SAND2018-8239C

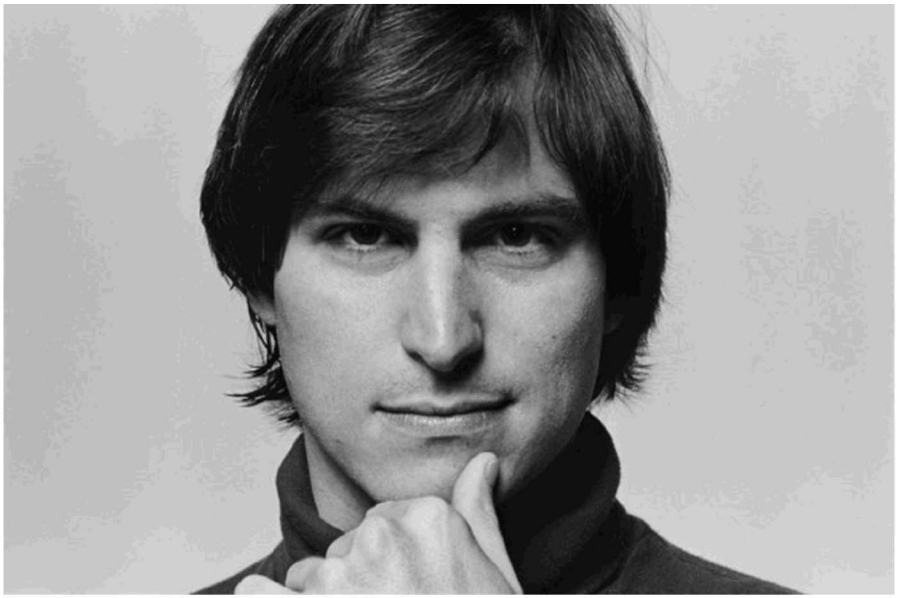
MLDL

Machine Learning and Deep Learning Conference 2018

Capsule Networks: Capturing Presence and *Orientation* of Representations

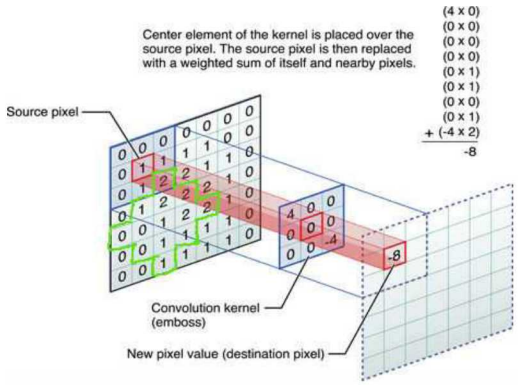
- JD Doak, 5853
- Joshua Coon, 5448
- Rolando Fernandez, 5448
- Mark T. Louie, 5548
- Theo Stangebye, 5448

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

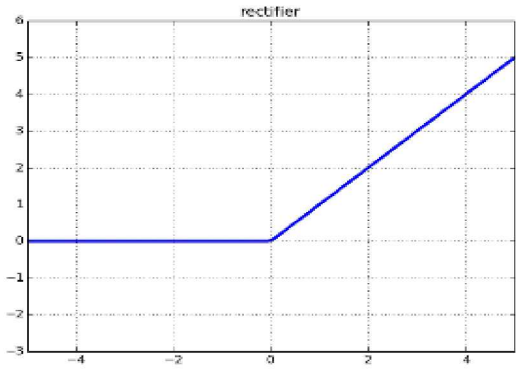


Comparison of Traditional CNNs to CapNets

Convolutional Layer



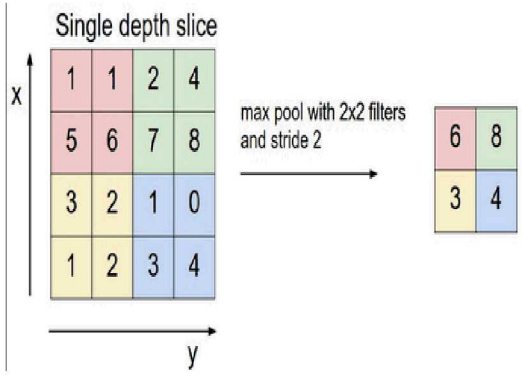
RELU Layer



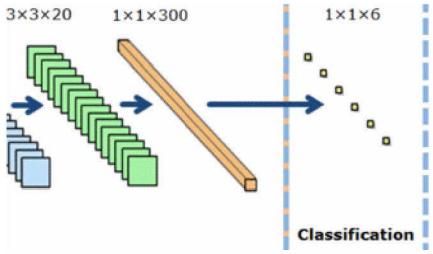
Traditional CNN

1. Initialize all trainable weights to random.
2. Forward propagate images (one at a time or in batches) through network.
    1. Convolutional filter layer
    2. RELU layer
    3. Pooling layer
    4. Fully Connected final layer
3. Calculate loss function= $\sum(target - output)^2$.
4. Backpropagate through network to calculate partial derivative of error w.r.t. weights.
5. Reduce error through stochastic gradient descent to adjust weights on trainable parameters to reduce error.
6. Repeat 2-5 until convergence.

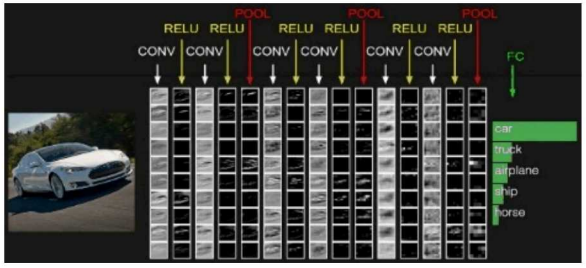
Max Pooling Layer



Fully Connected Layer

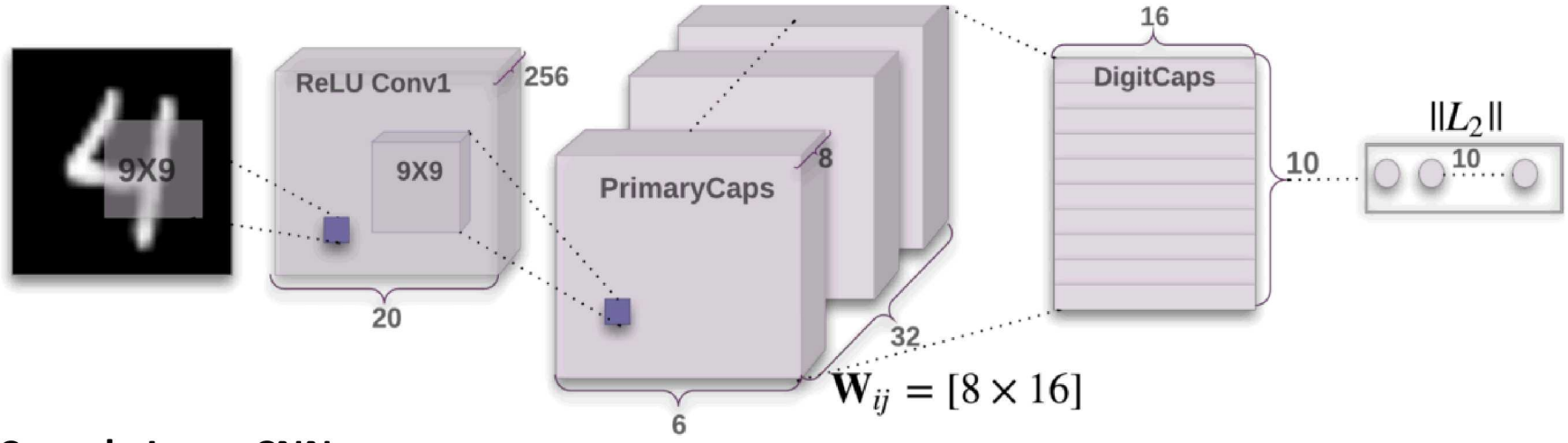


Combination = Convolutional Neural Network



Comparison of Traditional CNNs to CapNets (Cont.)

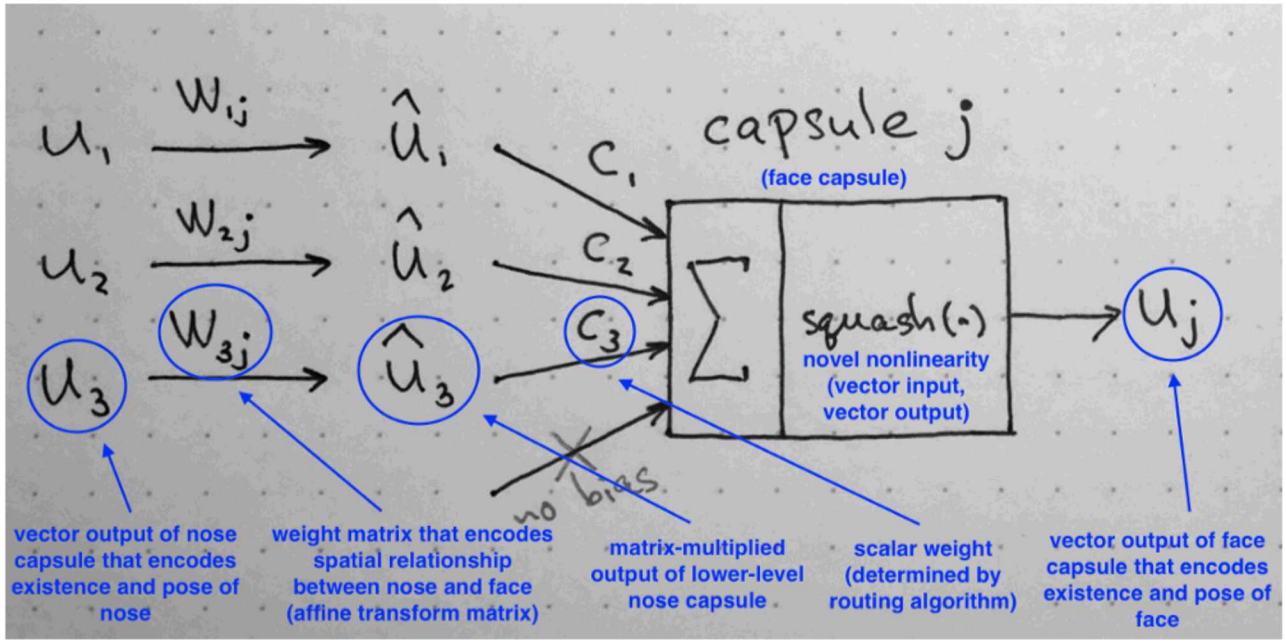
Capsule Network



Capsule Layer CNN

1. Changes forward propagation layer.
    1. RELU + Pooling -> Primary Capsule Layer
    2. Primary capsule layer routes to secondary capsule layer.
    3. Routing weights learned from "routing-by-agreement" operation.
2. Changes loss function: $L_c = T_c \max(0, m^+ - ||v_c||)^2 + \lambda(1 - T_c) \max(0, ||v_c|| - m^-)^2$ where T_c is correct one-hot encoding, m^+ = 0.9, $m^- = 0.1$, $||v_c||$= length of capsule vector at category c, and λ is a user-selectable weight

Routing By Agreement



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 $\mathbf{v}_j$

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

additional "squashing" | unit scaling

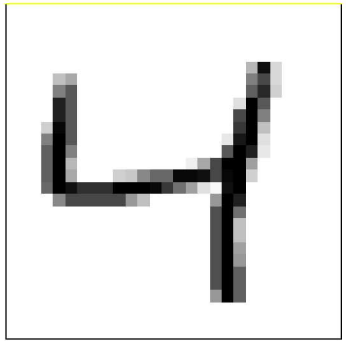
Notes:

- Overview of capsule networks
- Occurs during forward pass through network
- Show how implemented on MNIST network
- MNIST is database of ~50,000 hand written digits that are 28x28
- Not as complicated as it seems

Capsule Network: MNIST Example

Input Layer

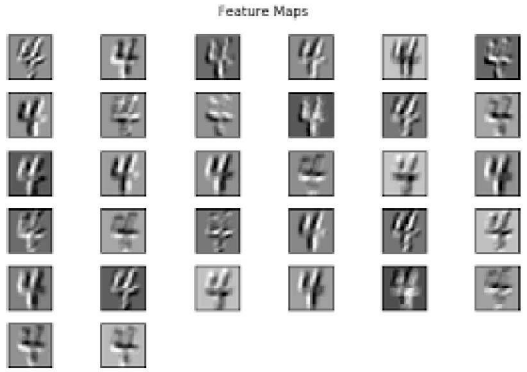
28 x 28 Input



9x9 kernel; stride 1; 256 filters

Convolutional Layer

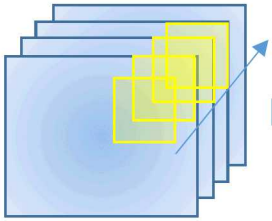
20 x 20 x 256 [height X width X filters]



9x9x256 kernel, stride 2,
32 filters in 8 “capsules”

Primary Capsule Layer

6x6x32x8 [height X width X filters X capsules]



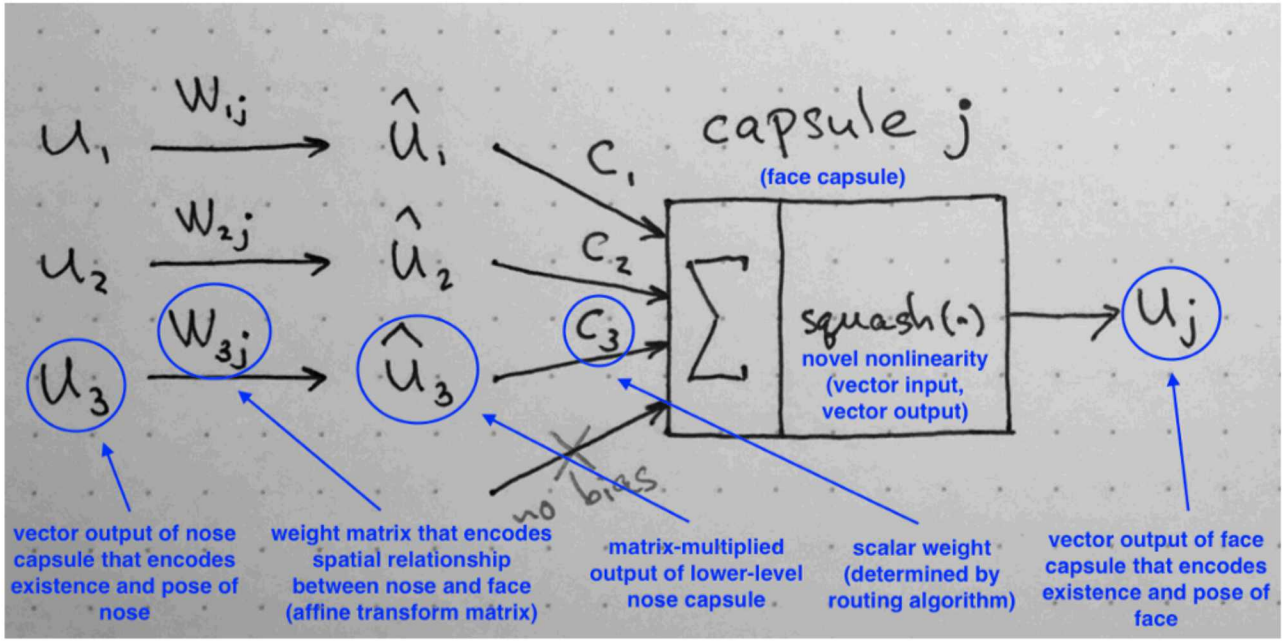
Reshape (6x6x32)x8
= 1152 vectors with 8
dimensions

Vector Input to Routing

Routing by Agreement

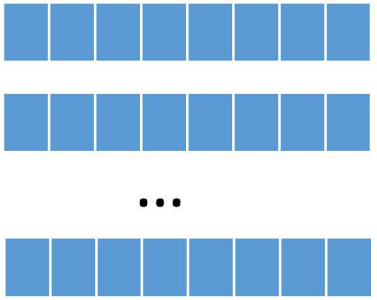
Secondary Capsule Layer(s)

Routing by Agreement: MNIST Example



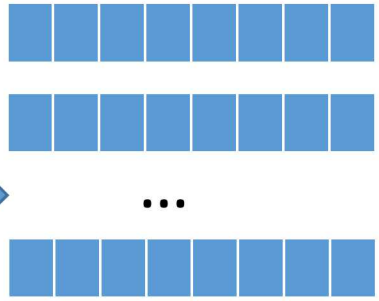
Loss function and categorical prediction

Output: 10 16 dimensional vectors (one vector for each digit)



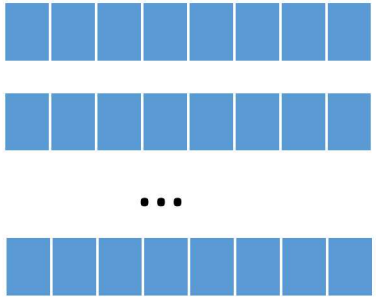
Calculate 1152 c_i weights by routing algorithm

1152 16 dimensional vectors;
$\hat{u}_i$ in picture above



Multiply each by 8x16 matrix;
w_{ij} in picture above

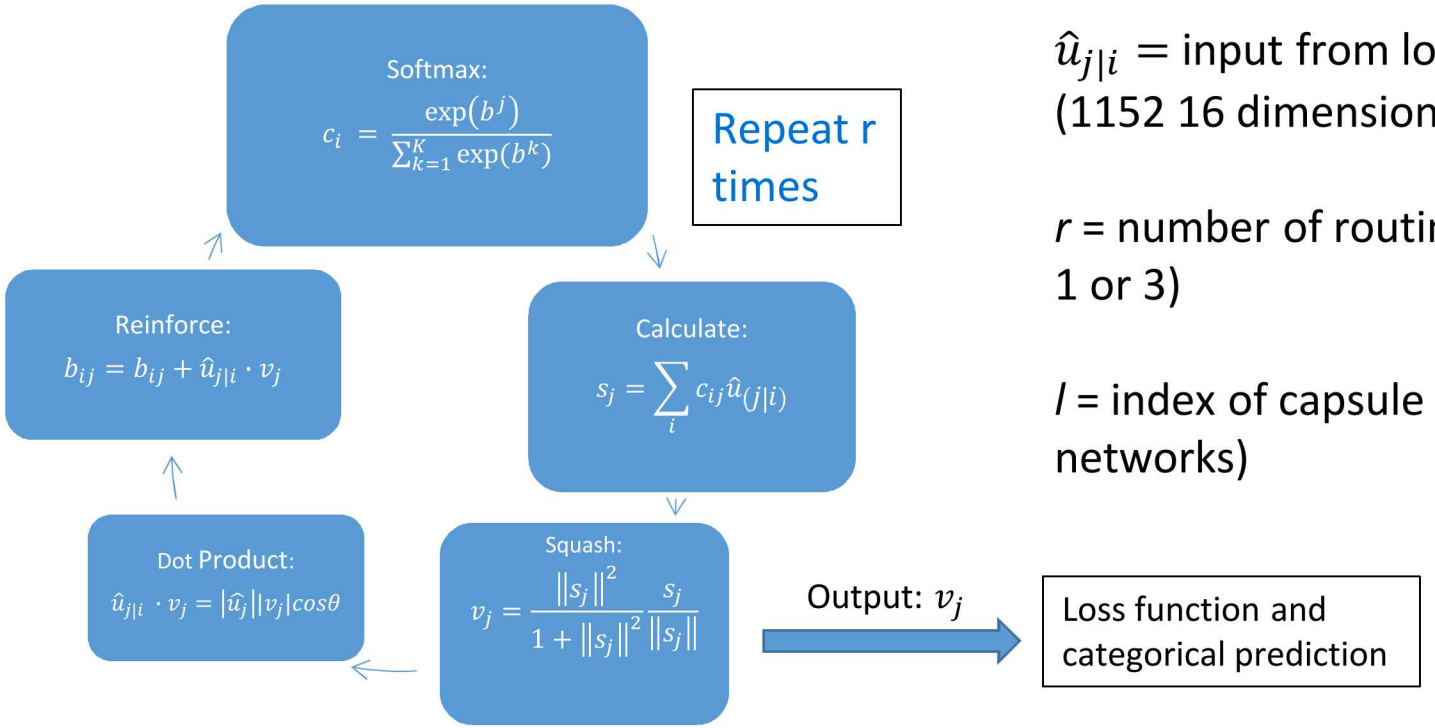
1152 8 dimensional vectors;
u_i in picture above



Routing Algorithm: MNIST Example

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 $\mathbf{v}_j$



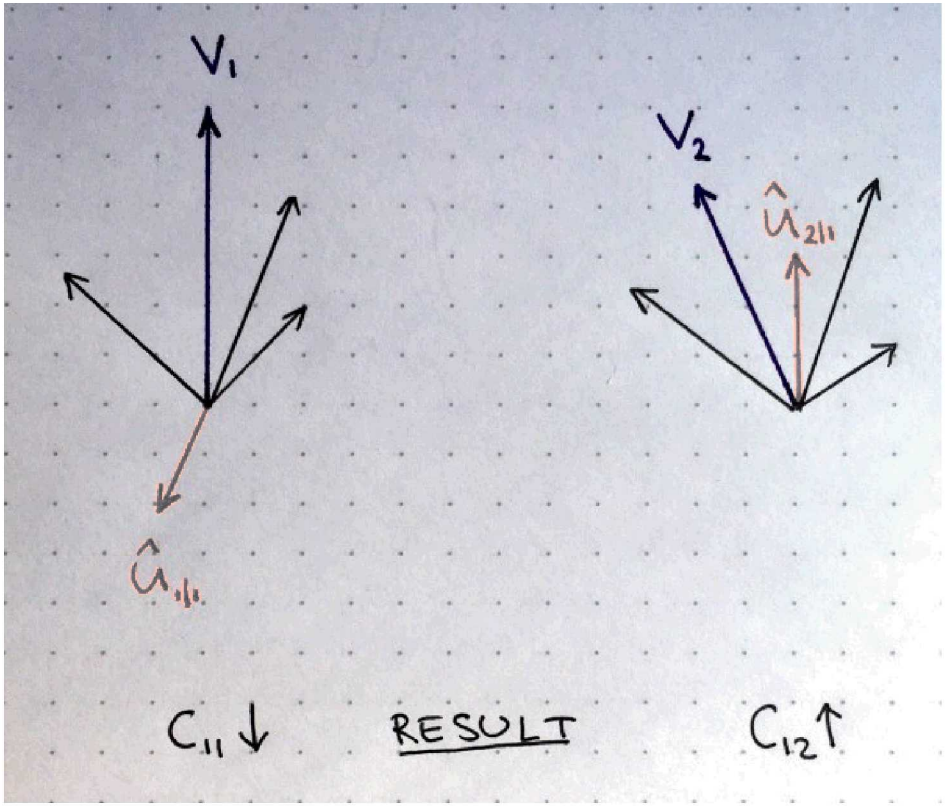
$\hat{u}_{j|i}$ = input from lower level capsules (1152 16 dimensional vectors)

r = number of routing iterations (typically 1 or 3)

l = index of capsule layer (for multicapsule networks)

Loss function and categorical prediction

Routing Algorithm: Intuition

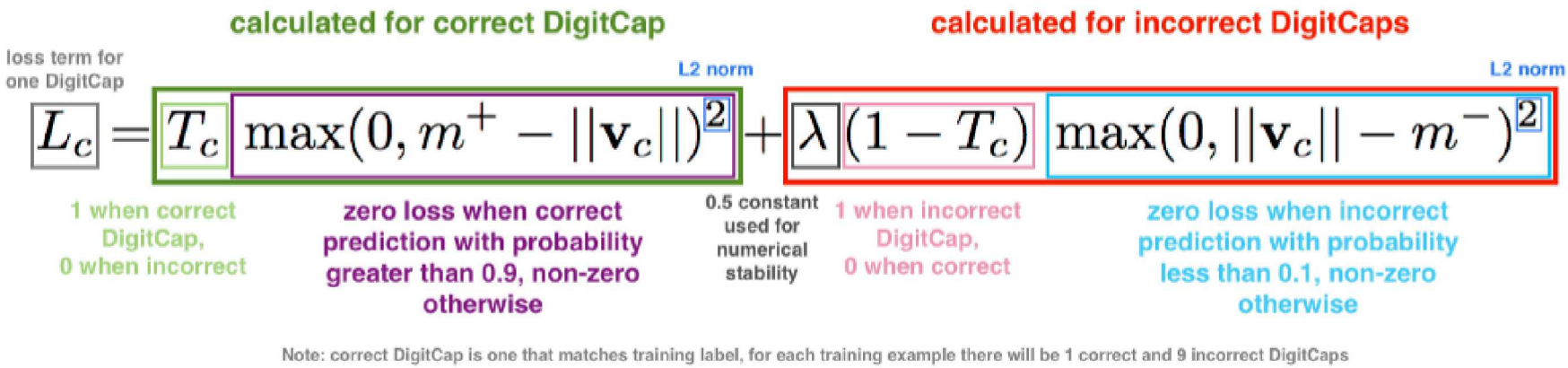


c_{12} is reinforced by routing algorithm because vector orientations agree

c_{11} is reinforced by routing algorithm because vector orientations disagree

Orange: Two inputs from lower level capsule
Purple: Output of two higher level capsules
Black: Remaining inputs from lower level capsules

Loss Function



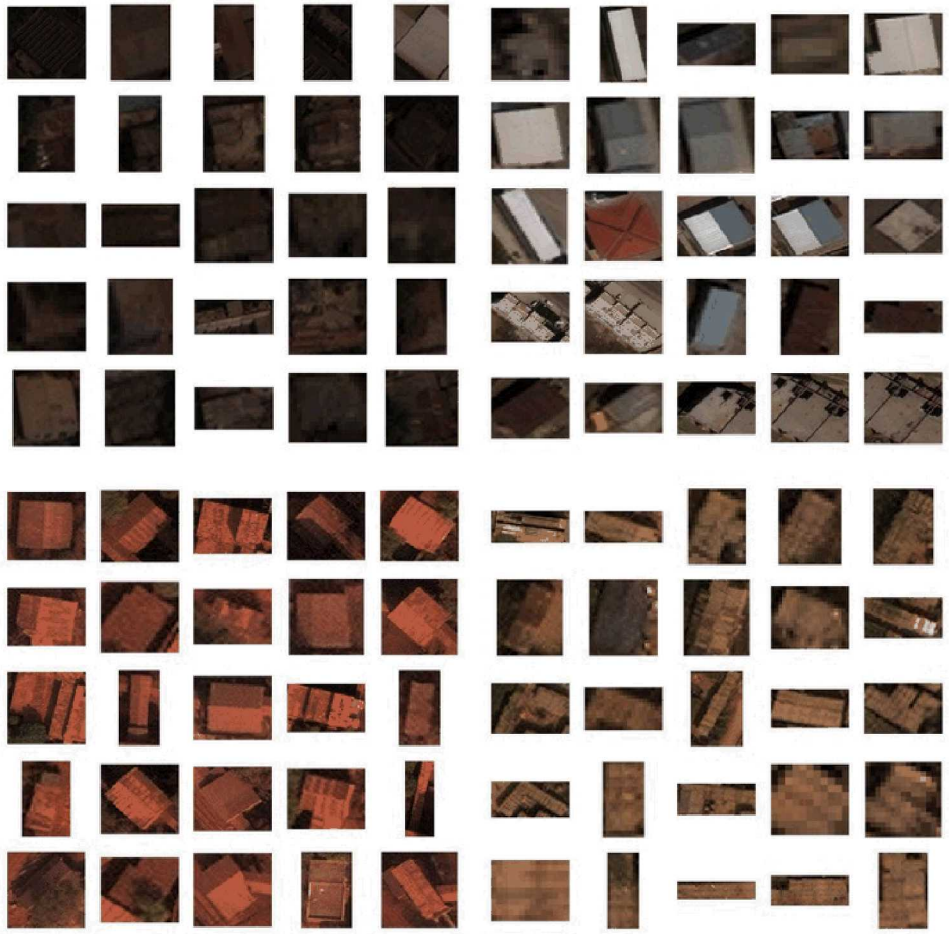
If the length of the vector in the correct capsule is > 0.9, loss is 0

If length of vector in the correct capsule is < 0.1, loss is 1

Description of the data used

Released to public from Defense Innovation Unit Experimental (DIUx) and National Geospatial-Intelligence Agency (NGA)

- 1 Million Object Instances
- 60 Classes
- 0.3 Meter Resolution
- Images labelled by humans
  - Amazon Mechanical Turk, etc.



Results

- 5 vehicle class subset of full dataset, ~3000 images.
  - Crane truck, dump truck, bulldozer, excavator, cement mixer
- Training overfits training dataset giving a high training accuracy, with 40% accuracy on withheld testing dataset.
  - 25% average precision reported in XView paper
    - Not quite apples-to-apples
    - Doing SSD on 300x300 chips, not tight chips
  - Will examine dropout layers
- Accuracy hampered by inaccurate labels in original dataset.
  - Student intern corrected labels
- 40% is still much better than chance
- Summer intern working on library implementation of CapNets in Tensorflow with plans for a PyTorch version.

CapNets Resources

- Web
  - Understanding Hinton's Capsule Networks. Parts I – IV, medium.com
  - Capsule Networks Are Shaking up AI — Here's How to Use Them, hackernoon.com
- Papers
  - Transforming Auto-encoders
  - Dynamic Routing Between Capsules
  - Matrix Capsules with EM Routing
    - Mentions that capsules are resistant to white box adversarial attack.
- Video
  - Capsule Networks, https://www.youtube.com/watch?v=pPN8d0E3900
- Code
  - Tensorflow implementation: https://github.com/naturomics/CapsNet-Tensorflow
  - PyTorch implementation https://github.com/gram-ai/capsule-networks


MACHINE LEARNING & DEEP LEARNING
MLDL