INTRODUCTION TO CAPSULENET

Fatemeh Saberi Khomami

OUTLINE

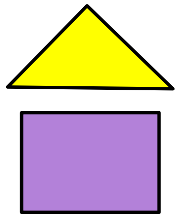
WHO IS GEOFFREY HINTON?

- Prof. Hinton is the great-grandson of George Boole - the mathematician who invented Boolean algebra.
- 1966 got interested in understanding “how does brain stores memory”
- Changed several majors in undergraduate at Cambridge University.
  Physiology and Physics → Philosophy → Psychology
- Received his PhD in Artificial Intelligence from Edinburgh in 1978
- Couldn’t find a job in Britain, Moved to California
- 1982, D. Rumelhart, G. Hinton, and R. Williams developed the Backprop algorithm, and published the paper into Nature in 1986

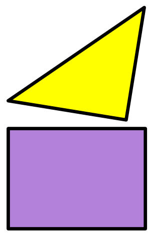
DRAWBACKS OF CNN

CNN is very good at capturing features that are present in the image but is weak at:

- Exploring the relative spatial relationship
- Exploring the relative size relationships
- Exploring the orientational characteristics



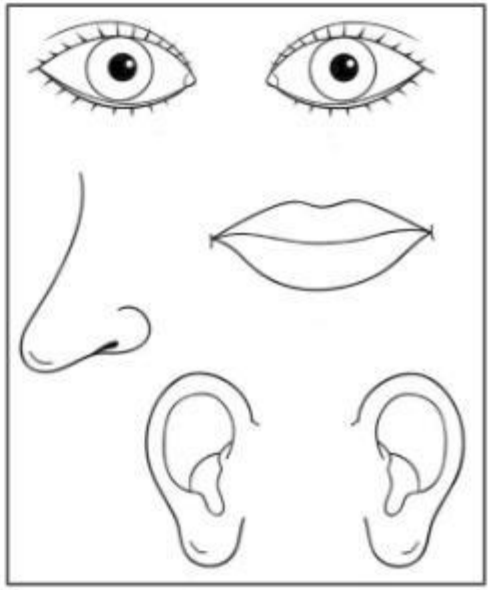
A house



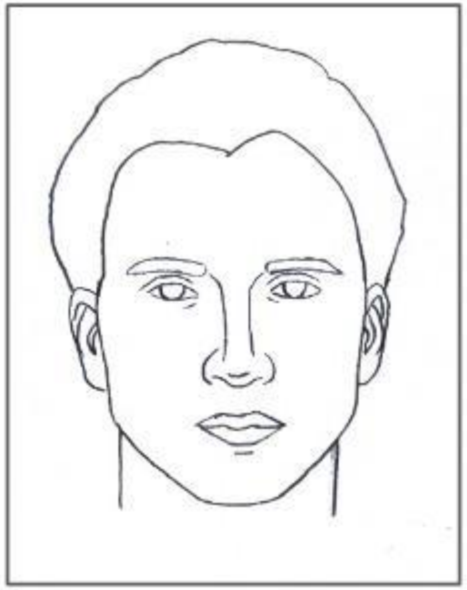
A boat

DRAWBACKS OF CNN

For example a CNN would label the left picture, a “Face”, while the spatial relationship of the face entities are not correct.



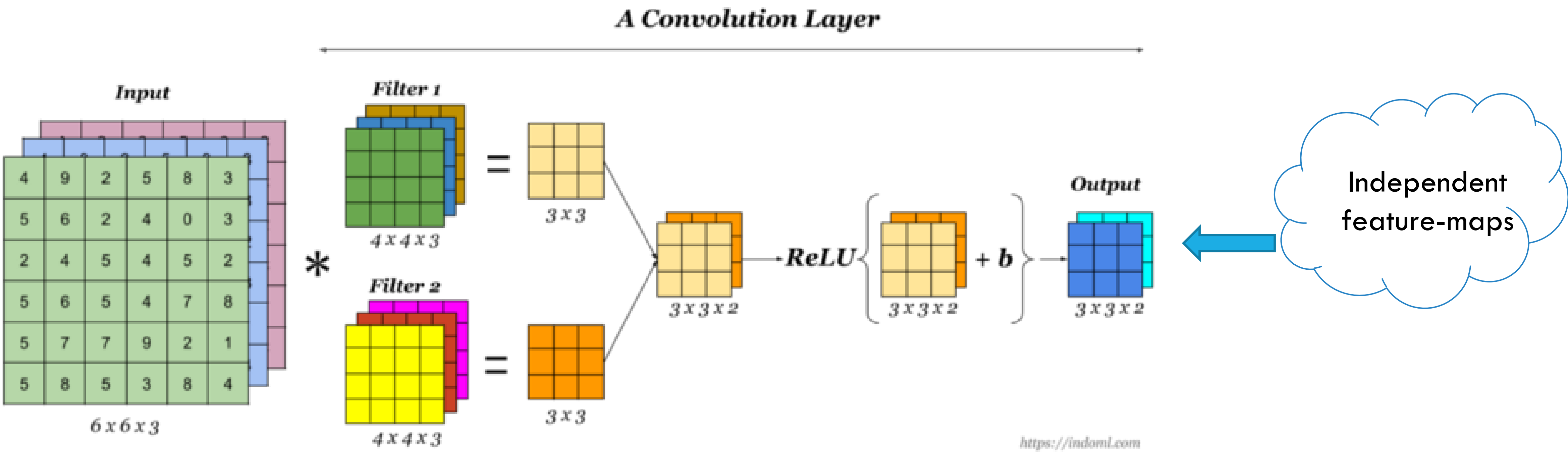
Face



Face

DRAWBACKS OF CNN

Due to the independency between the extracted features from the convolutional layers, the spatial relationship between entities will not be preserved.



DRAWBACKS OF CNN

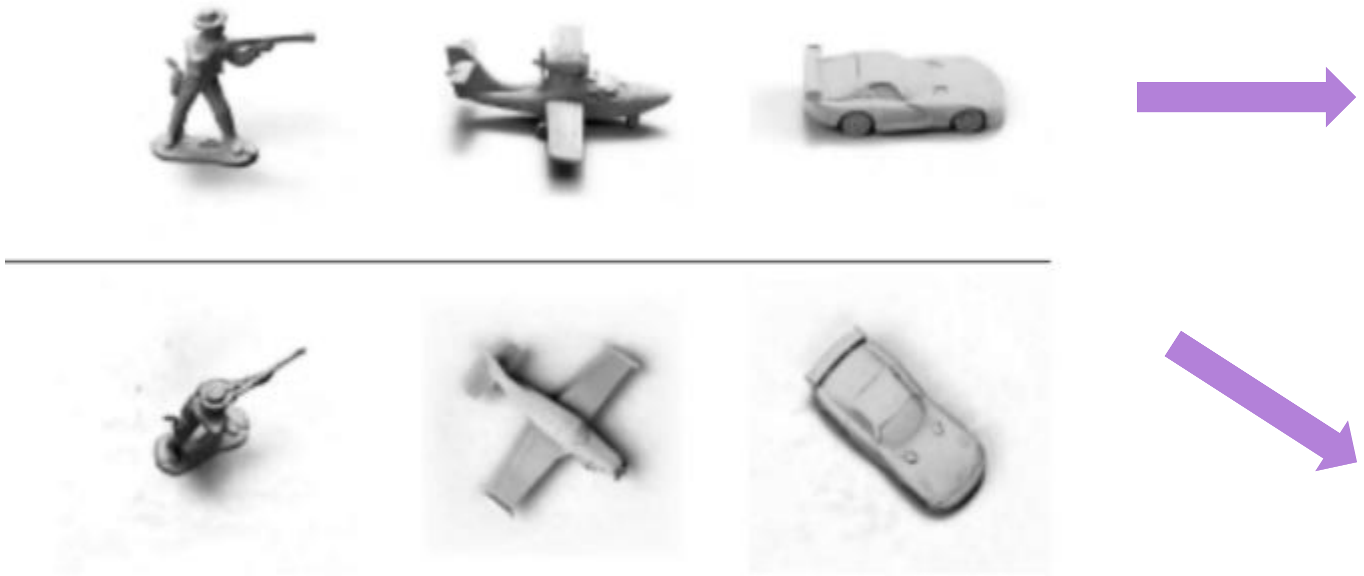
The key-problem here is:

Internal data representation of a CNN does not take into account important spatial hierarchies between simple and complex objects.

A PROPOSED SOLUTION

Hinton argues that in order to correctly do classification and object recognition, it is important to preserve hierarchical pose relationships between object parts.

When these relationships are built into the internal representation of data, the model can detect objects regardless of the viewpoint.



A PROPOSED SOLUTION: USE A CAPSULE!

A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part.

The length of the activity vector(capsule's output vector) $\Rightarrow$ Probability that the entity exists

Instantiation parameters $\Rightarrow$ Object's orientations

Capsule takes a vector as input and outputs a vector

HOW DOES A CAPSULE WORK?

We are replacing:

CNN's scalar-output features detectors with vector-output capsules

CNN's max-pooling with routing-by-agreement

But what is routing-by-agreement ?

It's a process that assists each active capsule to choose an appropriate parent in the next/above capsule layer. (will assign a part to a whole)

HOW DOES A CAPSULE WORK?

Notation:

Assume capsule j,

v_j is the capsule j's output

s_j is the capsule j's total input

Assume capsule i in the below layer of capsule j,

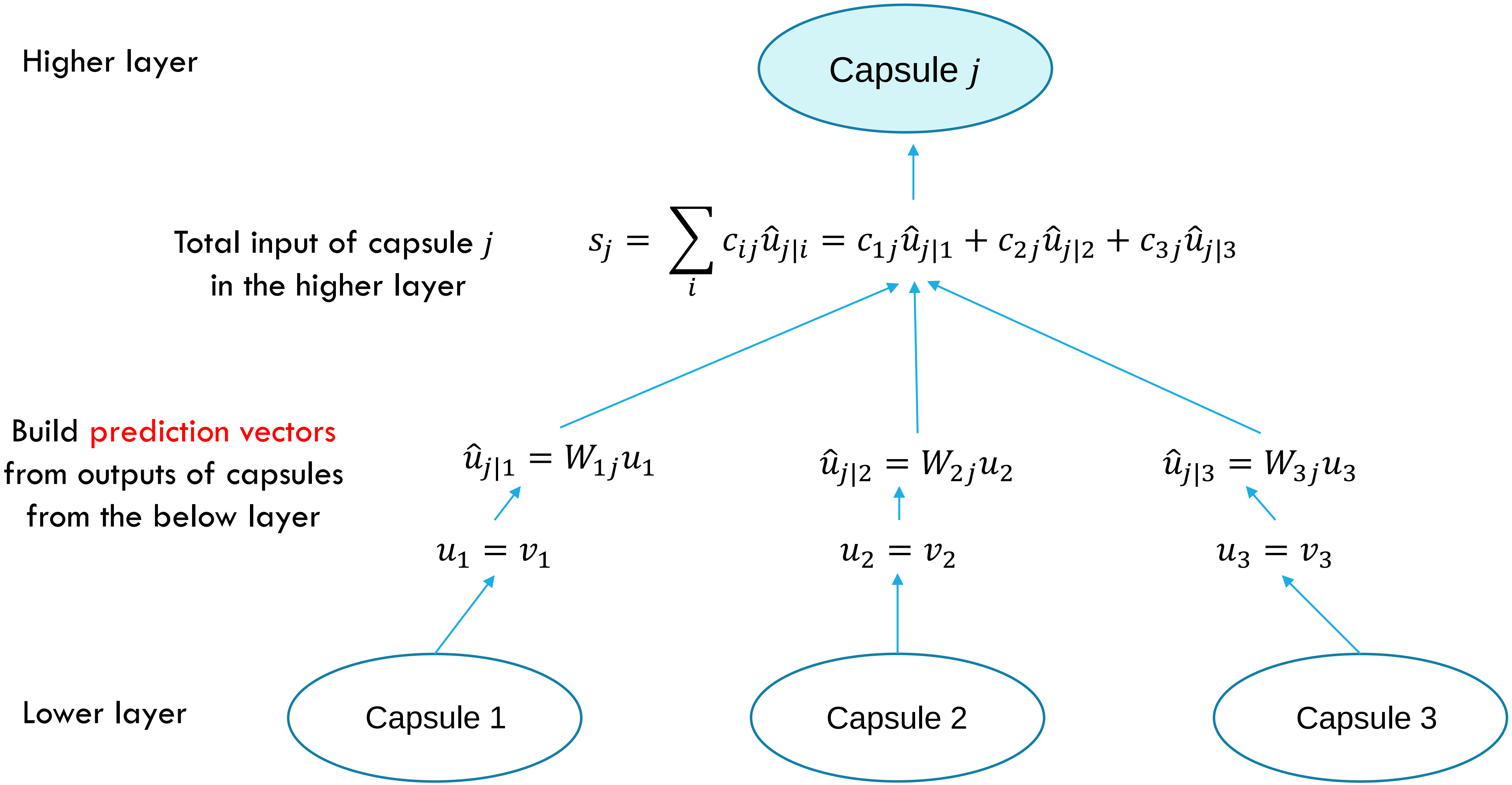
u_i is the capsule i's output

$\hat{u}_{j|i}$ is the prediction vector from the capsule i and is computed by: $\hat{u}_{j|i} = W_{ij} u_i$

c_{ij} is the coupling coefficients that are determined by the iterative dynamic routing process

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

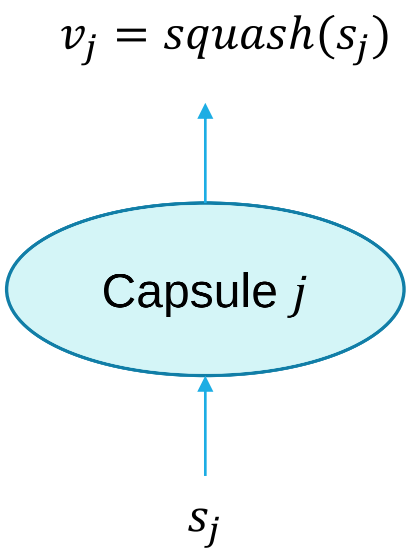
INPUT OF A CAPSULE



OUTPUT OF A CAPSULE

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

Squashing is a non-linear function that ensures the length of the output will always be in $(0,1)$. Hence, it's suitable to represent the presence probability of an entity in the image.



UPDATING THE COUPLING COEFFICIENTS

The coupling coefficients are iteratively refined by measuring the agreement between the current output v_j and the prediction made by capsule i in the below layer.

After each iteration, we update b_{ij} which is the constructor of c_{ij}

After each iteration: $b_{ij} = b_{ij} + \hat{u}_{j|i} \, . \, v_j$

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

OVERVIEW OF THE ROUTING ALGORITHM

Procedure 1 Routing algorithm.

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

PREDICTION VECTORS

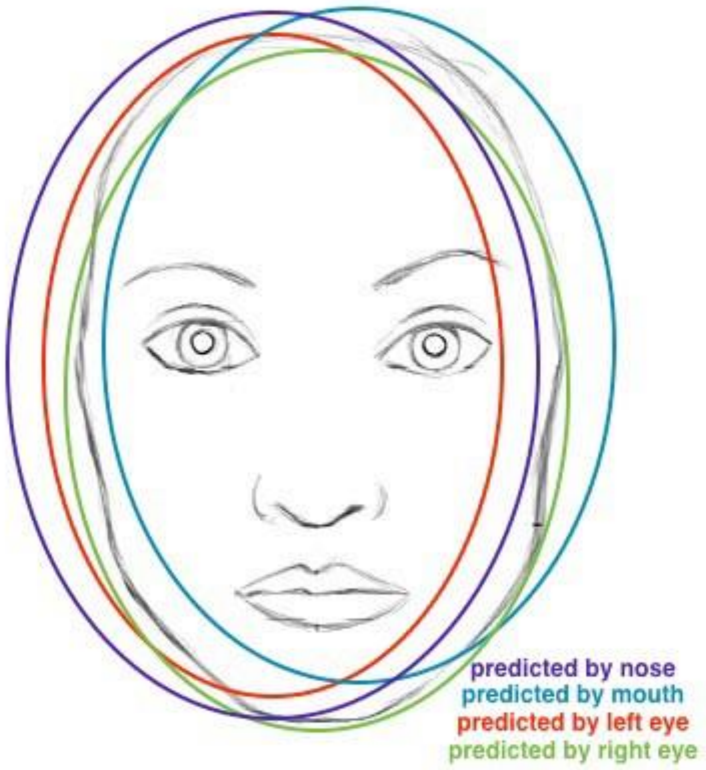
Remember prediction vectors ?

$\hat{u}_{j|i} = W_{ij} u_i$

W_{ij}s encode important spatial and other relationships between lower level features and higher level features.

Each of W_{ij}s are constructing $\hat{u}_{j|i}$s that will predict position of the higher level feature.

PREDICTION VECTORS



MARGIN LOSS FOR OBJECT EXISTENCE

We calculate the loss for object's capsule as follows:

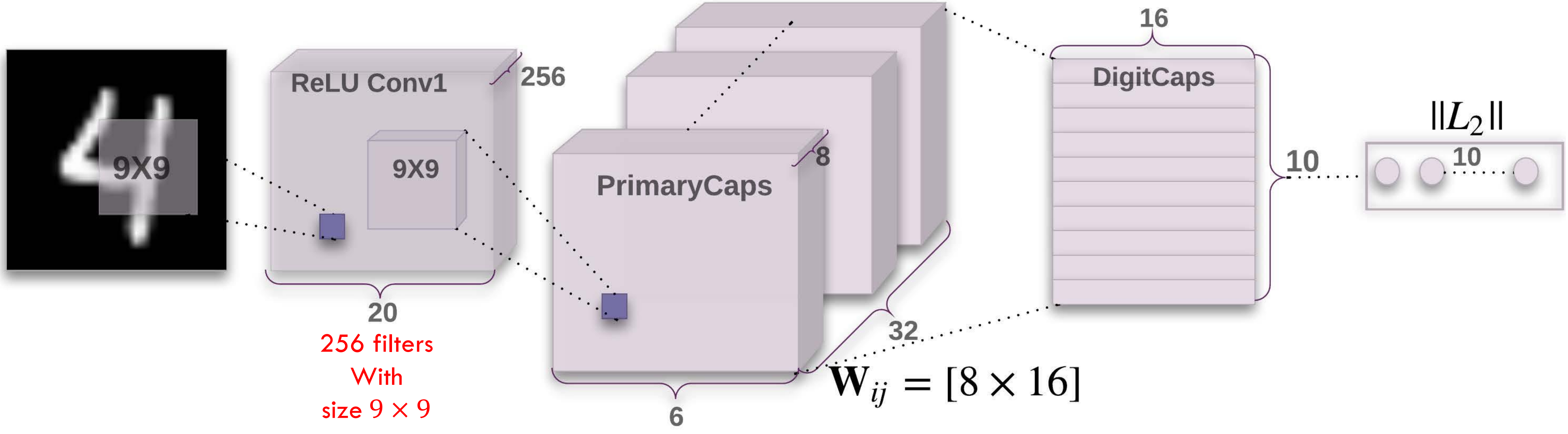
$$T_k = \begin{cases} 1 & \text{If the class k (object) is present} \\ 0 & \text{otherwise} \end{cases}$$

$$m^+ = 0.9 \;\&\; m^- = 0.1$$

$$L_k = T_k \,.\max(0, m^+ - \|v_k\|)^2 + \lambda(1 - T_k) \,.\max(0, \|v_k\| - m^-)^2$$

CAPSNET ARCHITECTURE-MNIST DATASET

The length of the activity vector of each capsule in DigitCaps layer indicates presence of an instance of each class and is used to calculate the classification loss.



The result is
256 feature-maps
with dimensions 20×20

CAPSNET ARCHITECTURE-PRIMARY CAPS LAYER

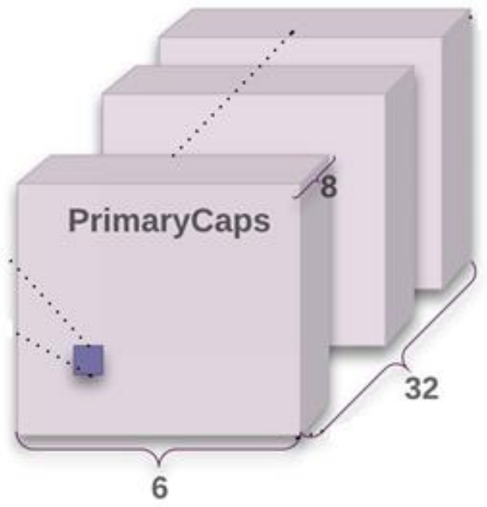
This layer has 32 primary capsules whose job is to take basic features detected by the Conv1 layer and produce combinations of the features.

This layer has 32 primary capsules.

Each of the 32 capsules applies eight $9 \times 9 \times 256$ convolutional kernels with stride 2 on the $20 \times 20 \times 256$ output of Conv1, and produce $6 \times 6 \times 8$ tensors. So, the output volume of this layer would have a shape of $6 \times 6 \times 8 \times 32$.

In total Primary Capsules has $[32 \times 6 \times 6]$ capsule outputs.

(each output is an 8D vector)

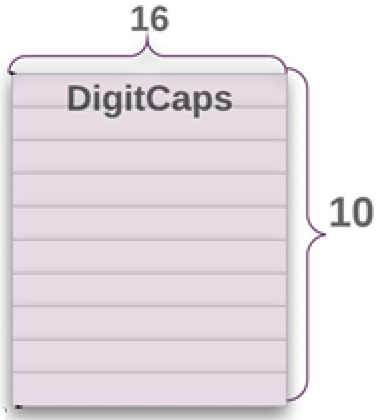


CAPSNET ARCHITECTURE-DIGIT CAPS LAYER

The final Layer (DigitCaps) has one 16D capsule per digit class and each of these capsules receives input from all the capsules in the layer below.

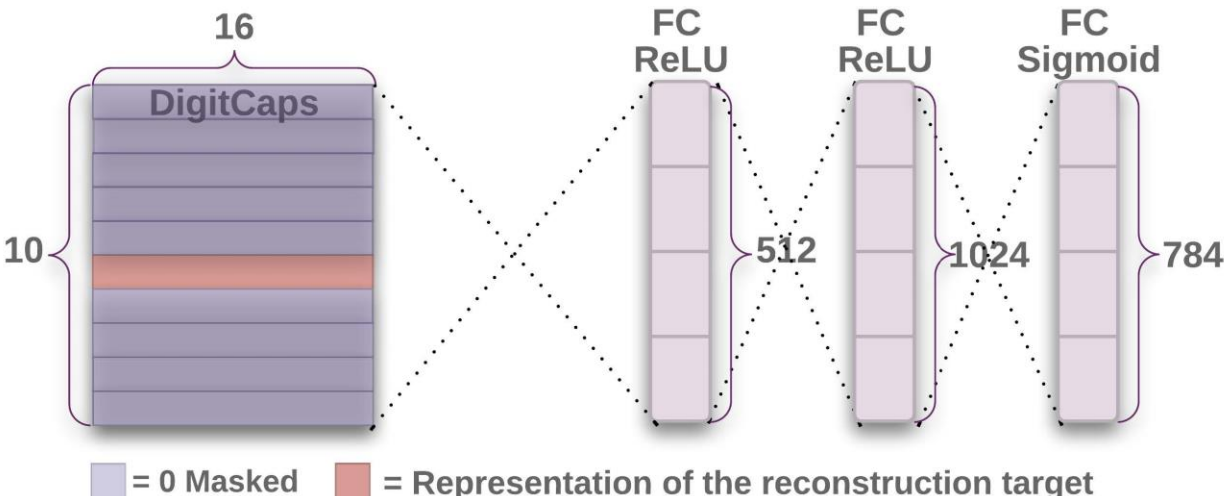
Each capsule in Digit Caps takes $6 \times 6 \times 32$ 8-dimensional vectors as inputs in total.

Each of these input vectors gets their own 8×16 weight matrix that maps 8-dimensional input space to the 16-dimensional capsule output space.



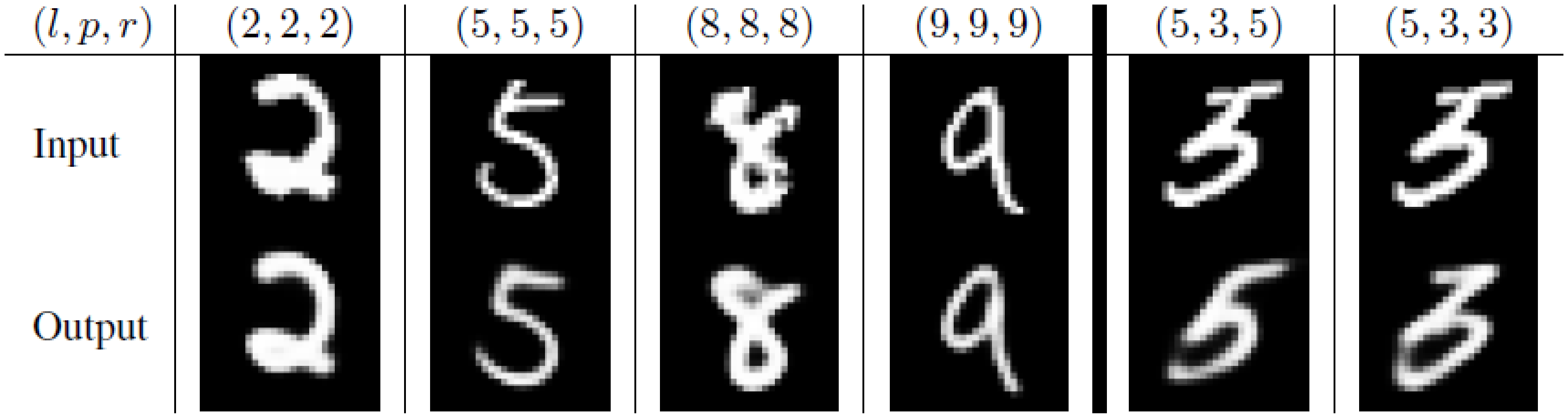
CAPSNET ARCHITECTURE-THE DECODER

Decoder takes a 16-dimensional vector from the correct DigitCap and learns to decode it into an image of a digit. Decoder takes the output of the correct DigitCap as input and learns to recreate an 28 by 28 pixels image. Decoder forces capsules to learn features that are useful for reconstructing the original image.



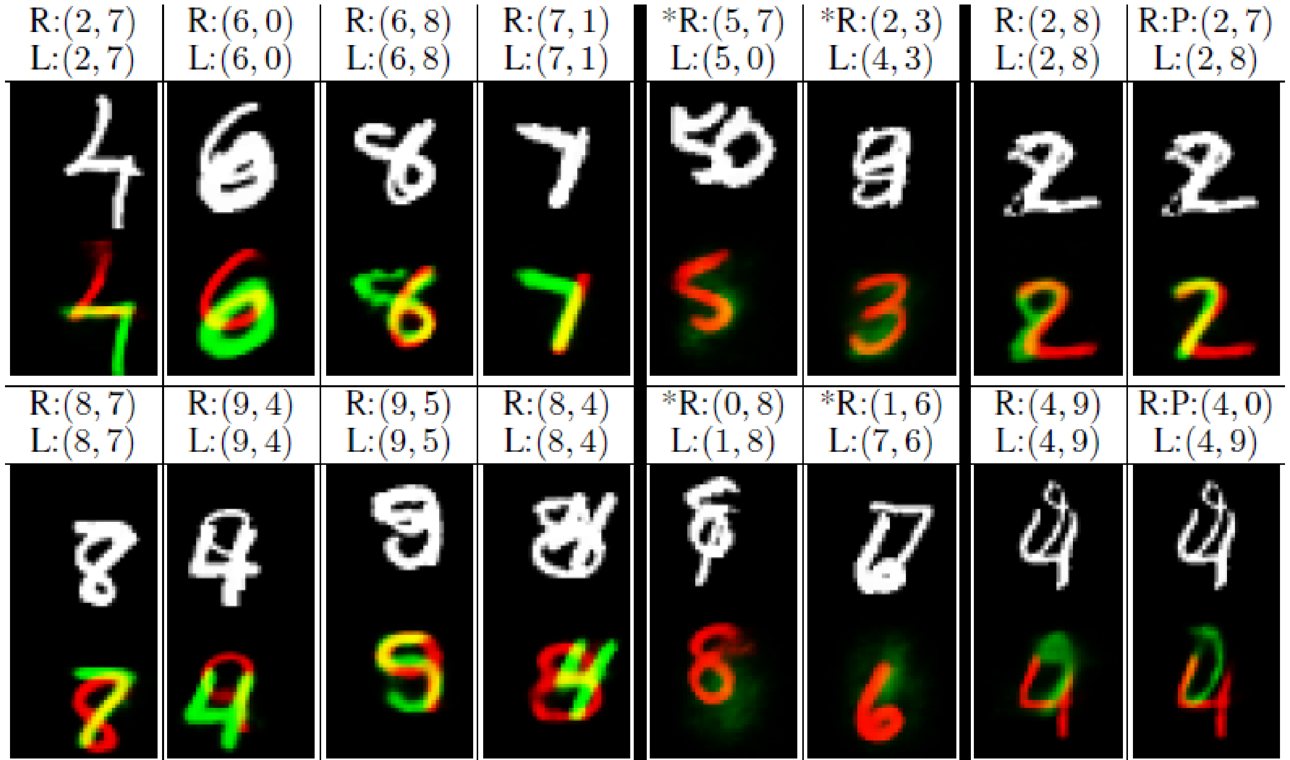
after reshaping
gives back
a 28x28 decoded image

CAPSNET ARCHITECTURE- RECONSTRUCT MNNIST



SEGMENTING HIGHLY OVERLAPPING DIGITS

Dynamic routing can be viewed as a parallel attention mechanism that allows each capsule at one level to attend to some active capsules at the level below and to ignore others. This should allow the model to recognize multiple objects in the image even if objects overlap.



BENCHMARK

The test error on MNIST and MultiMNIST of different architectures are as follows:

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**

CONCLUSION

Capsules preserve the intrinsic spatial relationship of the data, and perform better than CNN in:

- change in viewpoint
- image segmentation especially when we have overlapping objects
- detect logical relationship (position, orientation, size, etc.) of entities of an object

CapsNet compares to CNN, can perform all the above advantages with less data.

REFERENCES

1) S. Sabour, N. Frosst and G.E. Hinton, "Dynamic Routing Between Capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866

2) G.E. Hinton, S. Sabour and N. Frosst, "Matrix Capsules with EM routing," in *International Conference on Learning Representations Workshop*, 2018

3) M. Pechyonkin, "Understanding Hinton's Capsule Networks", in https://pechyonkin.me, 2018

4) A. Géron, "Capsule Networks (CapsNets)", https://www.youtube.com/watch?v=pPN8d0E3900, 2017

Thank you