Dynamic Routing Between Capsules

by S. Sabour, N. Frosst and G. Hinton (NIPS 2017)

---

presented by Karel Ha

27th March 2018

Pattern Recognition and Computer Vision Reading Group

Outline

Motivation

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

G. Hinton

http://condo.ca/wp-content/uploads/2017/03/Vector-director-Institute-artificial-intelligence-Toronto-MaRS-Discovery-District-H
ca_.jpg

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

G. Hinton

> [...] it makes much more sense to represent a pose as a small matrix that converts a vector of positional coordinates relative to the viewer into positional coordinates relative to the shape itself.

G. Hinton

http://condo.ca/wp-content/uploads/2017/03/
Vector-director-Institute-artificial-intelligence-Toronto-MaRS-Discovery-District-H
ca_.jpg

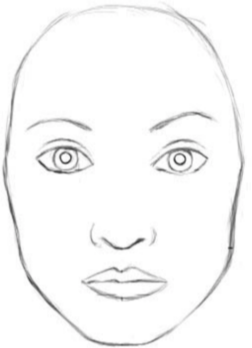
Part-Whole Geometric Relationships

"What is wrong with convolutional neural nets?"

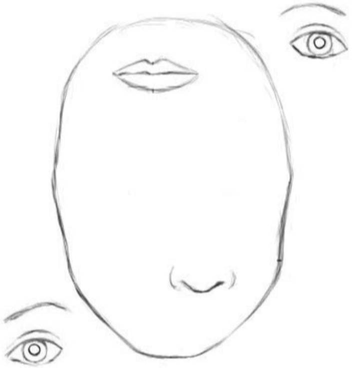
https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Part-Whole Geometric Relationships

"What is wrong with convolutional neural nets?"



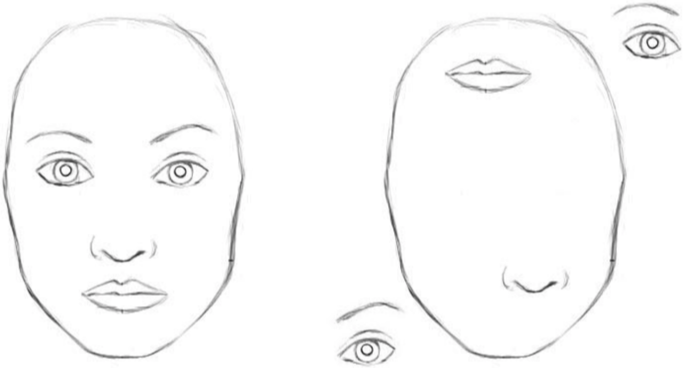
To a CNN (with MaxPool)...



https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Part-Whole Geometric Relationships

“What is wrong with convolutional neural nets?”



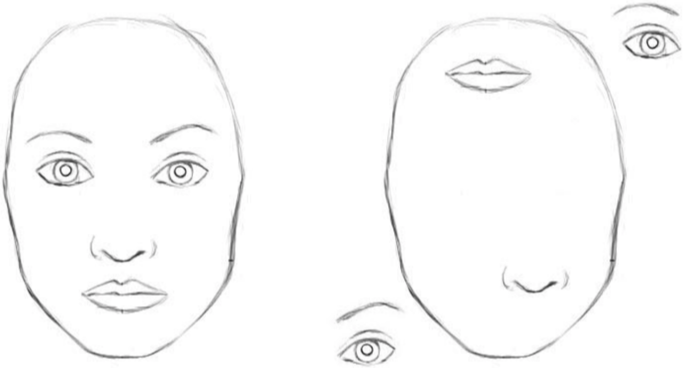
To a CNN (with MaxPool)...

- ...both pictures are similar, since they both contain similar elements.

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Part-Whole Geometric Relationships

"What is wrong with convolutional neural nets?"



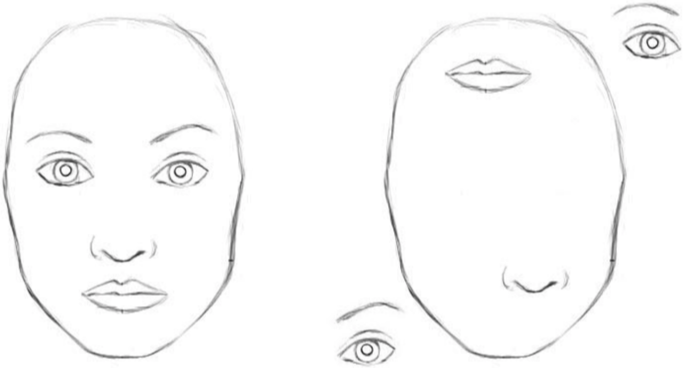
To a CNN (with MaxPool)...

- ...both pictures are similar, since they both contain similar elements.
- ...a mere presence of objects can be a very **strong indicator** to consider a face in the image.

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Part-Whole Geometric Relationships

"What is wrong with convolutional neural nets?"



To a CNN (with MaxPool)...

- ...both pictures are similar, since they both contain similar elements.
- ...a mere presence of objects can be a very **strong indicator** to consider a face in the image.
- ...orientational and relative spatial relationships are not very important.

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Part-Whole Geometric Relationships

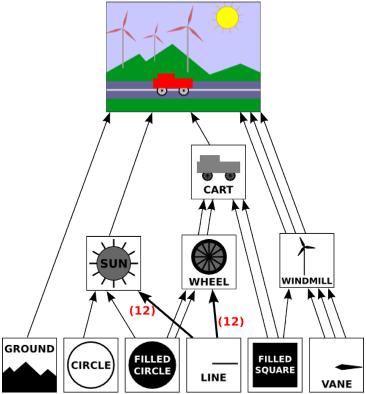
Scene Graphs from Computer Graphics

http://math.hws.edu/graphicsbook/c2/scene-graph.png

Part-Whole Geometric Relationships

Scene Graphs from Computer Graphics

...takes into account relative positions of objects.

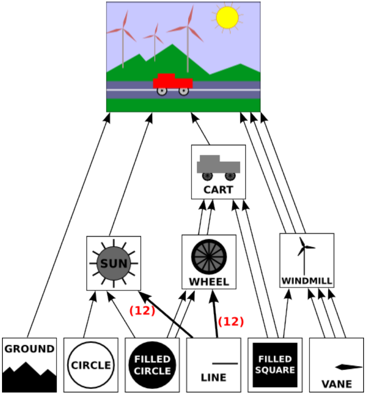


http://math.hws.edu/graphicsbook/c2/scene-graph.png

Part-Whole Geometric Relationships

Scene Graphs from Computer Graphics

...takes into account relative positions of objects.



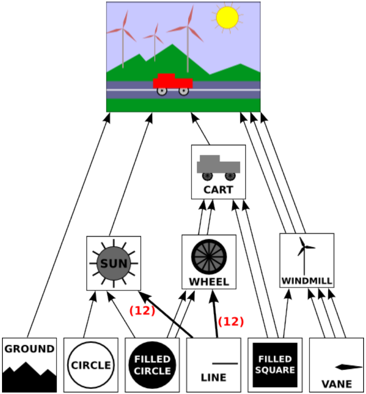
The internal representation in computer memory:

http://math.hws.edu/graphicsbook/c2/scene-graph.png

Part-Whole Geometric Relationships

Scene Graphs from Computer Graphics

...takes into account relative positions of objects.



The internal representation in computer memory:

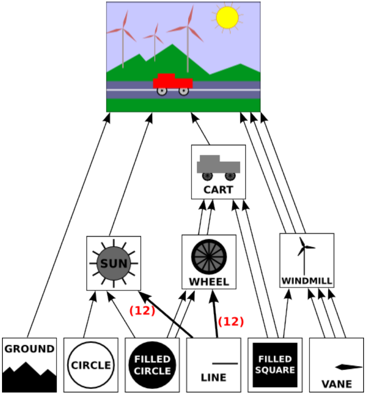
a) arrays of geometrical objects

http://math.hws.edu/graphicsbook/c2/scene-graph.png

Part-Whole Geometric Relationships

Scene Graphs from Computer Graphics

...takes into account relative positions of objects.



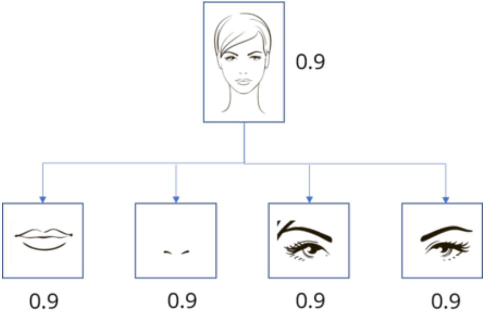
The internal representation in computer memory:

a) arrays of geometrical objects
b) matrices representing their **relative positions** and **orientations**

http://math.hws.edu/graphicsbook/c2/scene-graph.png

Part-Whole Geometric Relationships

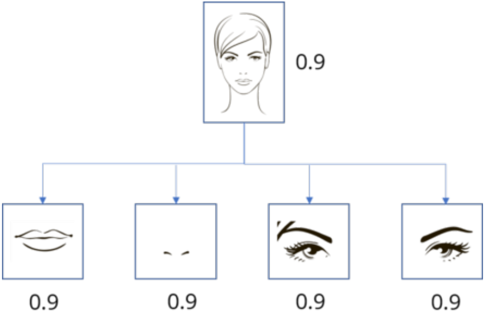
Inverse (Computer) Graphics



https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2017/12/CNN-Capsule-Networks-Edureka-442x300.png

Part-Whole Geometric Relationships

Inverse (Computer) Graphics

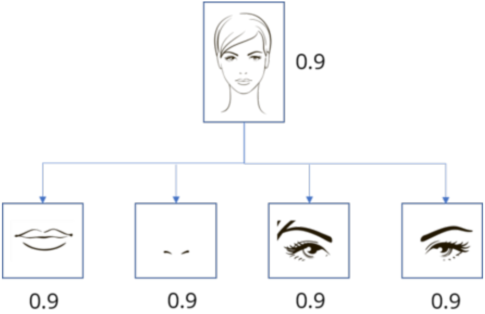


Inverse graphics:

https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2017/12/CNN-Capsule-Networks-Edureka-442x300.png

Part-Whole Geometric Relationships

Inverse (Computer) Graphics



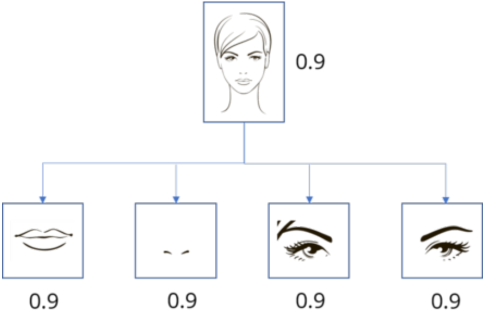
Inverse graphics:

- from visual information received by eyes

https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2017/12/CNN-Capsule-Networks-Edureka-442x300.png

Part-Whole Geometric Relationships

Inverse (Computer) Graphics



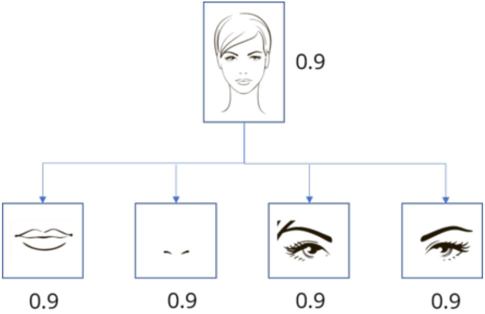
Inverse graphics:

- from visual information received by eyes
- deconstruct a hierarchical representation of the world around us

https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2017/12/CNN-Capsule-Networks-Edureka-442x300.png

Part-Whole Geometric Relationships

Inverse (Computer) Graphics



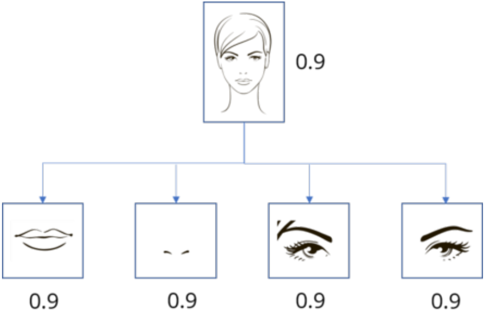
Inverse graphics:

- from visual information received by eyes
- deconstruct a hierarchical representation of the world around us
- and try to match it with already learned patterns and relationships stored in the brain

https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2017/12/CNN-Capsule-Networks-Edureka-442x300.png

Part-Whole Geometric Relationships

Inverse (Computer) Graphics



Inverse graphics:

- from visual information received by eyes
- deconstruct a hierarchical representation of the world around us
- and try to match it with already learned patterns and relationships stored in the brain
- relationships between 3D objects using a **"pose"** (= translation plus rotation)

https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2017/12/CNN-Capsule-Networks-Edureka-442x300.png

Pose Equivariance and the Viewing Angle

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Pose Equivariance and the Viewing Angle

We have probably never seen these exact pictures, but we can still immediately recognize the object in it...

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Pose Equivariance and the Viewing Angle

We have probably never seen these exact pictures, but we can still immediately recognize the object in it...

- internal representation in the brain: independent of the viewing angle

Pose Equivariance and the Viewing Angle

We have probably never seen these exact pictures, but we can still immediately recognize the object in it...

- internal representation in the brain: independent of the viewing angle
- quite hard for a CNN: no built-in understanding of 3D space

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Pose Equivariance and the Viewing Angle

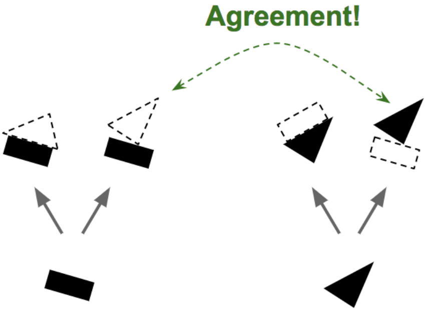
We have probably never seen these exact pictures, but we can still immediately recognize the object in it...

- internal representation in the brain: independent of the viewing angle
- quite hard for a CNN: no built-in understanding of 3D space
- much easier for a CapsNet: these relationships are explicitly modeled

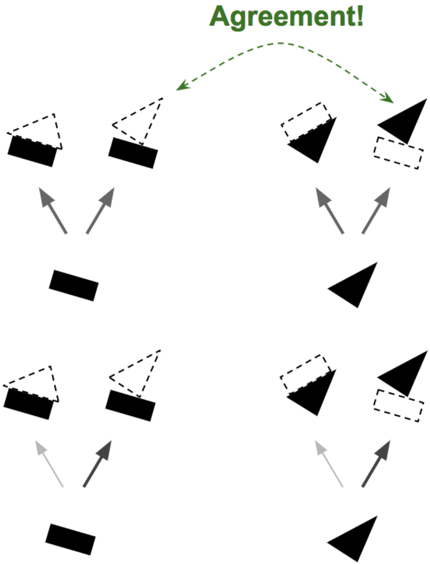
https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Routing by an Agreement: High-Dimensional Coincidence

Routing by an Agreement: High-Dimensional Coincidence



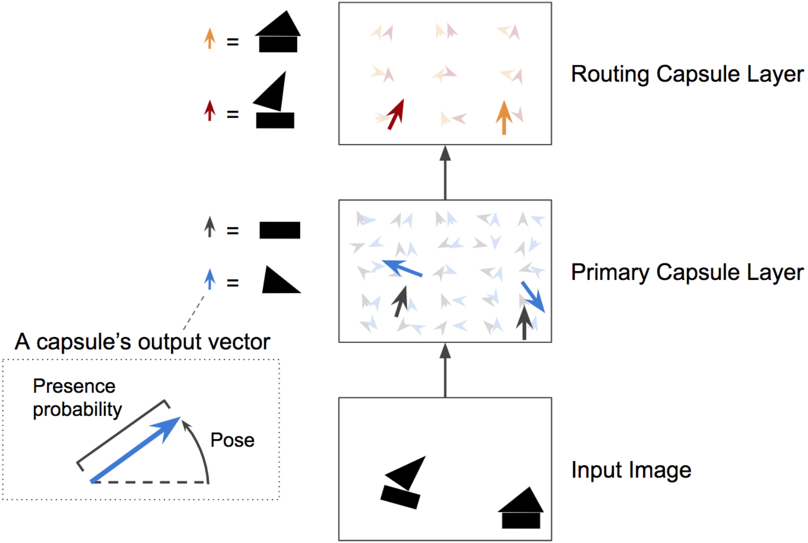
Routing by an Agreement: High-Dimensional Coincidence



https://www.oreilly.com/ideas/introducing-capsule-networks

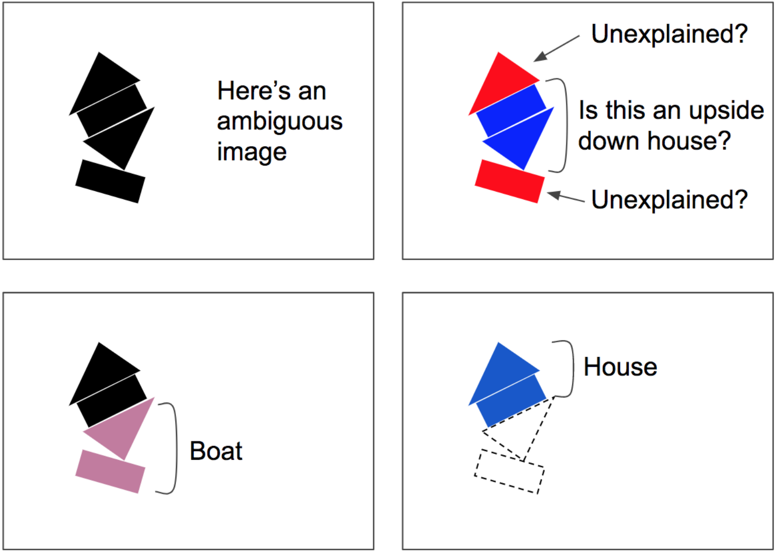
Routing by an Agreement: Illustrative Overview

Routing by an Agreement: Illustrative Overview

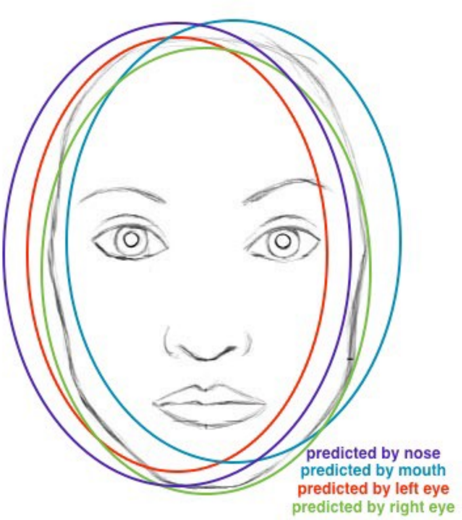


https://www.oreilly.com/ideas/introducing-capsule-networks

Routing by an Agreement: Recognizing Ambiguity in Images



Routing: Lower Levels Voting for Higher-Level Feature



(Sabour, Frosst and Hinton [2017])

How to do it?

(mathematically)

Capsule

What Is a Capsule?

a group of neurons that:

- perform some complicated **internal computations** on their inputs

What Is a Capsule?

a group of neurons that:

- perform some complicated **internal computations** on their inputs
- encapsulate their results into a **small vector** of highly informative outputs

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66

What Is a Capsule?

a group of neurons that:

- perform some complicated **internal computations** on their inputs
- encapsulate their results into a **small vector** of highly informative outputs
- recognize an implicitly defined visual entity (over a limited domain of viewing conditions and deformations)

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66

What Is a Capsule?

a group of neurons that:

- perform some complicated **internal computations** on their inputs
- encapsulate their results into a **small vector** of highly informative outputs
- recognize an implicitly defined visual entity (over a limited domain of viewing conditions and deformations)
- encode the **probability** of the entity being present

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66

What Is a Capsule?

a group of neurons that:

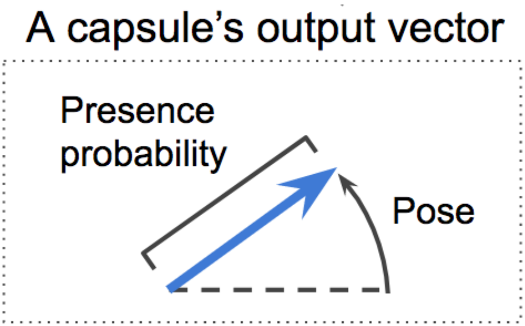
- perform some complicated **internal computations** on their inputs
- encapsulate their results into a **small vector** of highly informative outputs
- recognize an implicitly defined visual entity (over a limited domain of viewing conditions and deformations)
- encode the **probability** of the entity being present
- encode **instantiation parameters**

  pose, lighting, deformation relative to entity's (implicitly defined) canonical version

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66

Output As A Vector

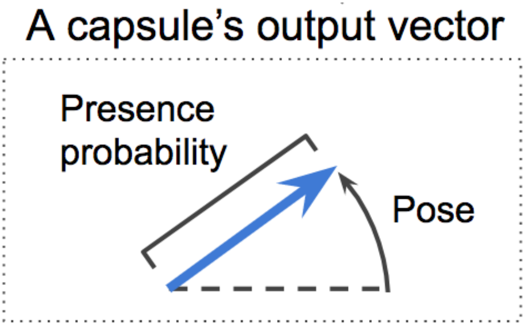
Output As A Vector



- probability of presence: locally **invariant**

  E.g. if 0, 3, 2, 0, 0 leads to 0, 1, 0, 0, then 0, 0, 3, 2, 0 should also lead to 0, 1, 0, 0.

Output As A Vector



- probability of presence: locally **invariant**

  E.g. if 0, 3, 2, 0, 0 leads to 0, 1, 0, 0, then 0, 0, 3, 2, 0 should also lead to 0, 1, 0, 0.

- instantiation parameters: **equivariant**

  E.g. if 0, 3, 2, 0, 0 leads to 0, 1, 0, 0, then 0, 0, 3, 2, 0 might lead to 0, 0, 1, 0.

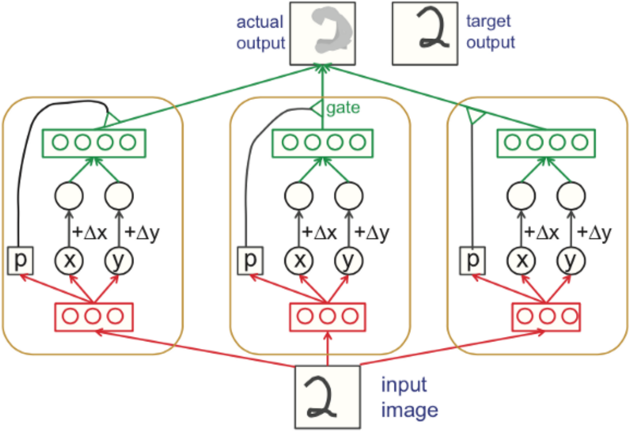
https://www.oreilly.com/ideas/introducing-capsule-networks

Previous Version of Capsules

for illustration taken from "Transforming Auto-Encoders" (Hinton, Krizhevsky and Wang [2011])

Previous Version of Capsules

for illustration taken from "Transforming Auto-Encoders" (Hinton, Krizhevsky and Wang [2011])



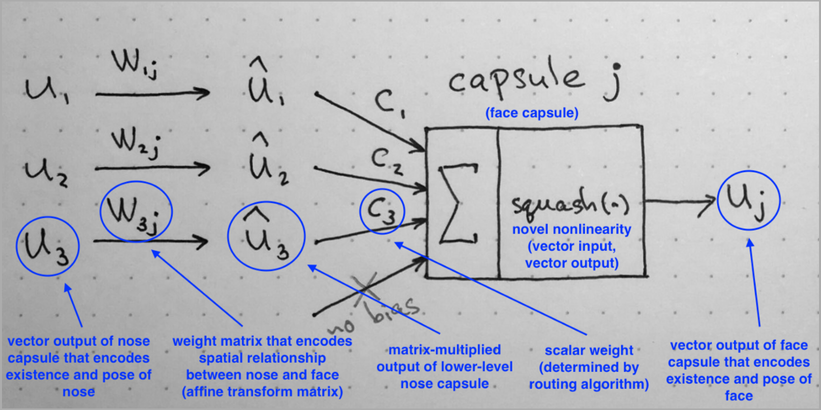
three capsules of a transforming auto-encoder (that models translation)

(Hinton, Krizhevsky and Wang [2011])

Capsule’s Vector Flow

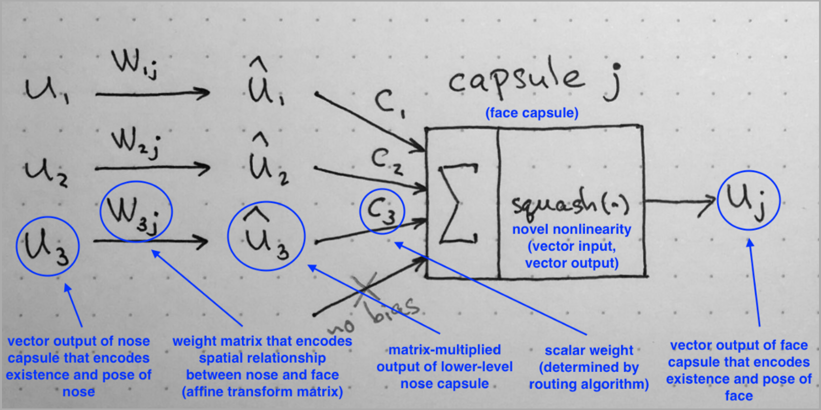
https://cdn-images-1.medium.com/max/1250/1*GbmQ2X9NQoGuJ1M-EOD67g.png

Capsule's Vector Flow



https://cdn-images-1.medium.com/max/1250/1*GbmQ2X9NQoGuJ1M-EOD67g.png

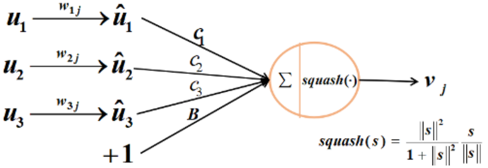
Capsule's Vector Flow

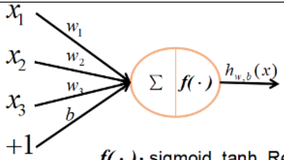


Note: no bias (included in affine transformation matrices W_{ij}'s)

https://cdn-images-1.medium.com/max/1250/1*GbmQ2X9NQoGuJ1M-EOD67g.png

		capsule VS.	traditional neuron
Input from low-level neurons/capsules		**vector**(u_i)	**scalar**(x_i)
Operations	Linear/Affine Transformation	$\hat{u}_{j\|i} = W_{ij} u_i + B_j$ (Eq. 2)	$a_{j\|i} = w_{ij} x_i + b_j$
	Weighting	$s_j = \sum_i c_{ij} \hat{u}_{j\|i}$ (Eq. 2)	$z_j = \sum_{i=1}^{3} 1 \cdot a_{j\|i}$
	Summation		
	Non-linearity activation	$v_j = squash(s_j)$ (Eq. 1)	$h_{w,b}(x) = f(z_j)$
output		**vector**(v_j)	**scalar**(h)



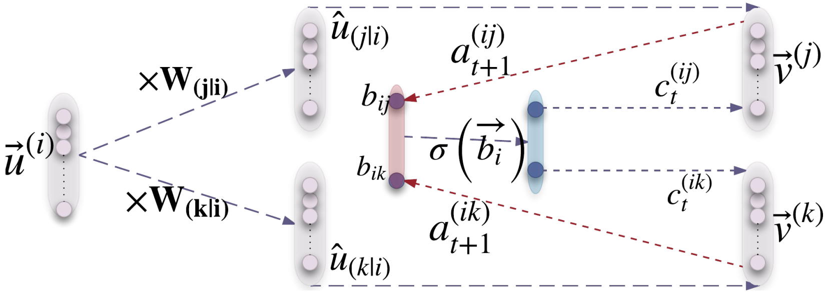


$f(\cdot)$: sigmoid, tanh, ReLU, etc.

Capsule = New Version Neuron!
vector in, vector out VS. scalar in, scalar out

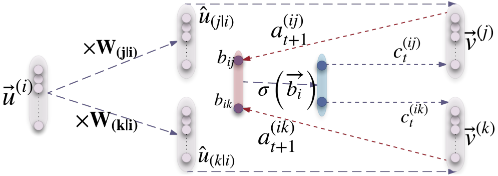
Routing by an Agreement

Capsule Schema with Routing



(Sabour, Frosst and Hinton [2017])

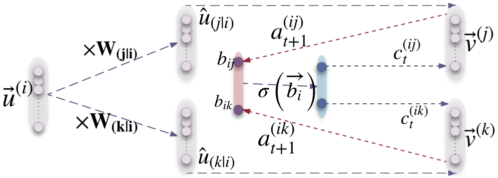
Routing Softmax



$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{1}$$

(Sabour, Frosst and Hinton [2017])

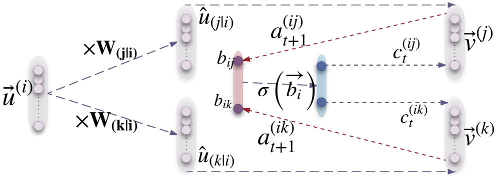
Prediction Vectors



$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i \qquad (2)$$

(Sabour, Frosst and Hinton [2017])

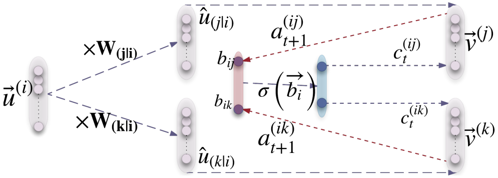
Total Input



$$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i} \tag{3}$$

(Sabour, Frosst and Hinton [2017])

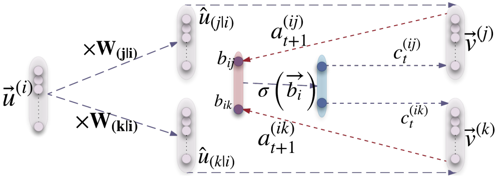
Squashing: (vector) non-linearity



$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||} \tag{4}$$

(Sabour, Frosst and Hinton [2017])

Squashing: (vector) non-linearity



$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||} \tag{4}$$

(Sabour, Frosst and Hinton [2017])

Squashing: Plot for 1-D input

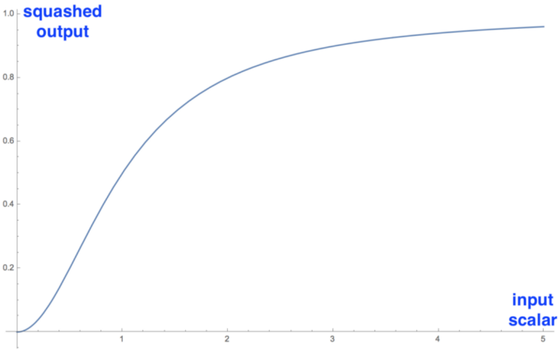
$$\mathbf{v}_j = \underbrace{\boxed{\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}}}_{\text{additional “squashing”}} \underbrace{\boxed{\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}}}_{\text{unit scaling}}$$

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66

Squashing: Plot for 1-D input

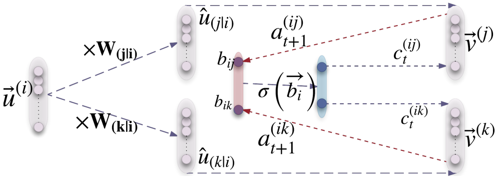
$$\mathbf{v}_j = \boxed{\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}}\boxed{\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}}$$

additional "squashing" unit scaling



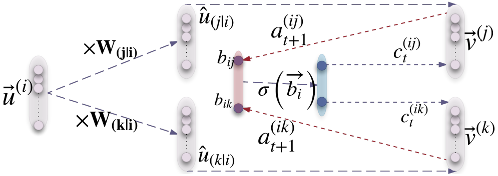
https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66

Routing Algorithm



(Sabour, Frosst and Hinton [2017])

Routing Algorithm

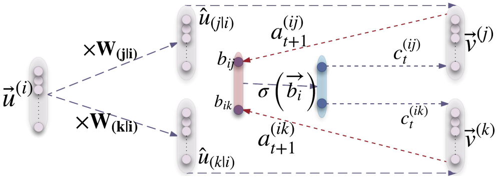


Algorithm Dynamic Routing between Capsules

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)

(Sabour, Frosst and Hinton [2017])

Routing Algorithm

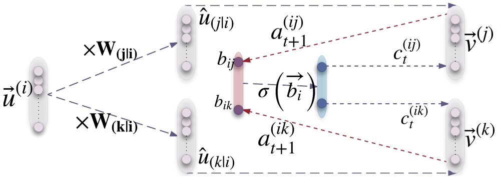


Algorithm Dynamic Routing between Capsules

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.

(Sabour, Frosst and Hinton [2017])

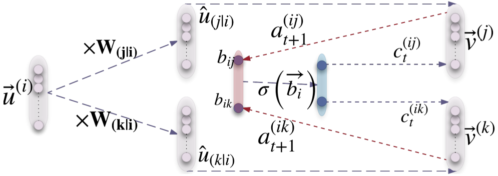
Routing Algorithm



Algorithm Dynamic Routing between Capsules

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**

(Sabour, Frosst and Hinton [2017])

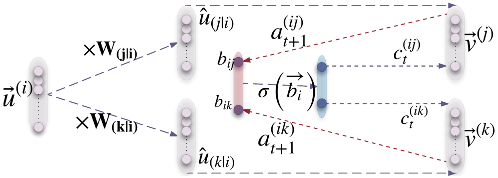


Algorithm Dynamic Routing between Capsules

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` from Eq. 1

(Sabour, Frosst and Hinton [2017])

Routing Algorithm



Algorithm Dynamic Routing between Capsules

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)

2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.

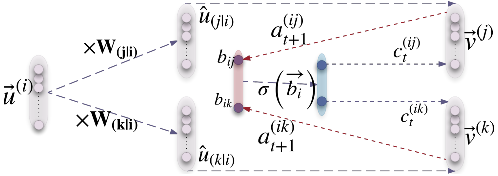
3: **for** r iterations **do**

4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` from Eq. 1

5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$ ▷ total input from Eq. 3

(Sabour, Frosst and Hinton [2017])

Routing Algorithm



Algorithm Dynamic Routing between Capsules

procedure $\textsc{Routing}(\hat{\boldsymbol{u}}_{j|i}, r, l)$

  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.

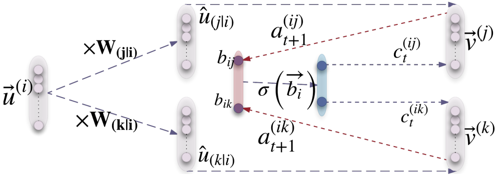
  for r iterations **do**

    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` from Eq. 1

    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$ ▷ total input from Eq. 3

    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` from Eq. 4

(Sabour, Frosst and Hinton [2017])

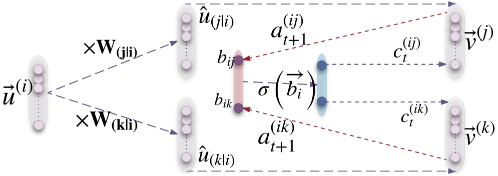


Algorithm Dynamic Routing between Capsules

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` from Eq. 1
        for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$ ▷ total input from Eq. 3
        for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` from Eq. 4
        for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$

(Sabour, Frosst and Hinton [2017])

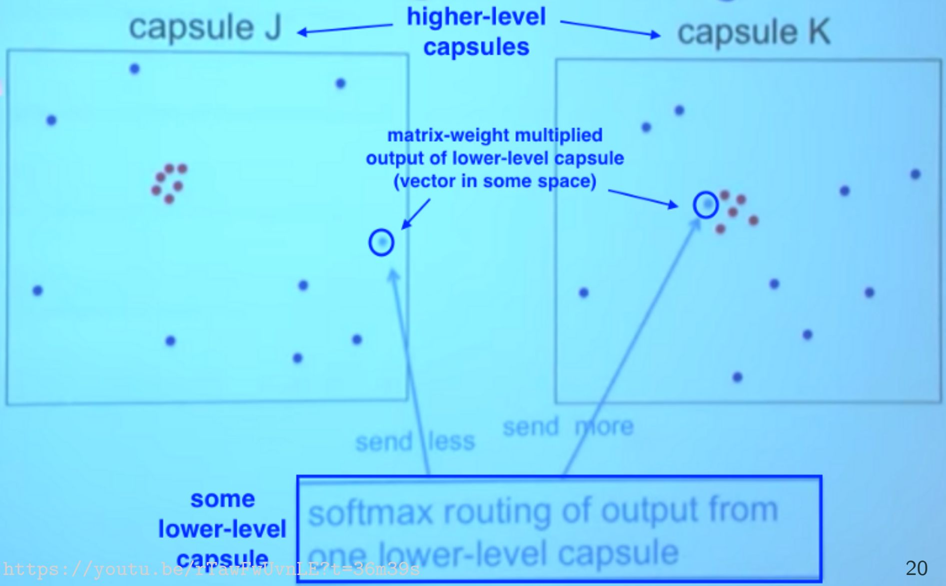
Routing Algorithm



Algorithm Dynamic Routing between Capsules

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` from Eq. 1
        for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$ ▷ total input from Eq. 3
        for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` from Eq. 4
        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 based on agreement



https://youtu.be/rTawFwUvnLE?t=36m39s

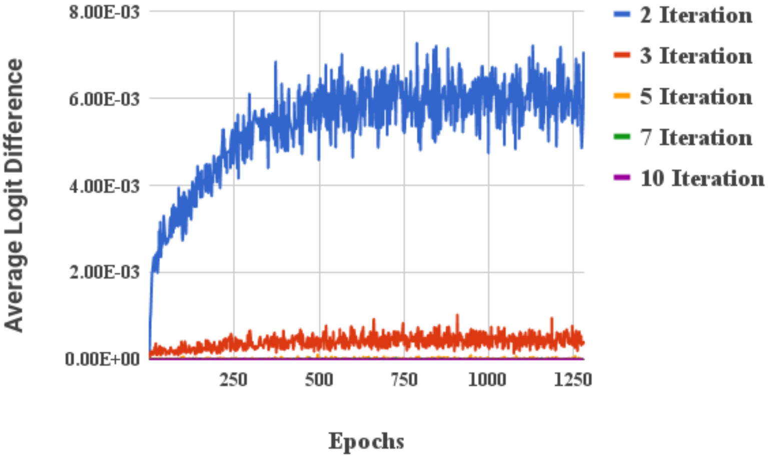
Average Change of Each Routing Logit b_{ij}

(by each routing iteration during training)

(Sabour, Frosst and Hinton [2017])

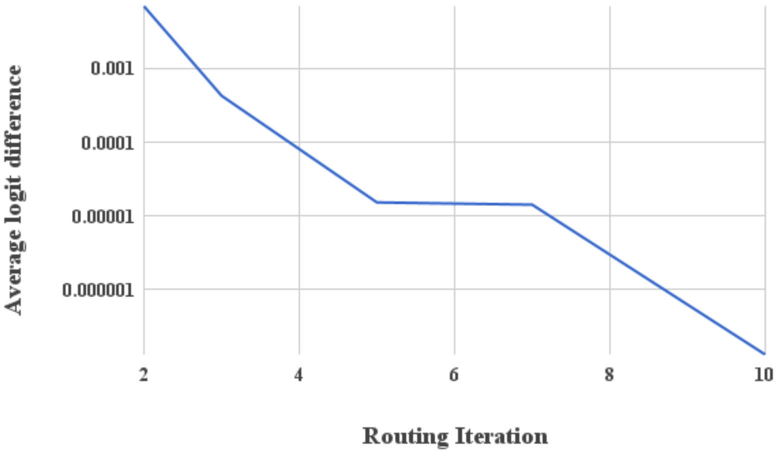
Average Change of Each Routing Logit b_{ij}

(by each routing iteration during training)



(Sabour, Frosst and Hinton [2017])

Log Scale of Final Differences



(Sabour, Frosst and Hinton [2017])

Training Loss of CapsNet on CIFAR10

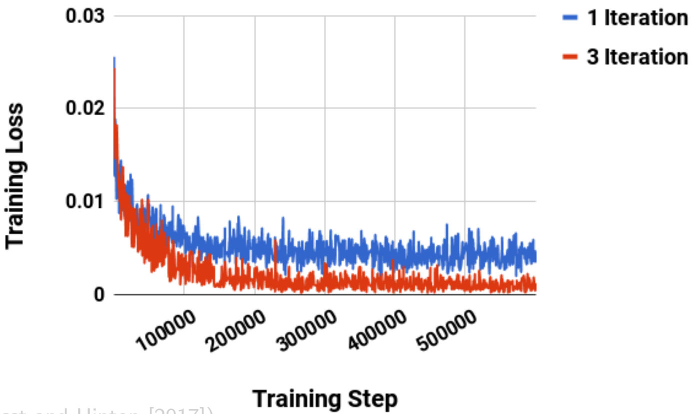
(batch size of 128)

The CapsNet with 3 routing iterations optimizes the loss faster and converges to a lower loss at the end.

Training Loss of CapsNet on CIFAR10

(batch size of 128)

The CapsNet with 3 routing iterations optimizes the loss faster and converges to a lower loss at the end.

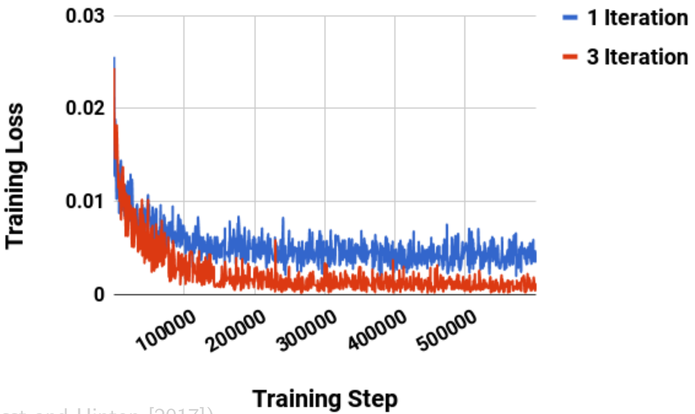


(Sabour, Frosst and Hinton [2017])

Training Loss of CapsNet on CIFAR10

(batch size of 128)

The CapsNet with 3 routing iterations optimizes the loss faster and converges to a lower loss at the end.

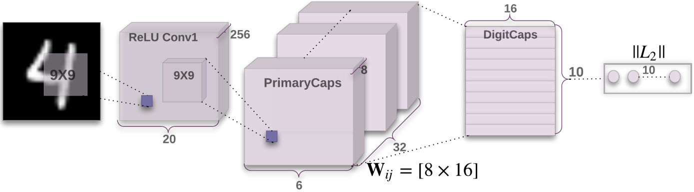


(Sabour, Frosst and Hinton [2017])

Capsule Network

Architecture: Encoder-Decoder

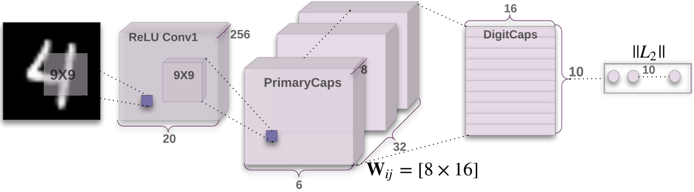
- encoder:



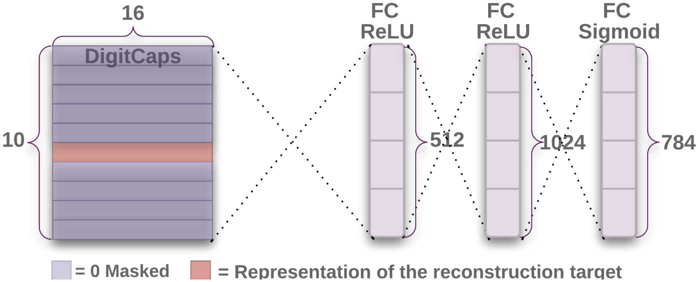
(Sabour, Frosst and Hinton [2017])

Architecture: Encoder-Decoder

- encoder:

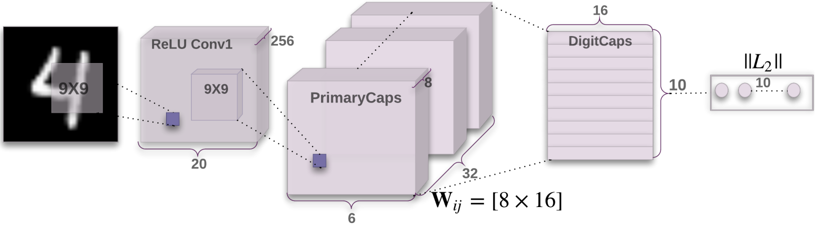


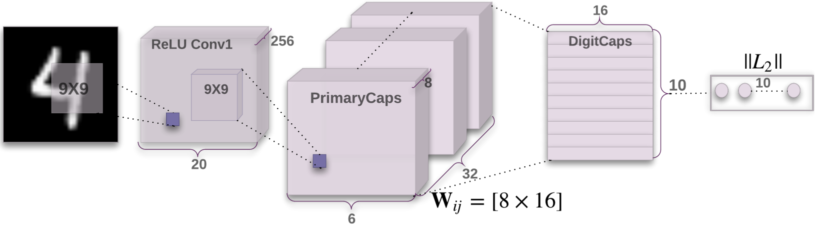
- decoder:



(Sabour, Frosst and Hinton [2017])

Encoder: CapsNet with 3 Layers

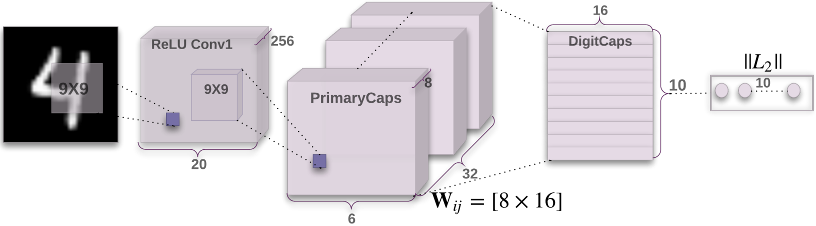




- input: 28 by 28 MNIST digit image

(Sabour, Frosst and Hinton [2017])

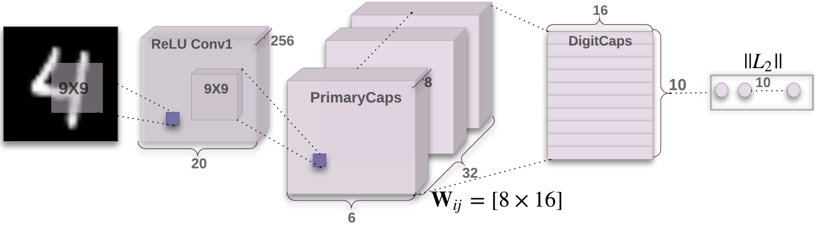
Encoder: CapsNet with 3 Layers



- input: 28 by 28 MNIST digit image
- output: 16-dimensional vector of instantiation parameters

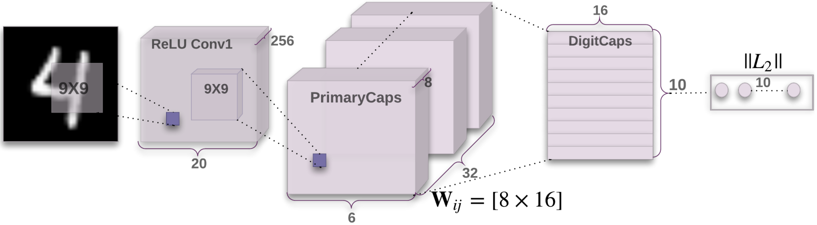
(Sabour, Frosst and Hinton [2017])

Encoder Layer 1: (Standard) Convolutional Layer



(Sabour, Frosst and Hinton [2017])

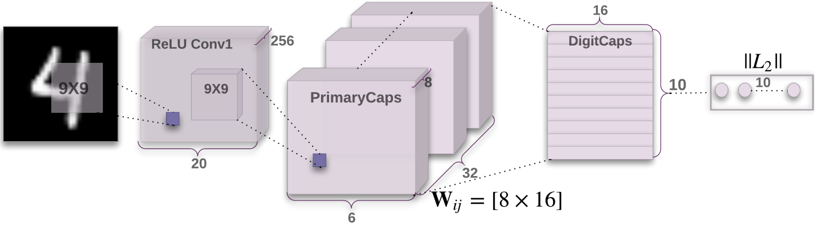
Encoder Layer 1: (Standard) Convolutional Layer



- input: 28×28 image (one color channel)

(Sabour, Frosst and Hinton [2017])

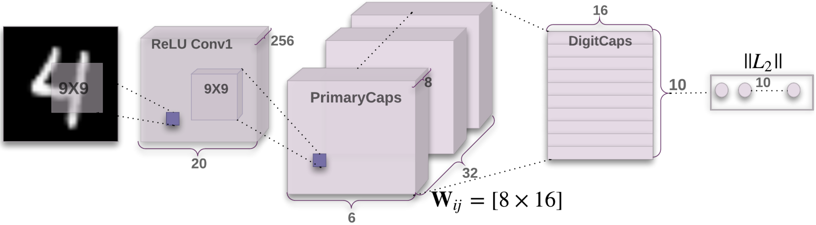
Encoder Layer 1: (Standard) Convolutional Layer



- input: 28×28 image (one color channel)
- output: $20 \times 20 \times 256$

(Sabour, Frosst and Hinton [2017])

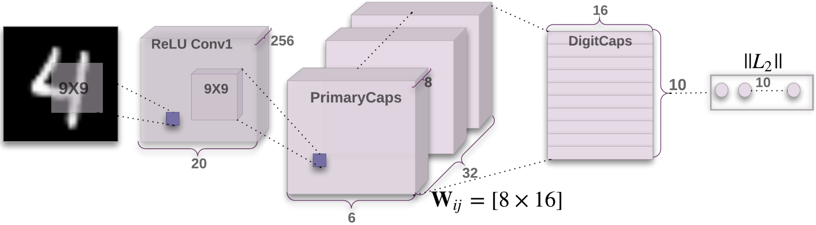
Encoder Layer 1: (Standard) Convolutional Layer



- input: 28×28 image (one color channel)
- output: $20 \times 20 \times 256$
- 256 kernels with size of $9 \times 9 \times 1$

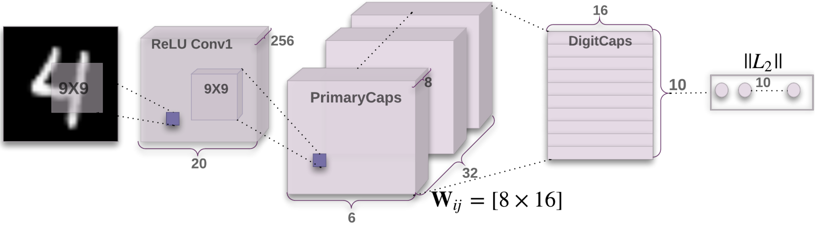
(Sabour, Frosst and Hinton [2017])

Encoder Layer 1: (Standard) Convolutional Layer



- input: 28×28 image (one color channel)
- output: $20 \times 20 \times 256$
- 256 kernels with size of $9 \times 9 \times 1$
- stride 1

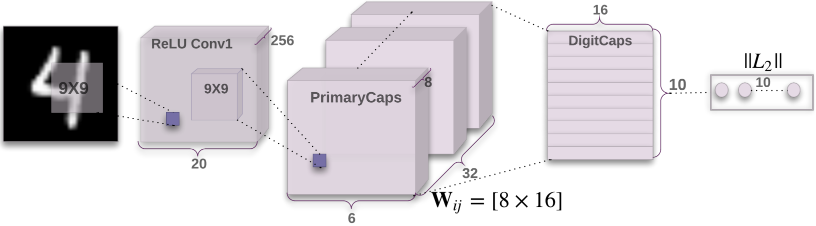
Encoder Layer 1: (Standard) Convolutional Layer



- input: 28×28 image (one color channel)
- output: $20 \times 20 \times 256$
- 256 kernels with size of $9 \times 9 \times 1$
- stride 1
- ReLU activation

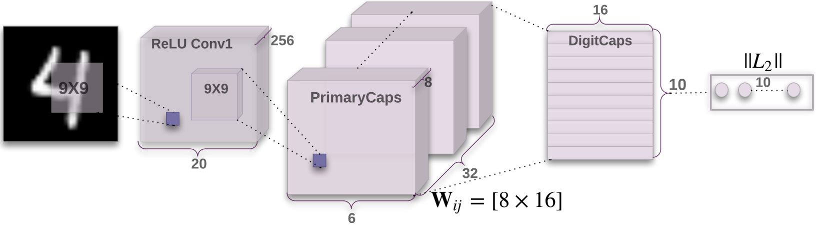
(Sabour, Frosst and Hinton [2017])

Encoder Layer 2: PrimaryCaps



(Sabour, Frosst and Hinton [2017])

Encoder Layer 2: PrimaryCaps

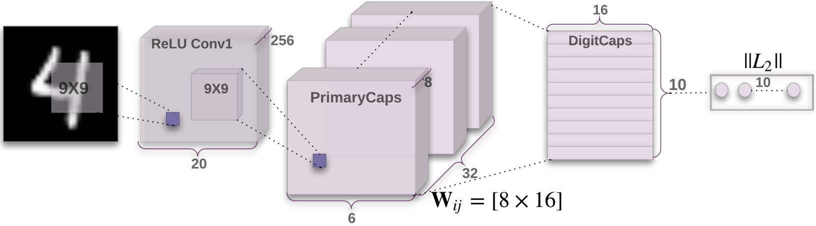


- input: $20 \times 20 \times 256$

  basic features detected by the convolutional layer

(Sabour, Frosst and Hinton [2017])

Encoder Layer 2: PrimaryCaps



- input: $20 \times 20 \times 256$

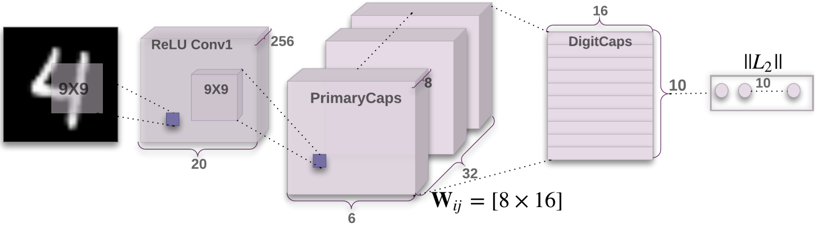
  basic features detected by the convolutional layer

- output: $6 \times 6 \times 8 \times 32$

  vector (activation) outputs of primary capsules

(Sabour, Frosst and Hinton [2017])

Encoder Layer 2: PrimaryCaps



- input: $20 \times 20 \times 256$

  basic features detected by the convolutional layer

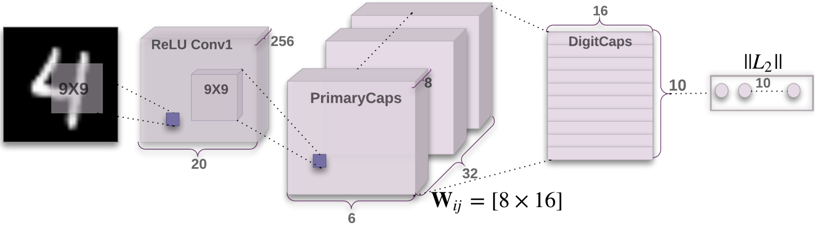
- output: $6 \times 6 \times 8 \times 32$

  vector (activation) outputs of primary capsules

- 32 primary capsules

(Sabour, Frosst and Hinton [2017])

Encoder Layer 2: PrimaryCaps



- input: $20 \times 20 \times 256$

  basic features detected by the convolutional layer

- output: $6 \times 6 \times 8 \times 32$

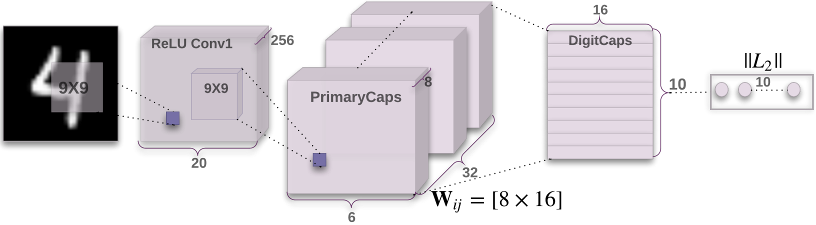
  vector (activation) outputs of primary capsules

- 32 primary capsules
- each applies eight $9 \times 9 \times 256$ convolutional kernels

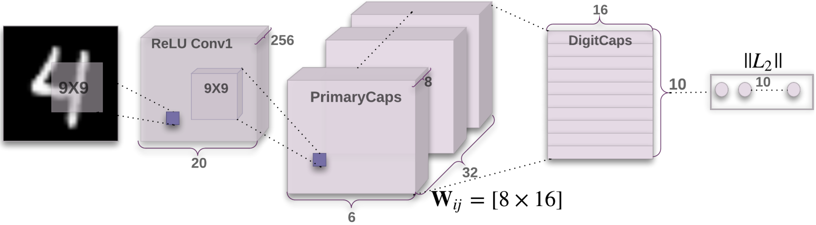
  to the $20 \times 20 \times 256$ input to produce $6 \times 6 \times 8$ output

(Sabour, Frosst and Hinton [2017])

Encoder Layer 3: DigitCaps



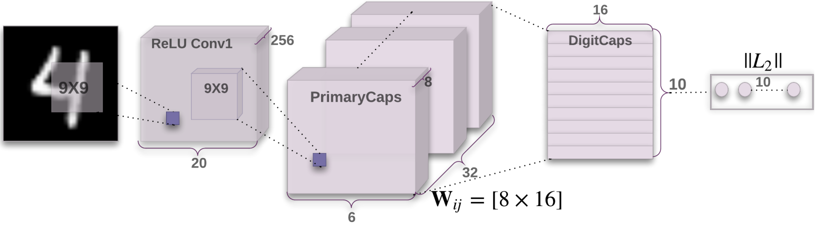
Encoder Layer 3: DigitCaps



- input: $6 \times 6 \times 8 \times 32$

  $(6 \times 6 \times 32)$-many 8-dimensional vector activations

Encoder Layer 3: DigitCaps



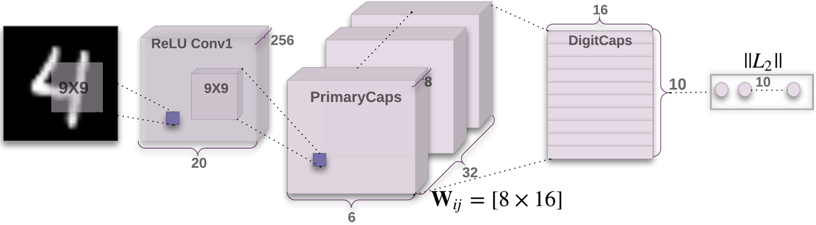
- input: $6 \times 6 \times 8 \times 32$

  $(6 \times 6 \times 32)$-many 8-dimensional vector activations

- output: 16×10

(Sabour, Frosst and Hinton [2017])

Encoder Layer 3: DigitCaps



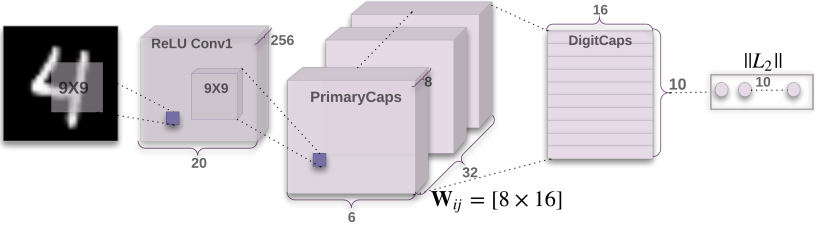
- input: $6 \times 6 \times 8 \times 32$

  $(6 \times 6 \times 32)$-many 8-dimensional vector activations

- output: 16×10
- 10 digit capsules

(Sabour, Frosst and Hinton [2017])

Encoder Layer 3: DigitCaps



- input: $6 \times 6 \times 8 \times 32$

  $(6 \times 6 \times 32)$-many 8-dimensional vector activations
- output: 16×10
- 10 digit capsules
- input vectors gets their own 8×16 weight matrix W_{ij}

  that maps 8-dimensional input space to the 16-dimensional capsule output space

(Sabour, Frosst and Hinton [2017])

Margin Loss

for a Digit Existence

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

https://medium.com/@pechyonkin/part-iv-capsnet-architecture-6a64422f7dce

Margin Loss

to Train the Whole Encoder

In other words, each DigitCap c has loss:

Margin Loss

to Train the Whole Encoder

In other words, each DigitCap c has loss:

$$L_c = \begin{cases} \max(0, m^+ - ||\mathbf{v}_c||)^2 & \text{iff a digit of class } c \text{ is present,} \\ \lambda \max(0, ||\mathbf{v}_c|| - m^-)^2 & \text{otherwise.} \end{cases}$$

(Sabour, Frosst and Hinton [2017])

Margin Loss

to Train the Whole Encoder

In other words, each DigitCap c has loss:

$$L_c = \begin{cases} \max(0, m^+ - ||\mathbf{v}_c||)^2 & \text{iff a digit of class } c \text{ is present,} \\ \lambda \max(0, ||\mathbf{v}_c|| - m^-)^2 & \text{otherwise.} \end{cases}$$

- $m^+ = 0.9$:

  The loss is 0 iff the **correct** DigitCap predicts the correct label with probability ≥ 0.9.

(Sabour, Frosst and Hinton [2017])

Margin Loss

to Train the Whole Encoder

In other words, each DigitCap c has loss:

$$L_c = \begin{cases} \max(0, m^+ - ||\mathbf{v}_c||)^2 & \text{iff a digit of class } c \text{ is present,} \\ \lambda \max(0, ||\mathbf{v}_c|| - m^-)^2 & \text{otherwise.} \end{cases}$$

- $m^+ = 0.9$:

  The loss is 0 iff the **correct** DigitCap predicts the correct label with probability ≥ 0.9.

- $m^- = 0.1$:

  The loss is 0 iff the **mismatching** DigitCap predicts an incorrect label with probability ≤ 0.1.

(Sabour, Frosst and Hinton [2017])

Margin Loss

to Train the Whole Encoder

In other words, each DigitCap c has loss:

$$L_c = \begin{cases} \max(0, m^+ - ||\mathbf{v}_c||)^2 & \text{iff a digit of class } c \text{ is present,} \\ \lambda \max(0, ||\mathbf{v}_c|| - m^-)^2 & \text{otherwise.} \end{cases}$$

- $m^+ = 0.9$:

  The loss is 0 iff the **correct** DigitCap predicts the correct label with probability ≥ 0.9.

- $m^- = 0.1$:

  The loss is 0 iff the **mismatching** DigitCap predicts an incorrect label with probability ≤ 0.1.

- $\lambda = 0.5$ is down-weighting of the loss for absent digit classes.

  It stops the initial learning from shrinking the lengths of the activity vectors.

(Sabour, Frosst and Hinton [2017])

Margin Loss

to Train the Whole Encoder

In other words, each DigitCap c has loss:

$$L_c = \begin{cases} \max(0, m^+ - ||\mathbf{v}_c||)^2 & \text{iff a digit of class } c \text{ is present,} \\ \lambda \max(0, ||\mathbf{v}_c|| - m^-)^2 & \text{otherwise.} \end{cases}$$

- $m^+ = 0.9$:

  The loss is 0 iff the **correct** DigitCap predicts the correct label with probability ≥ 0.9.

- $m^- = 0.1$:

  The loss is 0 iff the **mismatching** DigitCap predicts an incorrect label with probability ≤ 0.1.

- $\lambda = 0.5$ is down-weighting of the loss for absent digit classes.

  It stops the initial learning from shrinking the lengths of the activity vectors.

- Squares? Because there are L_2 norms in the loss function?

(Sabour, Frosst and Hinton [2017])

Margin Loss

to Train the Whole Encoder

In other words, each DigitCap c has loss:

$$L_c = \begin{cases} \max(0, m^+ - ||\mathbf{v}_c||)^2 & \text{iff a digit of class } c \text{ is present,} \\ \lambda \max(0, ||\mathbf{v}_c|| - m^-)^2 & \text{otherwise.} \end{cases}$$

- $m^+ = 0.9$:

  The loss is 0 iff the **correct** DigitCap predicts the correct label with probability ≥ 0.9.
- $m^- = 0.1$:

  The loss is 0 iff the **mismatching** DigitCap predicts an incorrect label with probability ≤ 0.1.
- $\lambda = 0.5$ is down-weighting of the loss for absent digit classes.

  It stops the initial learning from shrinking the lengths of the activity vectors.
- Squares? Because there are L_2 norms in the loss function?
- The total loss is **the sum of the losses** of all digit capsules.

(Sabour, Frosst and Hinton [2017])

Margin Loss

Function Value for Positive and for Negative Class

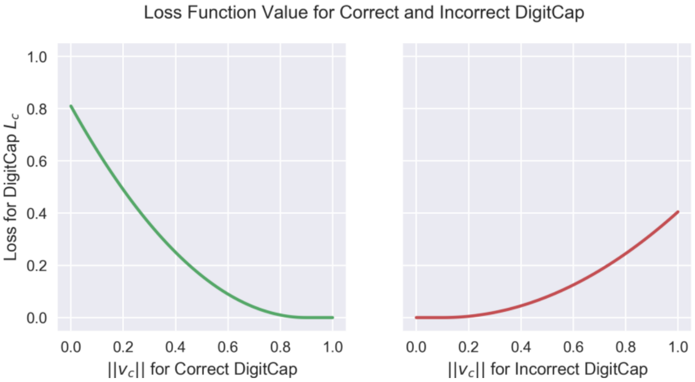
- For the **correct** DigitCap, the loss is 0 iff it predicts the correct label with probability ≥ 0.9.
- For the **mismatching** DigitCap, the loss is 0 iff it predicts an incorrect label with probability ≤ 0.1.

https://medium.com/@pechyonkin/part-iv-capsnet-architecture-6a64422f7dce

Margin Loss

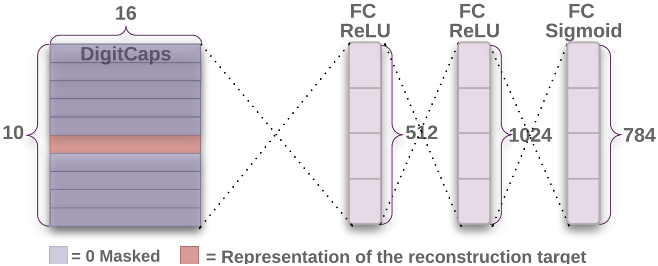
Function Value for Positive and for Negative Class

- For the **correct** DigitCap, the loss is 0 iff it predicts the correct label with probability ≥ 0.9.
- For the **mismatching** DigitCap, the loss is 0 iff it predicts an incorrect label with probability ≤ 0.1.



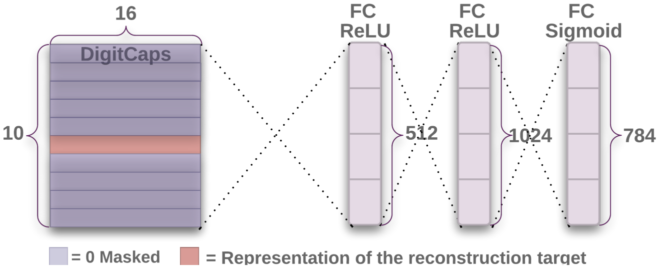
https://medium.com/@pechyonkin/part-iv-capsnet-architecture-6a64422f7dce

Decoder: Regularization of CapsNets



(Sabour, Frosst and Hinton [2017])

Decoder: Regularization of CapsNets



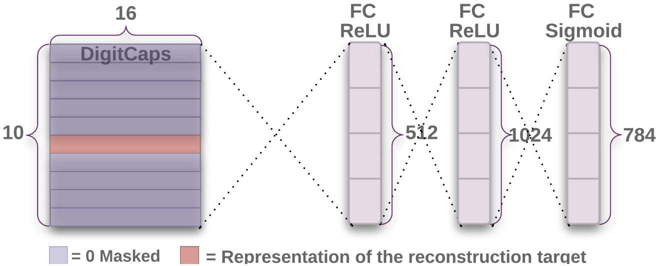
Decoder is used for regularization:

- decodes input from DigitCaps

  to recreate an image of a (28×28)-pixels digit

(Sabour, Frosst and Hinton [2017])

Decoder: Regularization of CapsNets



Decoder is used for regularization:

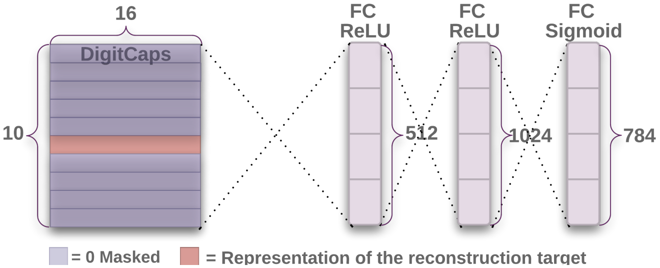
- decodes input from DigitCaps

  to recreate an image of a (28×28)-pixels digit

- with the loss function being the Euclidean distance

(Sabour, Frosst and Hinton [2017])

Decoder: Regularization of CapsNets



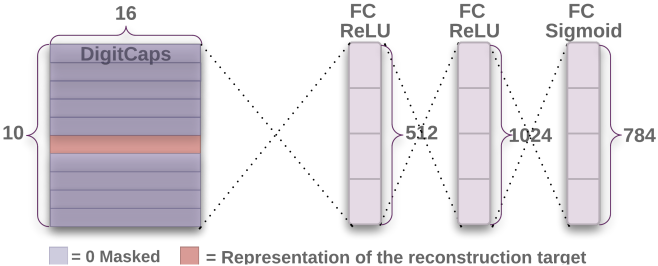
Decoder is used for regularization:

- decodes input from DigitCaps

  to recreate an image of a (28×28)-pixels digit
- with the loss function being the Euclidean distance
- ignores the negative classes

(Sabour, Frosst and Hinton [2017])

Decoder: Regularization of CapsNets



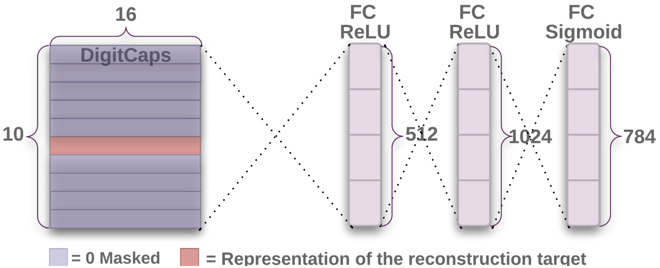
Decoder is used for regularization:

- decodes input from DigitCaps

  to recreate an image of a (28×28)-pixels digit
- with the loss function being the Euclidean distance
- ignores the negative classes
- forces capsules to learn features useful for reconstruction

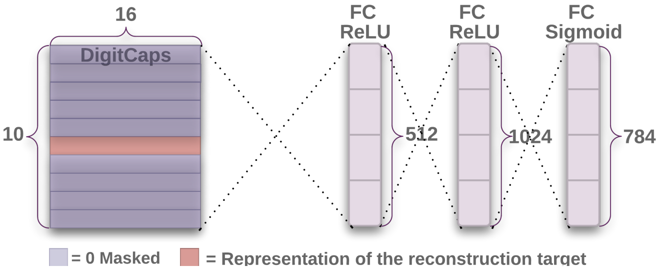
(Sabour, Frosst and Hinton [2017])

Decoder: 3 Fully Connected Layers



(Sabour, Frosst and Hinton [2017])

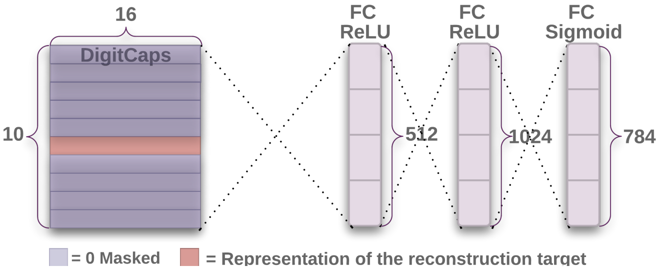
Decoder: 3 Fully Connected Layers



- Layer 4: from 16×10 input to 512 output, ReLU activations

(Sabour, Frosst and Hinton [2017])

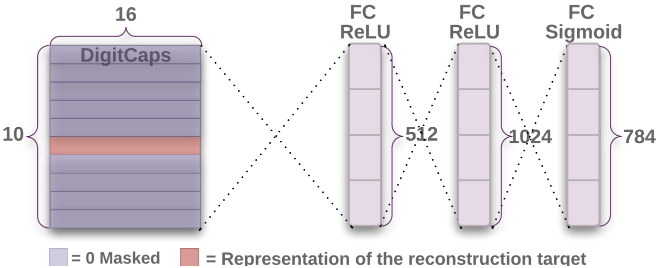
Decoder: 3 Fully Connected Layers



- Layer 4: from 16×10 input to 512 output, ReLU activations
- Layer 5: from 512 input to 1024 output, ReLU activations

(Sabour, Frosst and Hinton [2017])

Decoder: 3 Fully Connected Layers

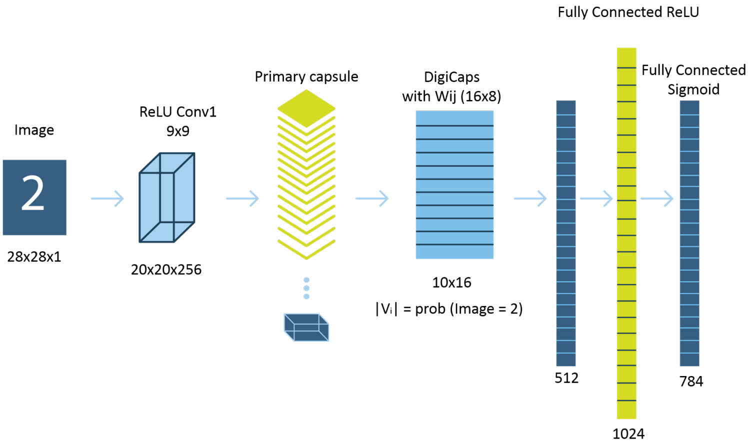


- Layer 4: from 16×10 input to 512 output, ReLU activations
- Layer 5: from 512 input to 1024 output, ReLU activations
- Layer 6: from 1024 input to 784 output, sigmoid activations

  (after reshaping it produces a (28×28)-pixels decoded image)

(Sabour, Frosst and Hinton [2017])

Architecture: Summary

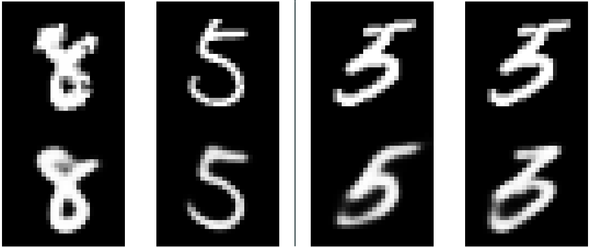


https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2017/12/Capsule-Neural-Network-Architecture-Capsule-Networks-Edureka.png

Experiments

MNIST Reconstructions (CapsNet, 3 routing iterations)

Label:	8	5	5	5
Prediction:	8	5	3	3
Reconstruction:	8	5	5	3



(Sabour, Frosst and Hinton [2017])

Dimension Perturbations

one of the 16 dimensions, by intervals of 0.05 in the range $[-0.25, 0.25]$:

Dimension Perturbations

one of the 16 dimensions, by intervals of 0.05 in the range $[-0.25, 0.25]$:

Interpretation	Reconstructions after perturbing
"scale and thickness"	
"localized part"	

(Sabour, Frosst and Hinton [2017])

Dimension Perturbations

one of the 16 dimensions, by intervals of 0.05 in the range $[-0.25, 0.25]$:

Interpretation	Reconstructions after perturbing
"scale and thickness"	
"localized part"	
"stroke thickness"	
"localized skew"	

(Sabour, Frosst and Hinton [2017])

Dimension Perturbations

one of the 16 dimensions, by intervals of 0.05 in the range $[-0.25, 0.25]$:

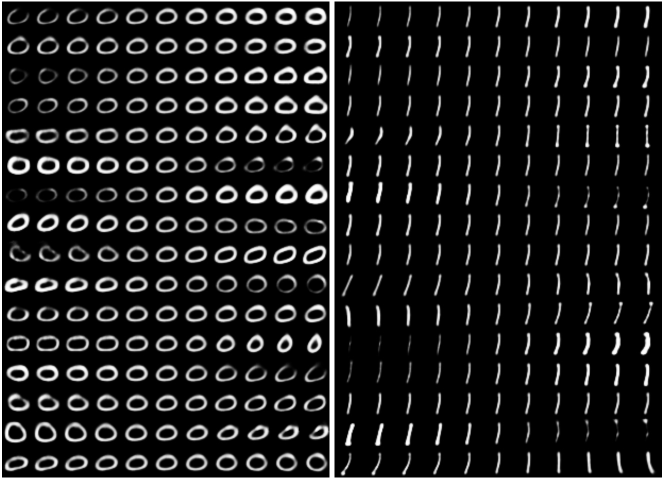
Interpretation	Reconstructions after perturbing
"scale and thickness"	
"localized part"	
"stroke thickness"	
"localized skew"	
"width and translation"	
"localized part"	

(Sabour, Frosst and Hinton [2017])

Dimension Perturbations: Latent Codes of 0 and 1

rows: DigitCaps dimensions

columns (from left to right): $+\{-0.25, -0.2, -0.15, -0.1, -0.05, 0, 0.05, 0.1, 0.15, 0.2, 0.25\}$

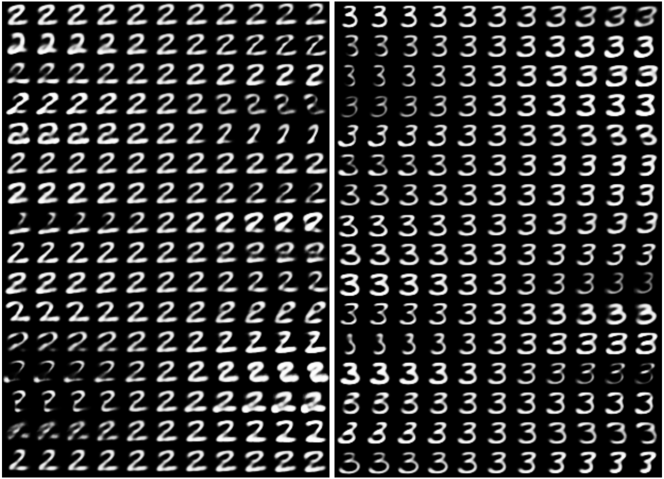


https://github.com/XifengGuo/CapsNet-Keras

Dimension Perturbations: Latent Codes of 2 and 3

rows: DigitCaps dimensions

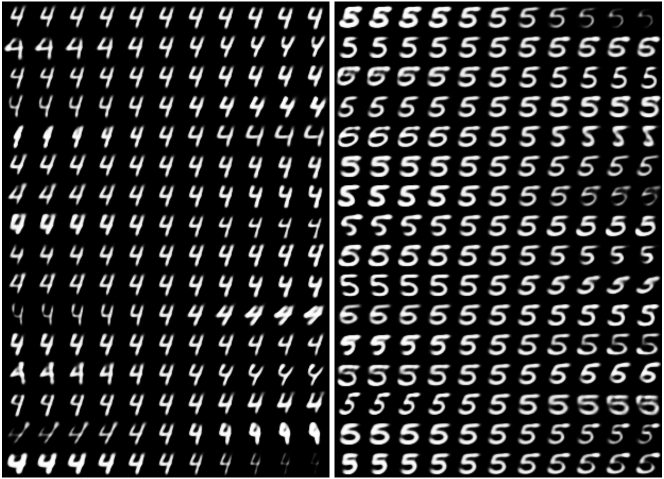
columns (from left to right): $+\{-0.25, -0.2, -0.15, -0.1, -0.05, 0, 0.05, 0.1, 0.15, 0.2, 0.25\}$



Dimension Perturbations: Latent Codes of 4 and 5

rows: DigitCaps dimensions

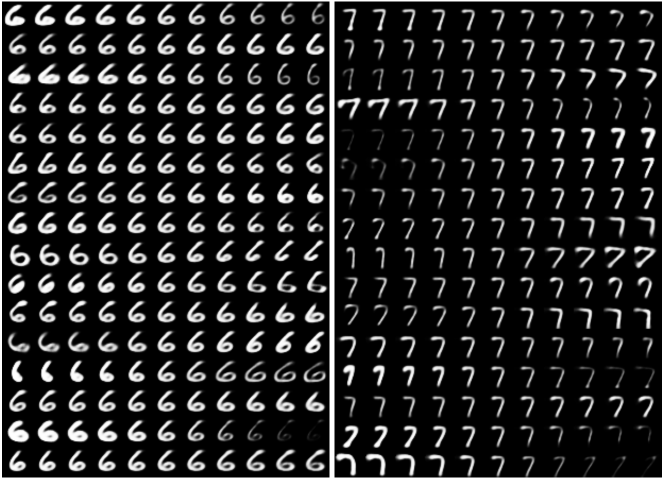
columns (from left to right): $+\ \{-0.25, -0.2, -0.15, -0.1, -0.05, 0, 0.05, 0.1, 0.15, 0.2, 0.25\}$



Dimension Perturbations: Latent Codes of 6 and 7

rows: DigitCaps dimensions

columns (from left to right): $+\ \{-0.25, -0.2, -0.15, -0.1, -0.05, 0, 0.05, 0.1, 0.15, 0.2, 0.25\}$

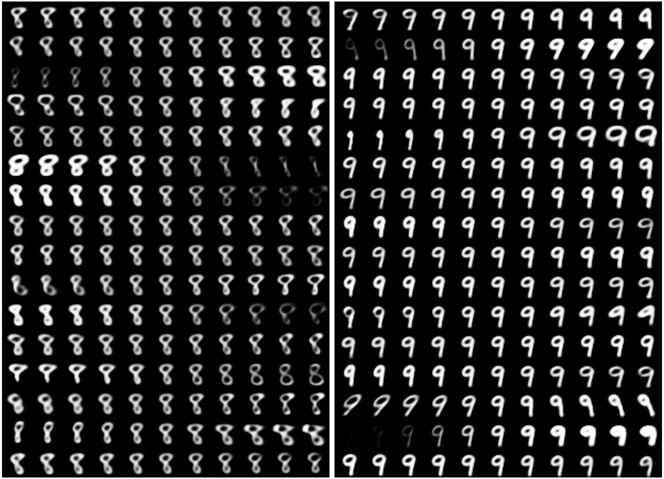


https://github.com/XifengGuo/CapsNet-Keras

Dimension Perturbations: Latent Codes of 8 and 9

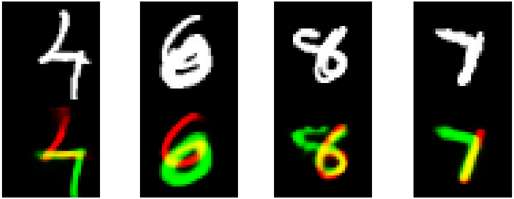
rows: DigitCaps dimensions

columns (from left to right): $+ \{-0.25, -0.2, -0.15, -0.1, -0.05, 0, 0.05, 0.1, 0.15, 0.2, 0.25\}$



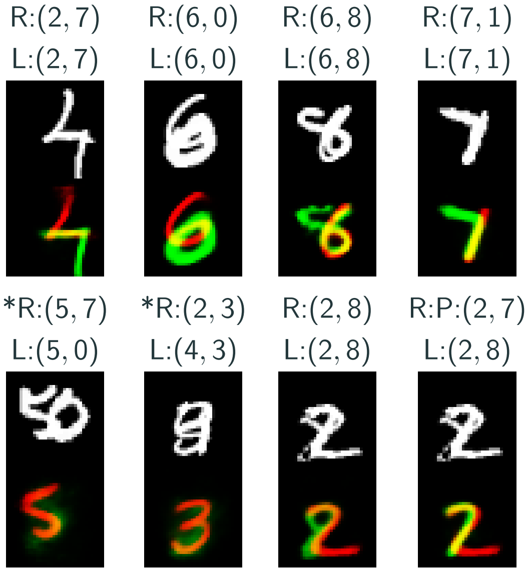
MultiMNIST Reconstructions (CapsNet, 3 routing iterations)

R:(2, 7)	R:(6, 0)	R:(6, 8)	R:(7, 1)
L:(2, 7)	L:(6, 0)	L:(6, 8)	L:(7, 1)



(Sabour, Frosst and Hinton [2017])

MultiMNIST Reconstructions (CapsNet, 3 routing iterations)

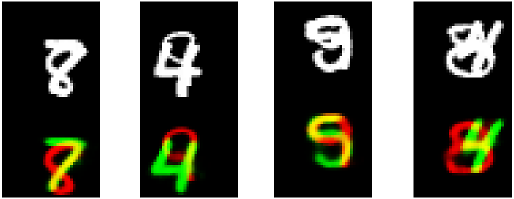


(Sabour, Frosst and Hinton [2017])

MultiMNIST Reconstructions (CapsNet, 3 routing iterations)

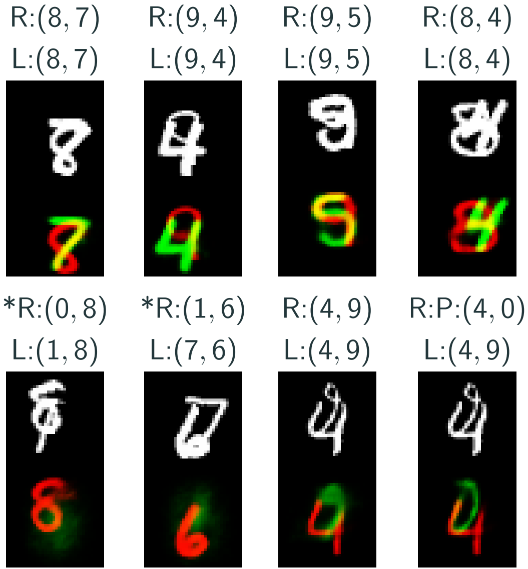
R:(8, 7) R:(9, 4) R:(9, 5) R:(8, 4)

L:(8, 7) L:(9, 4) L:(9, 5) L:(8, 4)



(Sabour, Frosst and Hinton [2017])

MultiMNIST Reconstructions (CapsNet, 3 routing iterations)



(Sabour, Frosst and Hinton [2017])

Results on MNIST and MultiMNIST

CapsNet classification test accuracy:

Results on MNIST and MultiMNIST

CapsNet classification test accuracy:

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**

(The MNIST average and standard deviation results are reported from 3 trials.)

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

CIFAR10

- 10.6% test error

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

CIFAR10

- 10.6% test error
- ensemble of 7 models

Results on Other Datasets

CIFAR10

- 10.6% test error
- ensemble of 7 models
- 3 routing iterations

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

CIFAR10

- 10.6% test error
- ensemble of 7 models
- 3 routing iterations
- 24×24 patches of the image

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

CIFAR10

- 10.6% test error
- ensemble of 7 models
- 3 routing iterations
- 24×24 patches of the image
- about what standard convolutional nets achieved when they were first applied to CIFAR10 (Zeiler and Fergus [2013])

(Sabour, Frosst and Hinton [2017])

Conclusion

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics
- no algorithm to implement and train a capsule network:

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics
- no algorithm to implement and train a capsule network:
  - now **dynamic routing algorithm**

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics
- no algorithm to implement and train a capsule network:
  - now **dynamic routing algorithm**
  - one of the reasons: computers not powerful enough in the pre-GPU-based era

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics
- no algorithm to implement and train a capsule network:
  - now **dynamic routing algorithm**
  - one of the reasons: computers not powerful enough in the pre-GPU-based era
- capable of learning to achieve **state-of-the art** performance by only using **a fraction of the data** compared to CNN

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics
- no algorithm to implement and train a capsule network:
  - now **dynamic routing algorithm**
  - one of the reasons: computers not powerful enough in the pre-GPU-based era
- capable of learning to achieve **state-of-the art** performance by only using **a fraction of the data** compared to CNN
  - task of telling digits apart: the human brain needs a couple of dozens of examples (hundreds at most).

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics
- no algorithm to implement and train a capsule network:
  - now **dynamic routing algorithm**
  - one of the reasons: computers not powerful enough in the pre-GPU-based era
- capable of learning to achieve **state-of-the art** performance by only using **a fraction of the data** compared to CNN
  - task of telling digits apart: the human brain needs a couple of dozens of examples (hundreds at most).
  - On the other hand, CNNs typically need tens of thousands of them.

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics
- no algorithm to implement and train a capsule network:
  - now **dynamic routing algorithm**
  - one of the reasons: computers not powerful enough in the pre-GPU-based era
- capable of learning to achieve **state-of-the art** performance by only using **a fraction of the data** compared to CNN
  - task of telling digits apart: the human brain needs a couple of dozens of examples (hundreds at most).
  - On the other hand, CNNs typically need tens of thousands of them.

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Benefits:

- a new building block usable in deep learning to better model hierarchical relationships
- representations similar to scene graphs in computer graphics
- no algorithm to implement and train a capsule network:
  - now **dynamic routing algorithm**
  - one of the reasons: computers not powerful enough in the pre-GPU-based era
- capable of learning to achieve **state-of-the art** performance by only using **a fraction of the data** compared to CNN
  - task of telling digits apart: the human brain needs a couple of dozens of examples (hundreds at most).
  - On the other hand, CNNs typically need tens of thousands of them.

Downsides:

- current implementations: much slower than other modern deep learning models

https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Thank you!

Questions?

Backup Slides

Results on Other Datasets

smallNORB (LeCun et al. [2004])

- 2.7% test error

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

smallNORB (LeCun et al. [2004])

- 2.7% test error
- 96×96 stereo grey-scale images

  resized to 48×48; during training processed random 32×32 crops; during test the central 32×32 patch

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

smallNORB (LeCun et al. [2004])

- 2.7% test error
- 96×96 stereo grey-scale images

  resized to 48×48; during training processed random 32×32 crops; during test the central 32×32 patch
- same CapsNet architecture as for MNIST

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

smallNORB (LeCun et al. [2004])

- 2.7% test error
- 96×96 stereo grey-scale images

  resized to 48×48; during training processed random 32×32 crops; during test the central 32×32 patch
- same CapsNet architecture as for MNIST
- on-par with the state-of-the-art (Ciresan et al. [2011])

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

SVHN (Netzer et al. [2011])

- 4.3% test error

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

SVHN (Netzer et al. [2011])

- 4.3% test error
- only 73257 images!

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

SVHN (Netzer et al. [2011])

- 4.3% test error
- only 73257 images!
- the number of first convolutional layer channels reduced to 64

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

SVHN (Netzer et al. [2011])

- 4.3% test error
- only 73257 images!
- the number of first convolutional layer channels reduced to 64
- the primary capsule layer: to 16 $6D$-capsules

(Sabour, Frosst and Hinton [2017])

Results on Other Datasets

SVHN (Netzer et al. [2011])

- 4.3% test error
- only 73257 images!
- the number of first convolutional layer channels reduced to 64
- the primary capsule layer: to 16 $6D$-capsules
- final capsule layer: $8D$-capsules

(Sabour, Frosst and Hinton [2017])

Further Reading i

References i