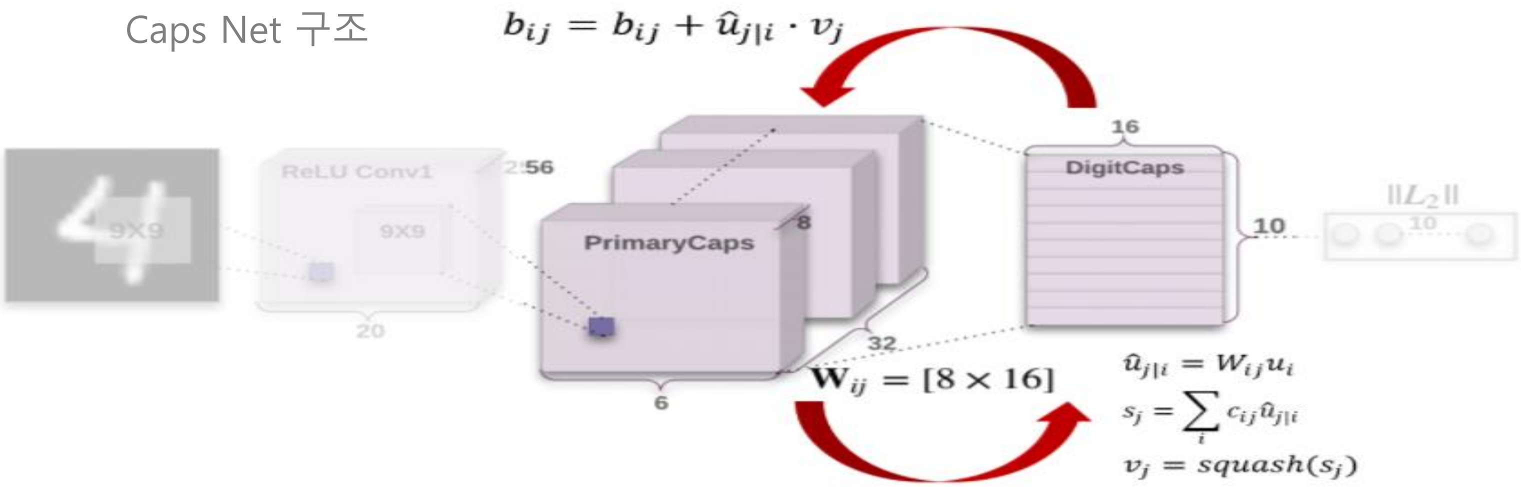
3. Capsule to Capsule layer 또는 Digit Capsule layer

- 6x6x32의 lower level capsules들이 10 higher level capsules과 연결하기 위해 총 1152 x 10의 weight matrices Wij가 필요함.
- 마지막 10개의 higher level capsules이 10개의 digit/class entity를 나타냄.

Dynamic routing between Capsules

Capsule Networks

Caps Net 구조



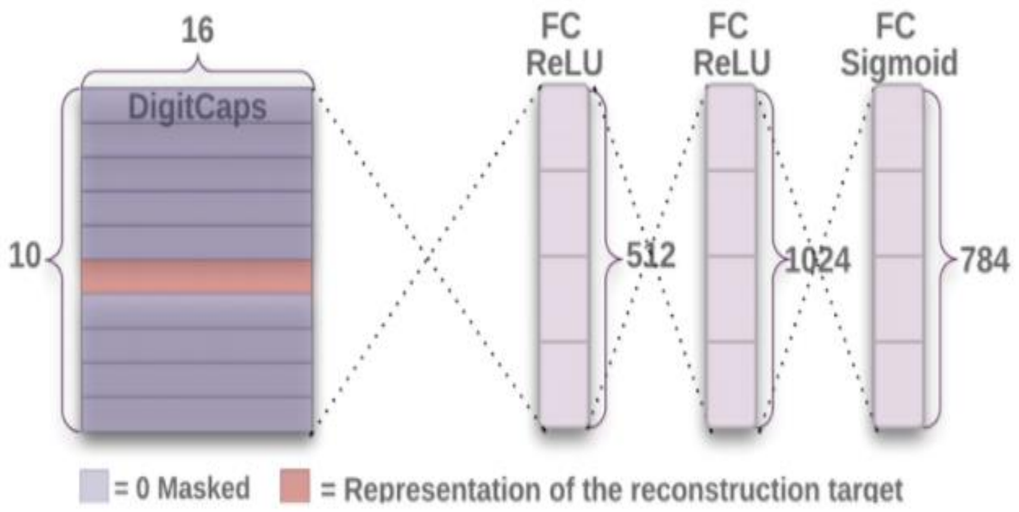
Routing 알고리즘이 적용되는 부분

Procedure 1 Routing algorithm.

procedure ROUTING($\hat{\mathbf{u}}_{j|i}$, 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 $\mathbf{v}_j$

Capsule Networks

Digit/class Reconstruction 과정



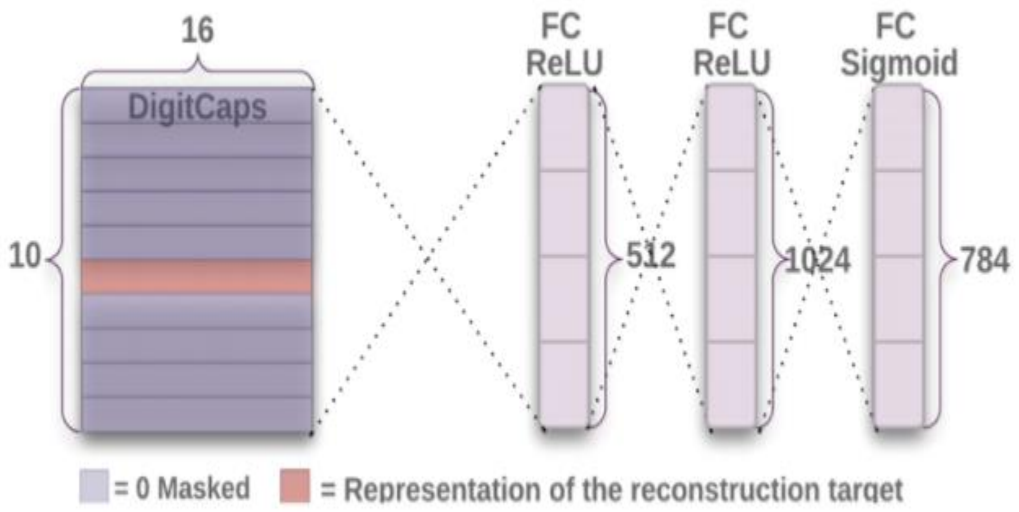
- Digit Caps를 완성하면, 이 벡터를 가지고 다시 digit을 원복 할 수 있습니다. **가장 큰 크기를 가지는 캡슐의 벡터** 원소 16개를 각각 512, 1024, 784 까지의 fully connected layer에 연결함.

- 마지막 단은 Sigmoid를 거쳐서 0과 1의 값을 가지게 하고 28 x 28로 바꾸면 reconstruct한 digit을 확인할 수 있음.

Dynamic routing between Capsules

Capsule Networks

Digit/class Reconstruction 과정



- Digit Caps를 완성하면, 이 벡터를 가지고 다시 digit을 원복 할 수 있습니다. **가장 큰 크기를 가지는 캡슐의 벡터** 원소 16개를 각각 512, 1024, 784 까지의 fully connected layer에 연결함.
- 마지막 단은 Sigmoid를 거쳐서 0과 1의 값을 가지게 하고 28 x 28로 바꾸면 reconstruct한 digit을 확인할 수 있음.

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

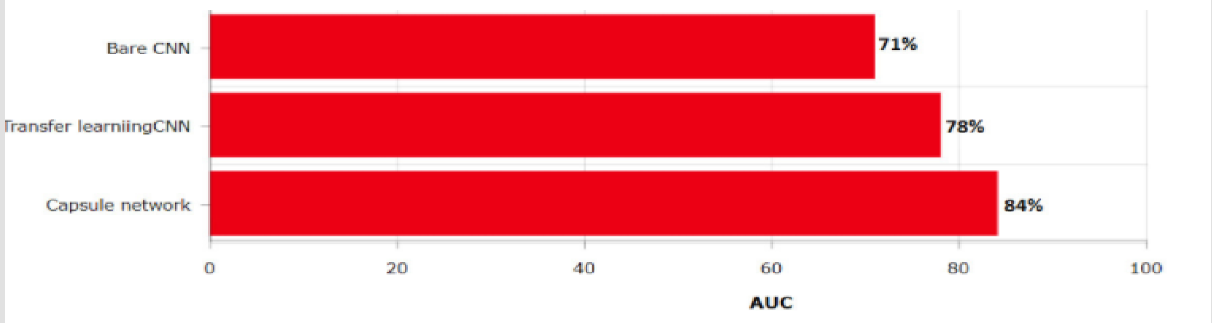
zero loss when incorrect prediction with probability less than 0.1, non-zero otherwise

Dynamic routing between Capsules

Contribution

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**

Dynamic routing 횟수와 reconstruction loss 전파 유무를 가지고 Caps Net을 학습시킨 테스트 결과1(MNIST)



Caps Net을 학습시킨 테스트 결과2 (lung cancer 400 images)

Dynamic routing between Capsules

끝