Carleton
UNIVERSITY
Canada's Capital University

Introduction to Capsule Networks

Vasileios Lioutas
School of Computer Science
vasileios.lioutas@carleton.ca

Table of contents

Why Capsule Networks?

Capsule Networks by Hinton

Dynamic Routing Between Capsules

Sara Sabour **Nicholas Frosst**

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

Abstract

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. We use the length of the activity vector to represent the probability that the entity exists and its orientation to represent the instantiation parameters. Active capsules at one level make predictions, via transformation matrices, for the instantiation parameters of higher-level capsules. When multiple predictions agree, a higher level capsule becomes active. We show that a discrimininatively trained, multi-layer capsule system achieves state-of-the-art performance on MNIST and is considerably better than a convolutional net at recognizing highly overlapping digits. To achieve these results we use an iterative routing-by-agreement mechanism: A lower-level capsule prefers to send its output to higher level capsules whose activity vectors have a big scalar product with the prediction coming from the lower-level capsule.

Published as a conference paper at ICLR 2018

MATRIX CAPSULES WITH EM ROUTING

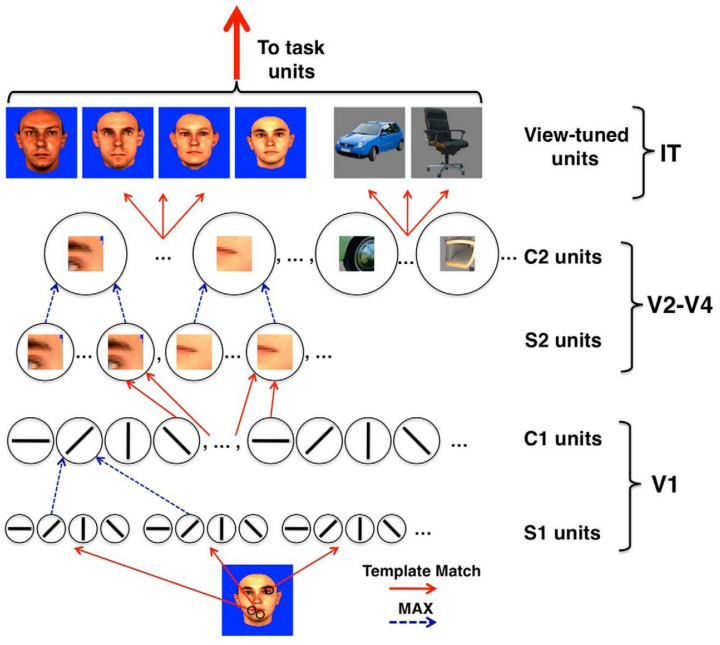
Geoffrey Hinton, Sara Sabour, Nicholas Frosst
Google Brain
Toronto, Canada
{geoffhinton, sasabour, frosst}@google.com

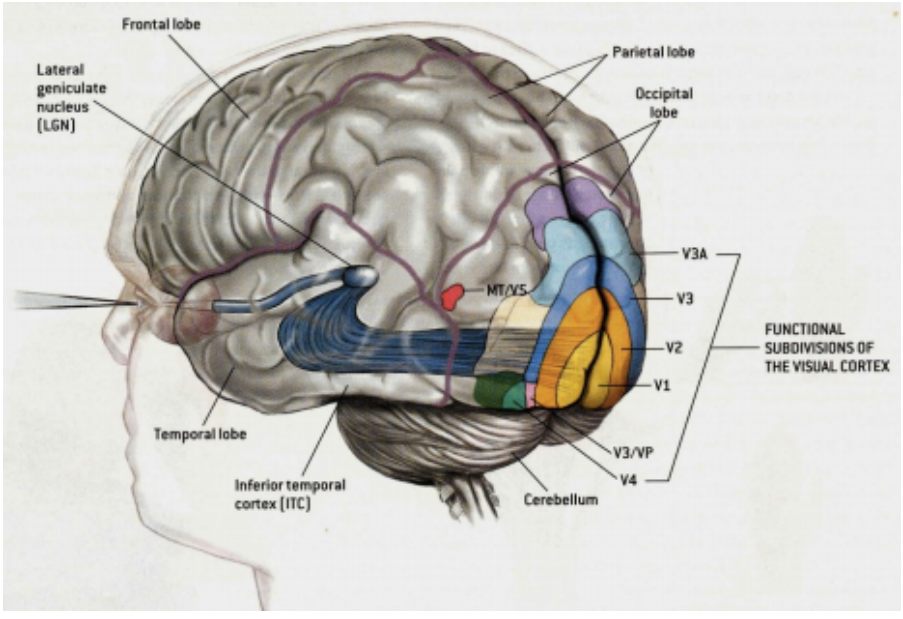
ABSTRACT

A capsule is a group of neurons whose outputs represent different properties of the same entity. Each layer in a capsule network contains many capsules. We describe a version of capsules in which each capsule has a logistic unit to represent the presence of an entity and a 4x4 matrix which could learn to represent the relationship between that entity and the viewer (the pose). A capsule in one layer votes for the pose matrix of many different capsules in the layer above by multiplying its own pose matrix by trainable viewpoint-invariant transformation matrices that could learn to represent part-whole relationships. Each of these votes is weighted by an assignment coefficient. These coefficients are iteratively updated for each image using the Expectation-Maximization algorithm such that the output of each capsule is routed to a capsule in the layer above that receives a cluster of similar votes. The transformation matrices are trained discriminatively by backpropagating through the unrolled iterations of EM between each pair of adjacent capsule layers. On the smallNORB benchmark, capsules reduce the number of test errors by 45% compared to the state-of-the-art. Capsules also show far more resistance to white box adversarial attacks than our baseline convolutional neural network.

Hierarchical model of the visual system

HMax Model, Riesenhuber and Poggio (1999)



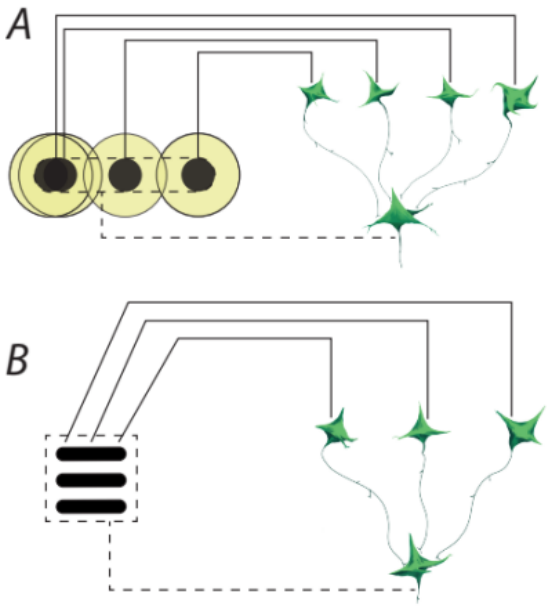


dotted line selects max pooled features from lower layer

Slides heavily inspired by Charles Martin presentation

Hierarchical model of the visual system

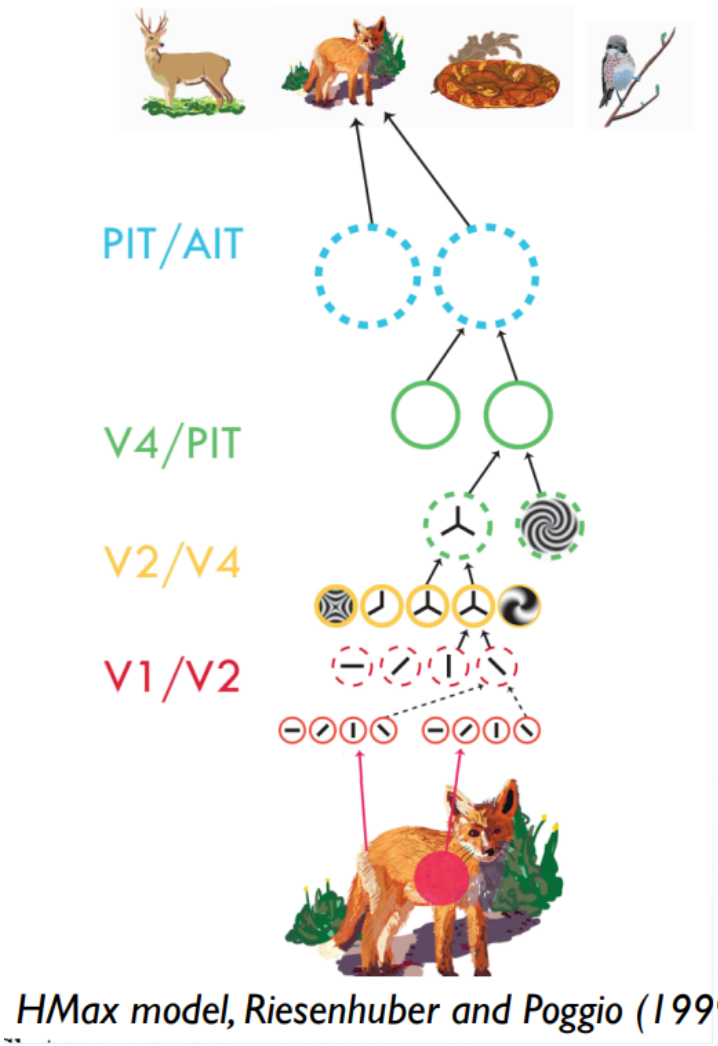
Pooling proposed by Hubel and Wiesel in 1962



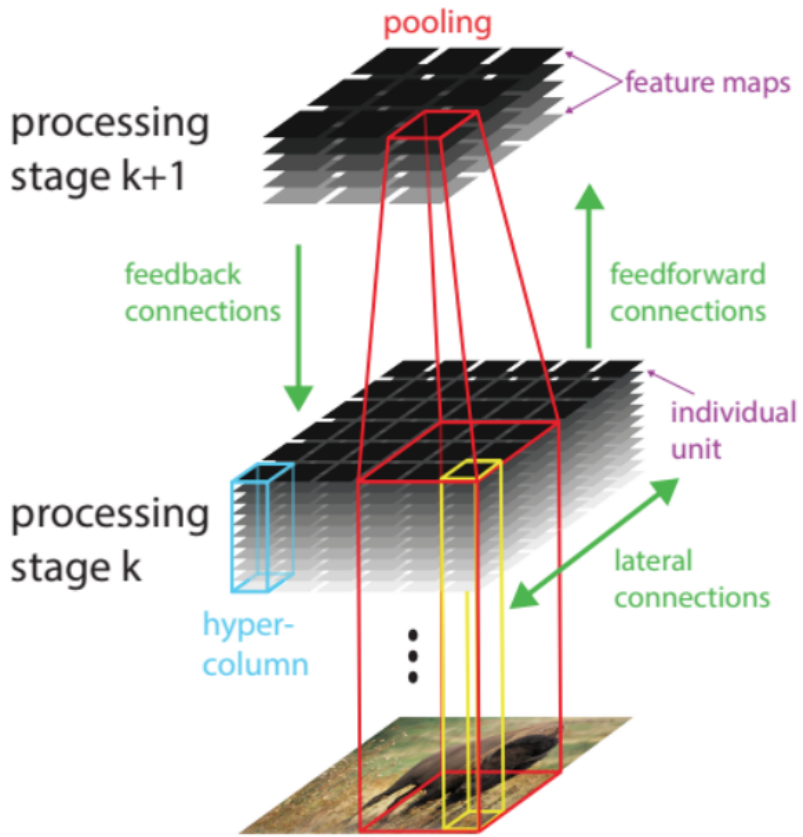
A. Receptive field (RF) of simple cell (green) formed by pooling over (center-surround) cells (yellow) in the same orientation row
B. RF of complex cell (green) formed by pooling over simple cells.

Slides heavily inspired by Charles Martin presentation

Hierarchical model of the visual system



HMax model, Riesenhuber and Poggio (1999)



ConvNets resemble hierarchical models (but notice the hyper-column)

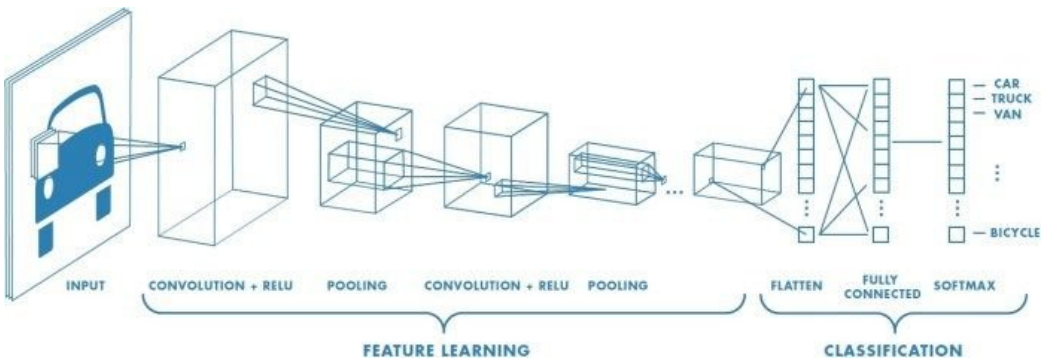
Slides heavily inspired by Charles Martin presentation

The problem with CNNs and Max-Pooling

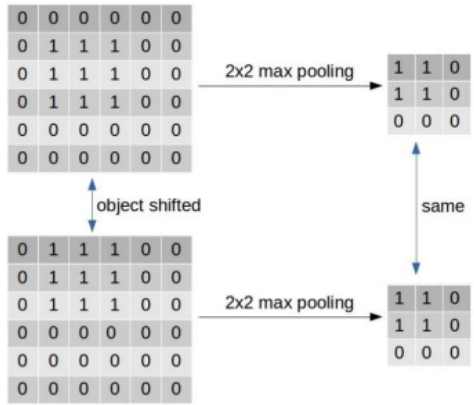
The brain embeds things in rectangular space (?), then:

- Translation is easy; Rotation is hard
- Experiment: time for mind to process rotation ∼ amount

ConvNets:



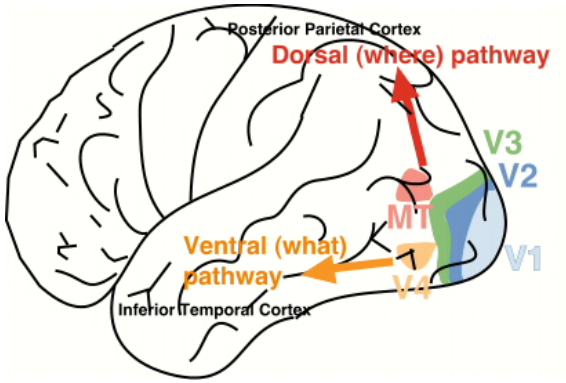
- The pooling operation **loses precise spatial relationships** between higher-level objects
- Pooling introduced small amounts of **crude translational invariance** at each level
- No *explicit* **pose** (orientation) information
- Can not distinguish left from right



A vision system needs to use the same knowledge at all locations in the image

Slides heavily inspired by Charles Martin presentation

2 streams hypothesis: what and where



Ventral: what objects are

Dorsal: where objects are in space

idea dates back to 1968

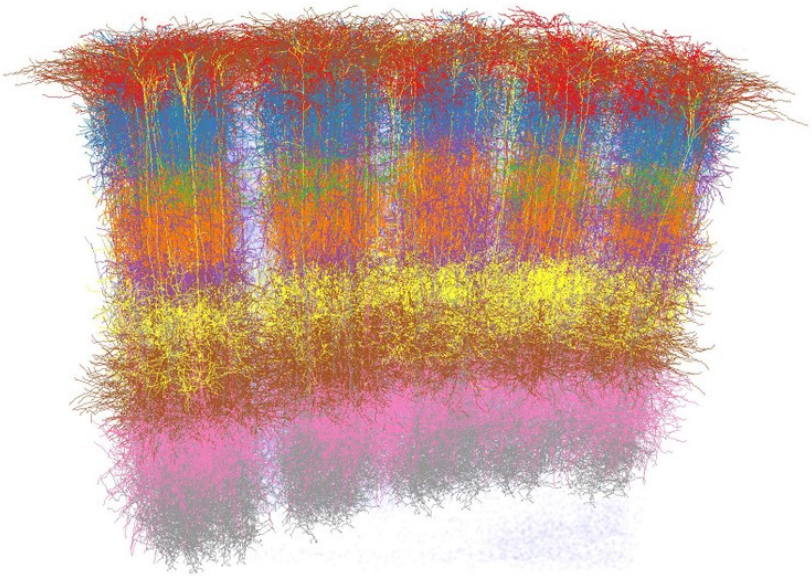
How do we know? Neurological disorders

Simultanagnosia: can only see one object at a time

lots of other evidence as well

Slides heavily inspired by Charles Martin presentation

Cortical Microcolumns

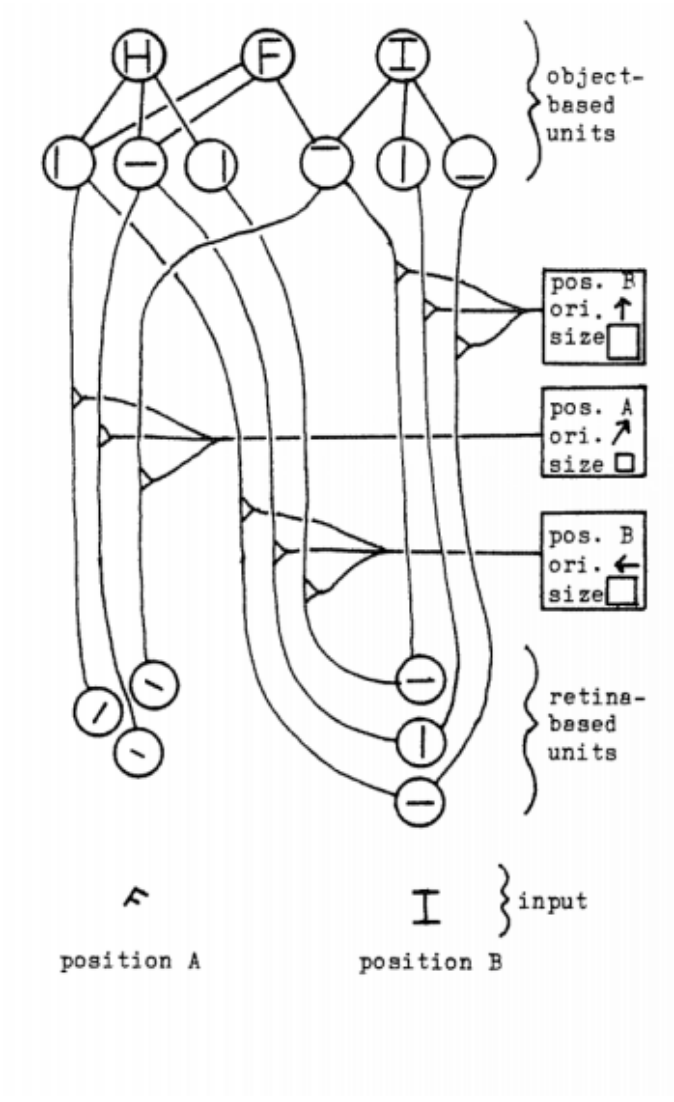


- Column through cortical layers of the brain 80-120 neurons (2X long in V1) share the same receptive field
- Capsules may encode: orientation, scale, velocity, color, etc.

part of Hubel and Wiesel, Nobel Prize 1981

Slides heavily inspired by Charles Martin presentation

Canonical object based frames of reference: Hinton 1981



In: Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, B.C. Canada, 1981

A PARALLEL COMPUTATION THAT ASSIGNS CANONICAL OBJECT-BASED FRAMES OF REFERENCE

Geoffrey E. Hinton

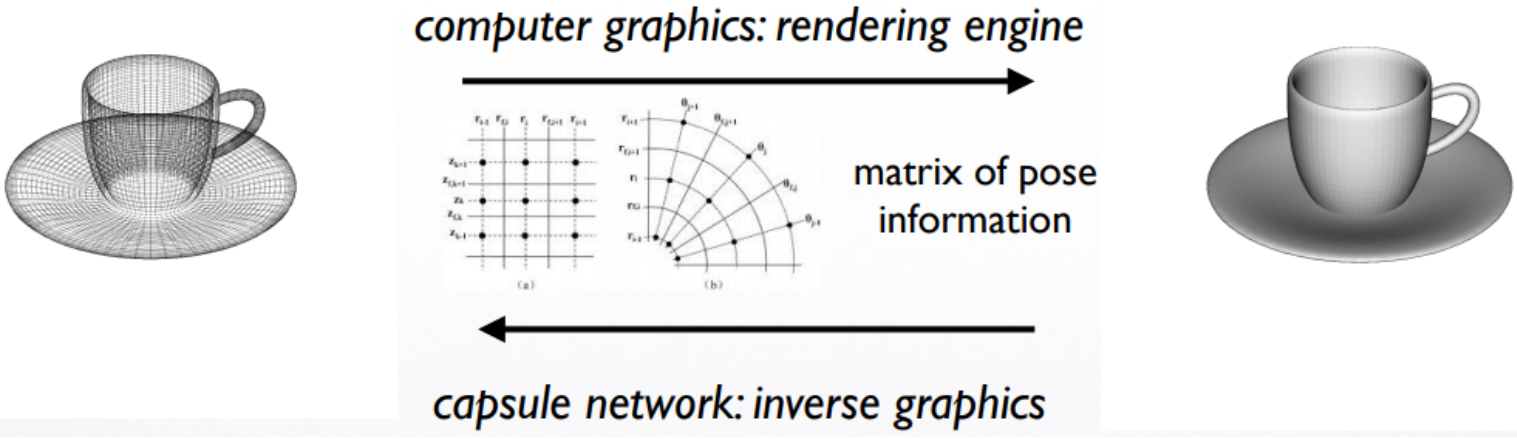
MRC Applied Psychology Unit
Cambridge, England

A kind of inverse computer graphics

Hinton has been thinking about this a long time

Slides heavily inspired by Charles Martin presentation

Inverse Computer Graphics



Hinton proposes that our brain does a kind-of inverse computer graphics transformation.

Slides heavily inspired by Charles Martin presentation

Invariance and Equivariance

- **Invariance** makes a classifier tolerant to small changes in the viewpoint. The idea of pooling is that it creates “summaries” of each sub-region. It also gives you a little bit of positional and translational invariance in object detection. This invariance also leads to triggering false positive for images which have the components of a recognized object but not in the correct order.
- **Equivariance** is invariance under a Symmetry and Transformations (translation, rotation, reflection and dilation). It makes a classifier understand the rotation or proportion change and adapt itself accordingly so that the spatial positioning inside an image, including relationships with other components, is not lost.

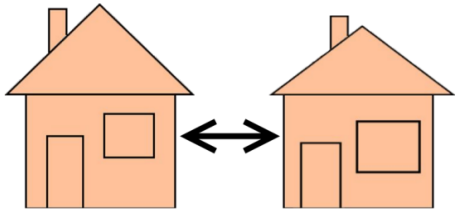


Figure 1: Useful Invariance

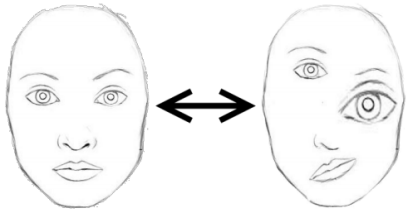


Figure 2: Problematic Invariance

As we discussed before, max pooling provides spatial Invariance, but Hinton argues that we need spatial Equivariance.

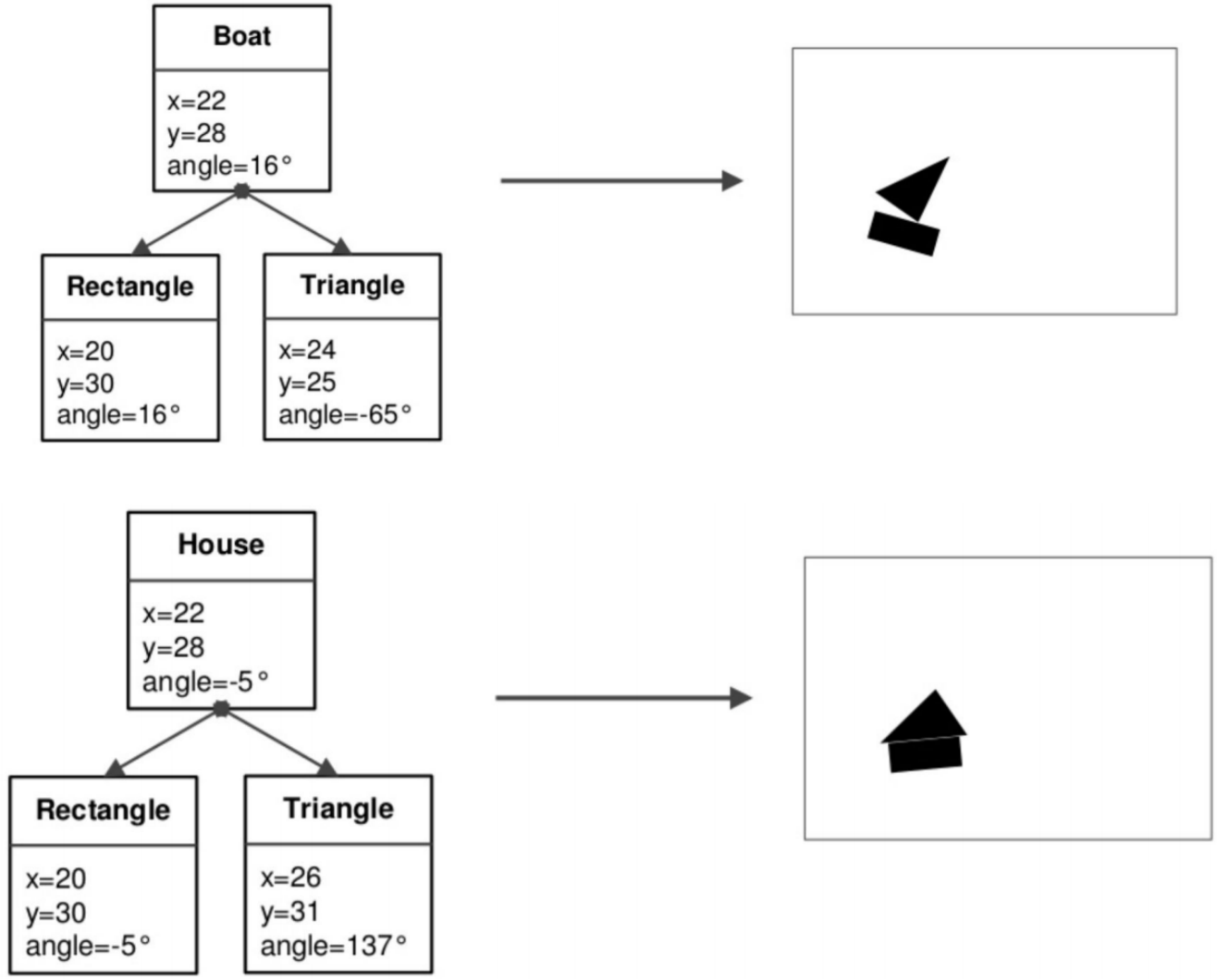
What is a Capsule and how does it work?

Capsule

Instead of aiming for viewpoint invariance in the activities of "neurons" that use a single scalar output to summarize the activities of a local pool of replicated feature detectors, artificial neural networks should use local "capsules".

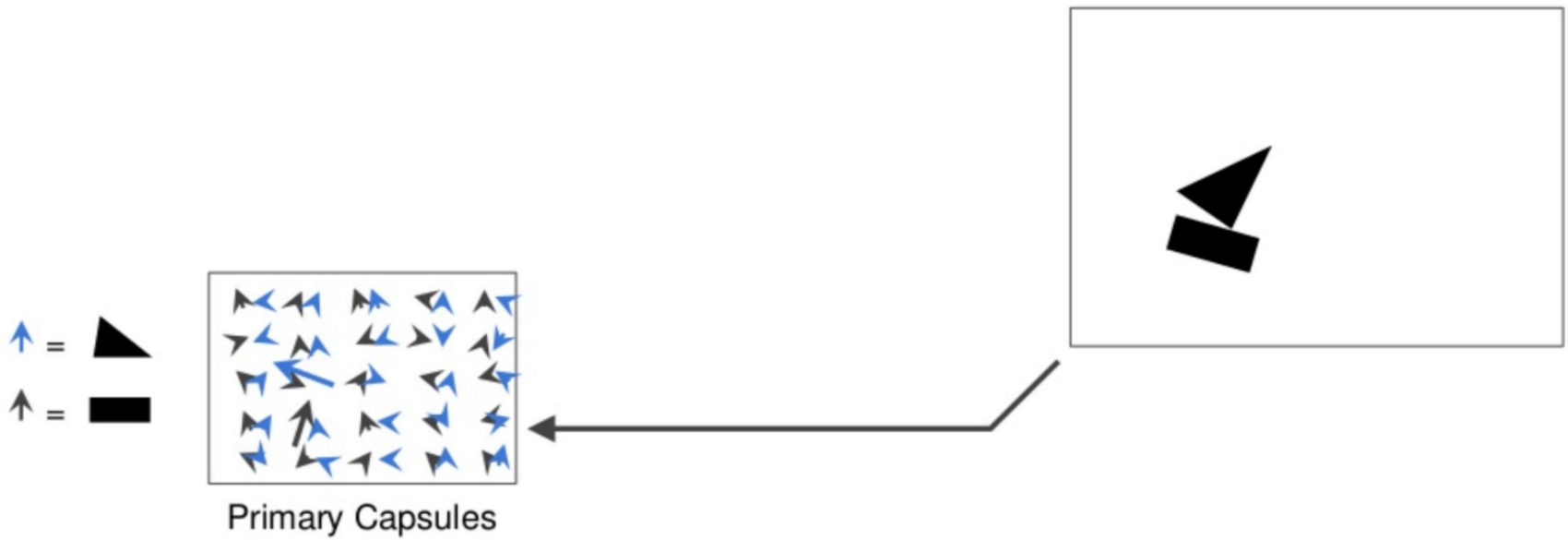
- A capsule is a group of neurons that not only capture the likelihood but also the attributes of a specific feature.
- The output of a capsule can be encoded using a vector and it outputs two things:
    1. the **probability** that the entity is present within its limited domain (expressed as the length of the vector)
    2. a set of **"instantiation parameters"** or in other words the generalized **pose** of the object. This set may include the precise position, lighting or deformation of the visual entity relative to an implicitly defined canonical version of that entity

A Toy Example



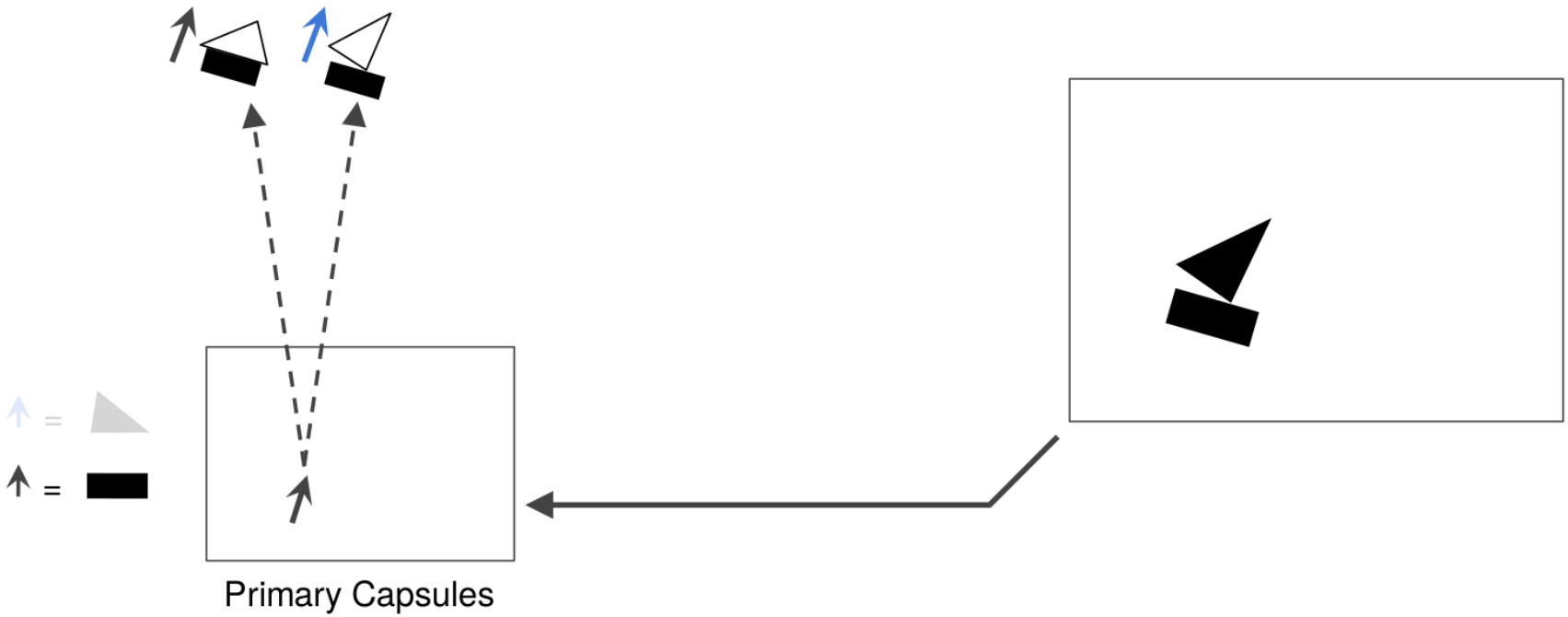
Slides heavily inspired by Aurélien Géron presentation

Primary Capsules



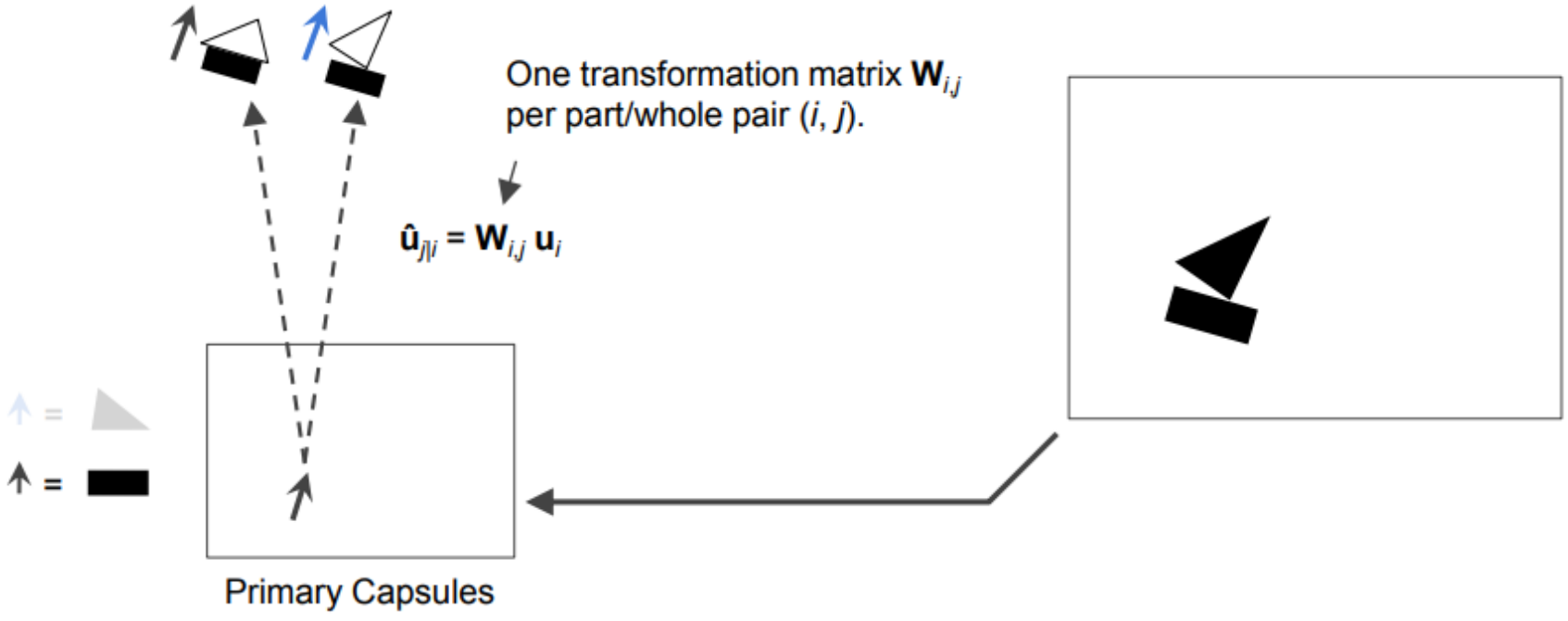
Slides heavily inspired by Aurélien Géron presentation

Predict Next Layer's Output



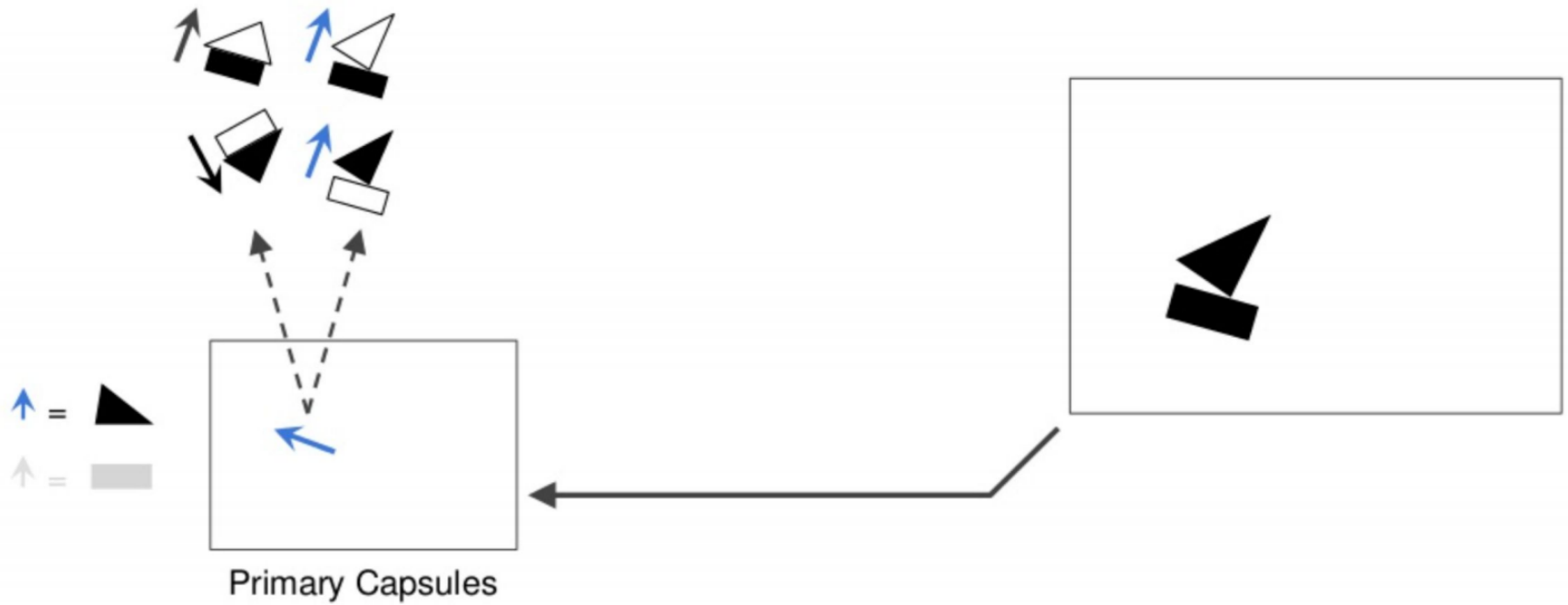
Slides heavily inspired by Aurélien Géron presentation

Predict Next Layer's Output



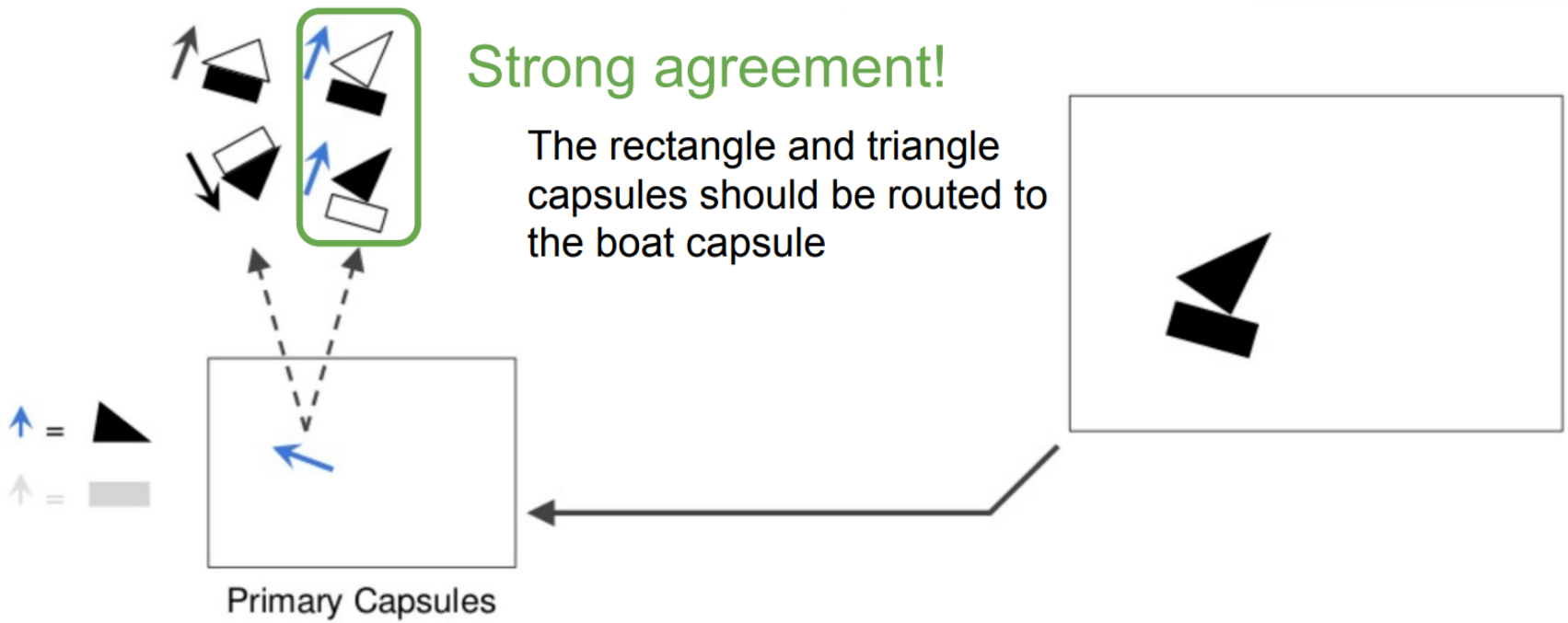
Slides heavily inspired by Aurélien Géron presentation

Predict Next Layer's Output



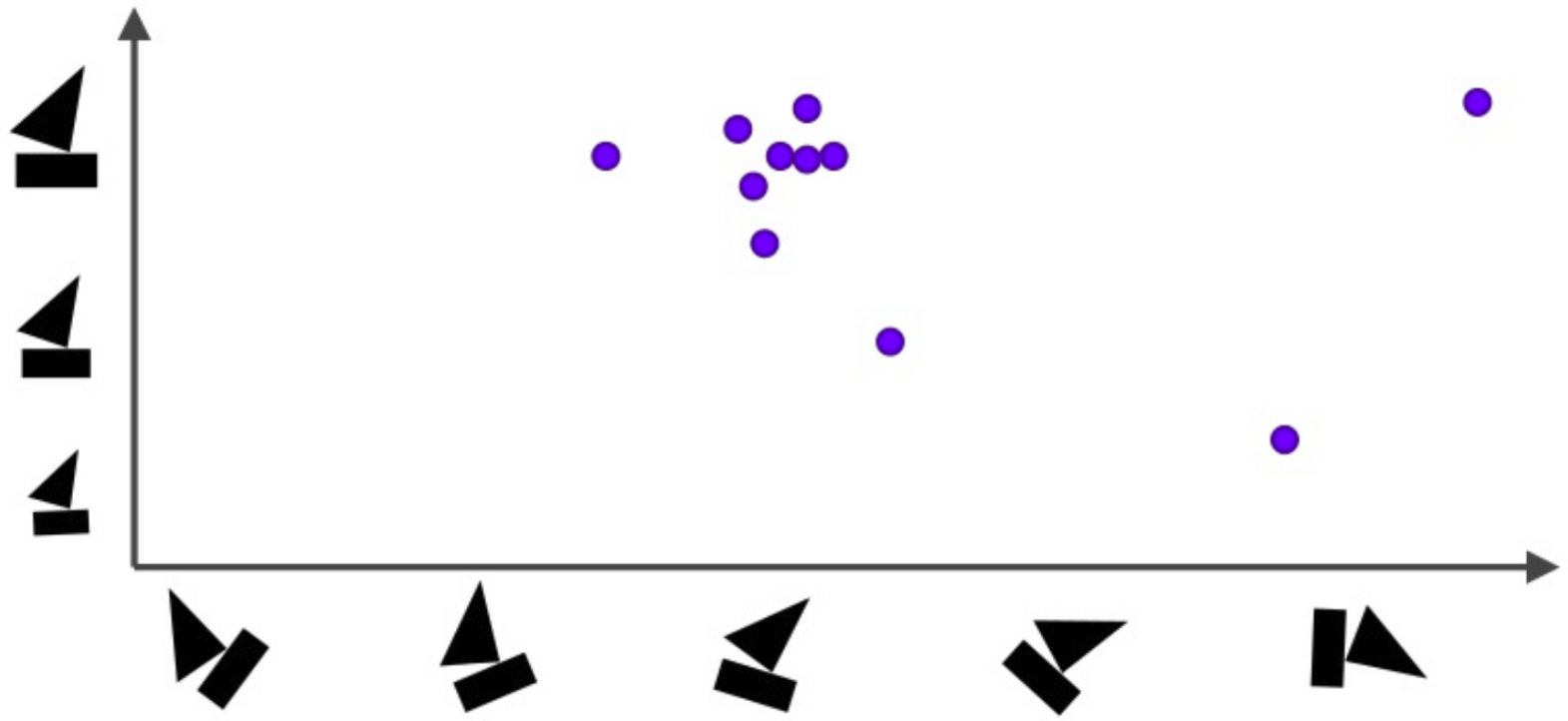
Slides heavily inspired by Aurélien Géron presentation

Routing by Agreement



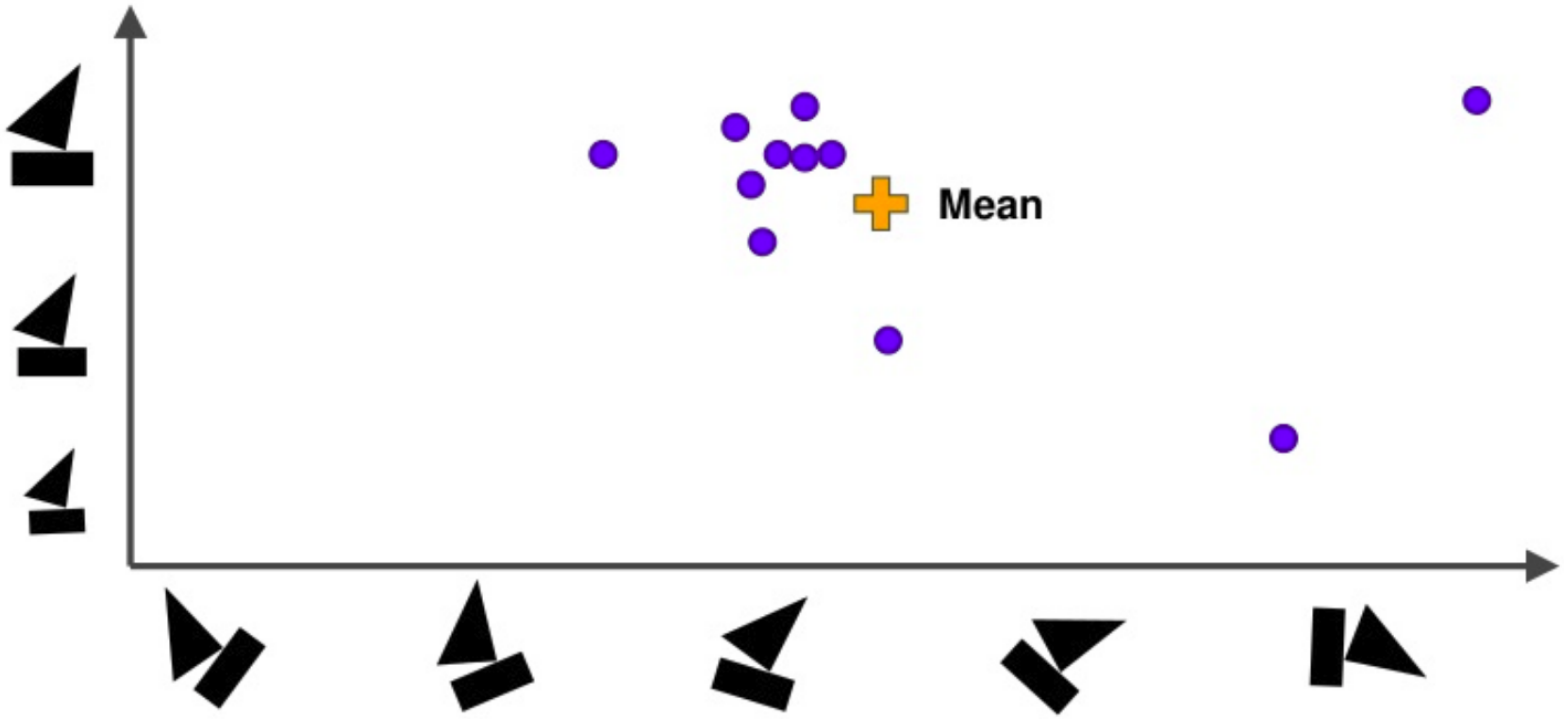
Slides heavily inspired by Aurélien Géron presentation

Clusters of Agreement



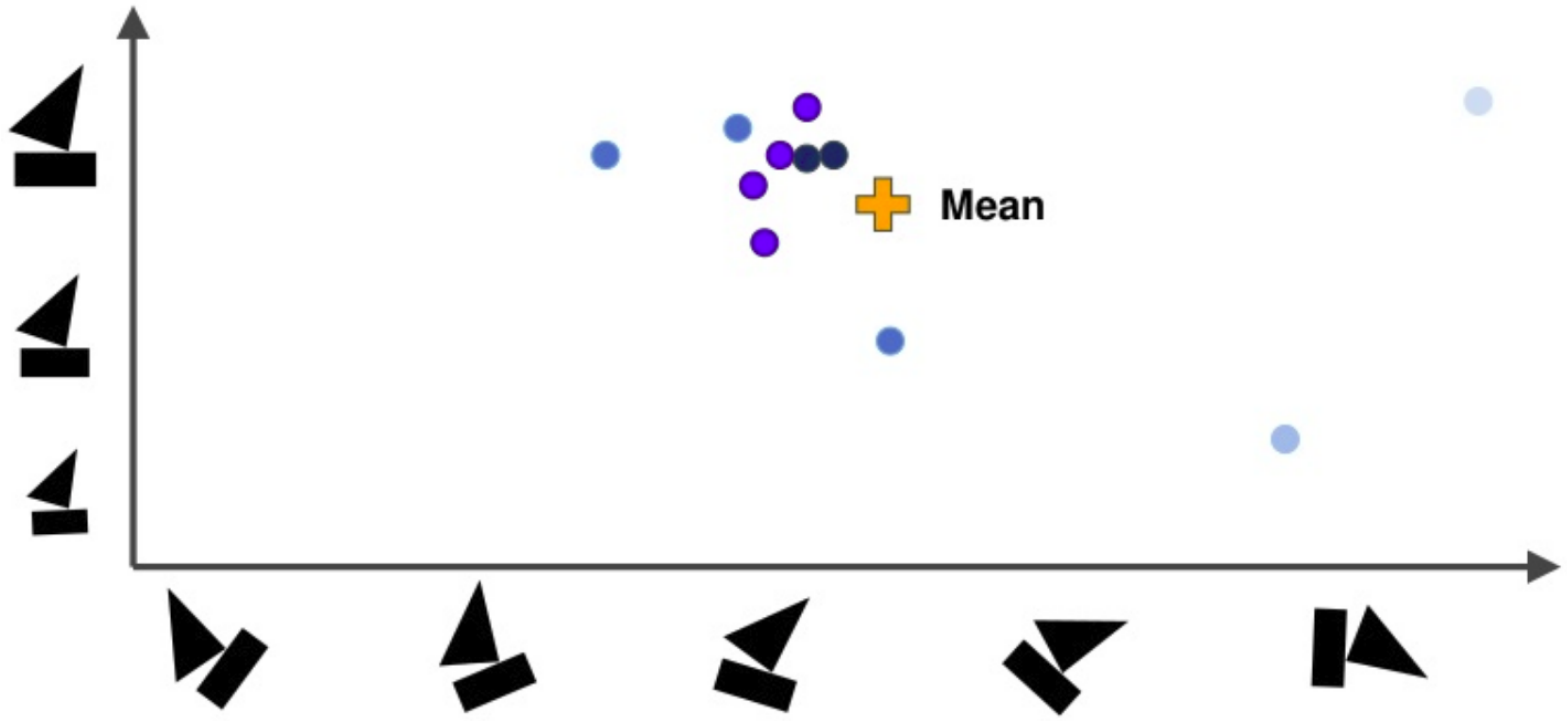
Slides heavily inspired by Aurélien Géron presentation

Clusters of Agreement



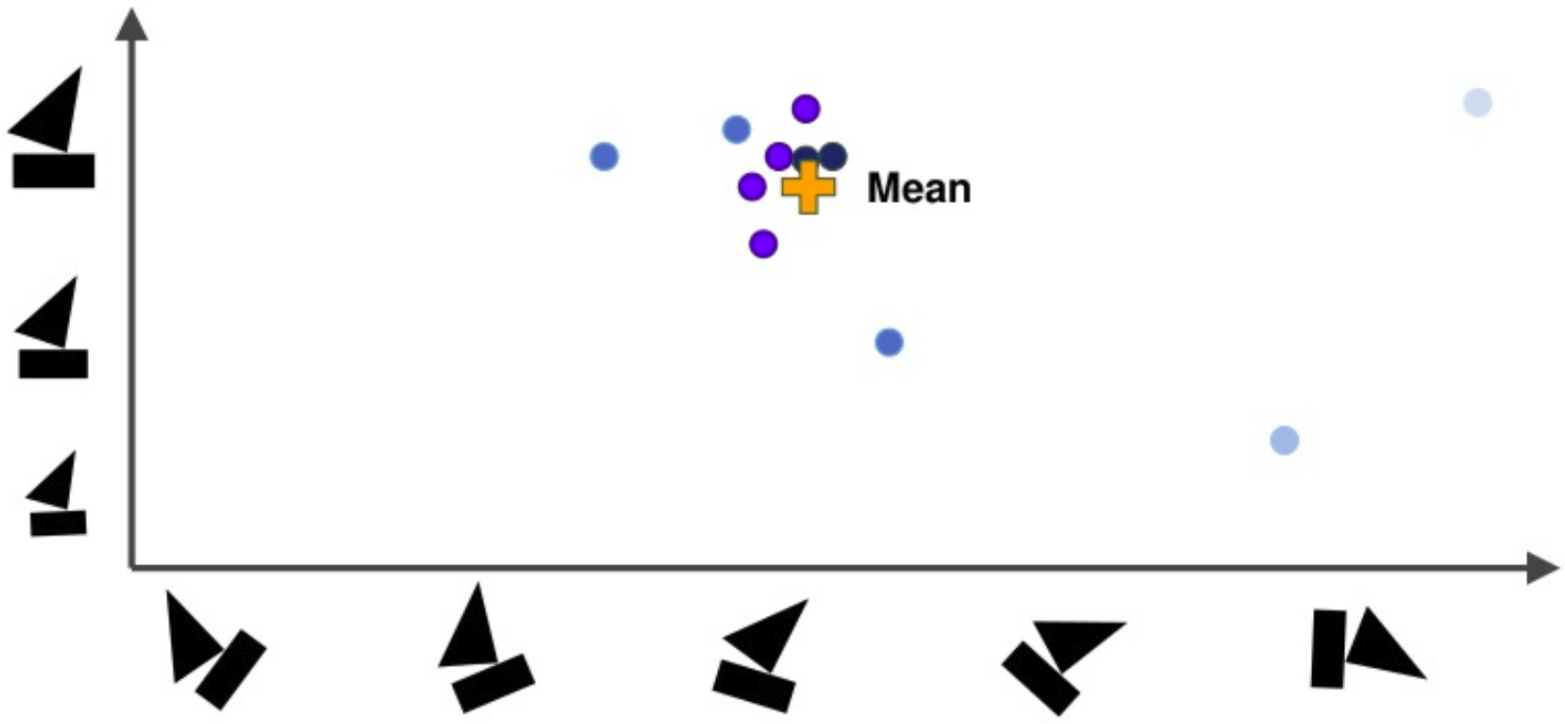
Slides heavily inspired by Aurélien Géron presentation

Clusters of Agreement



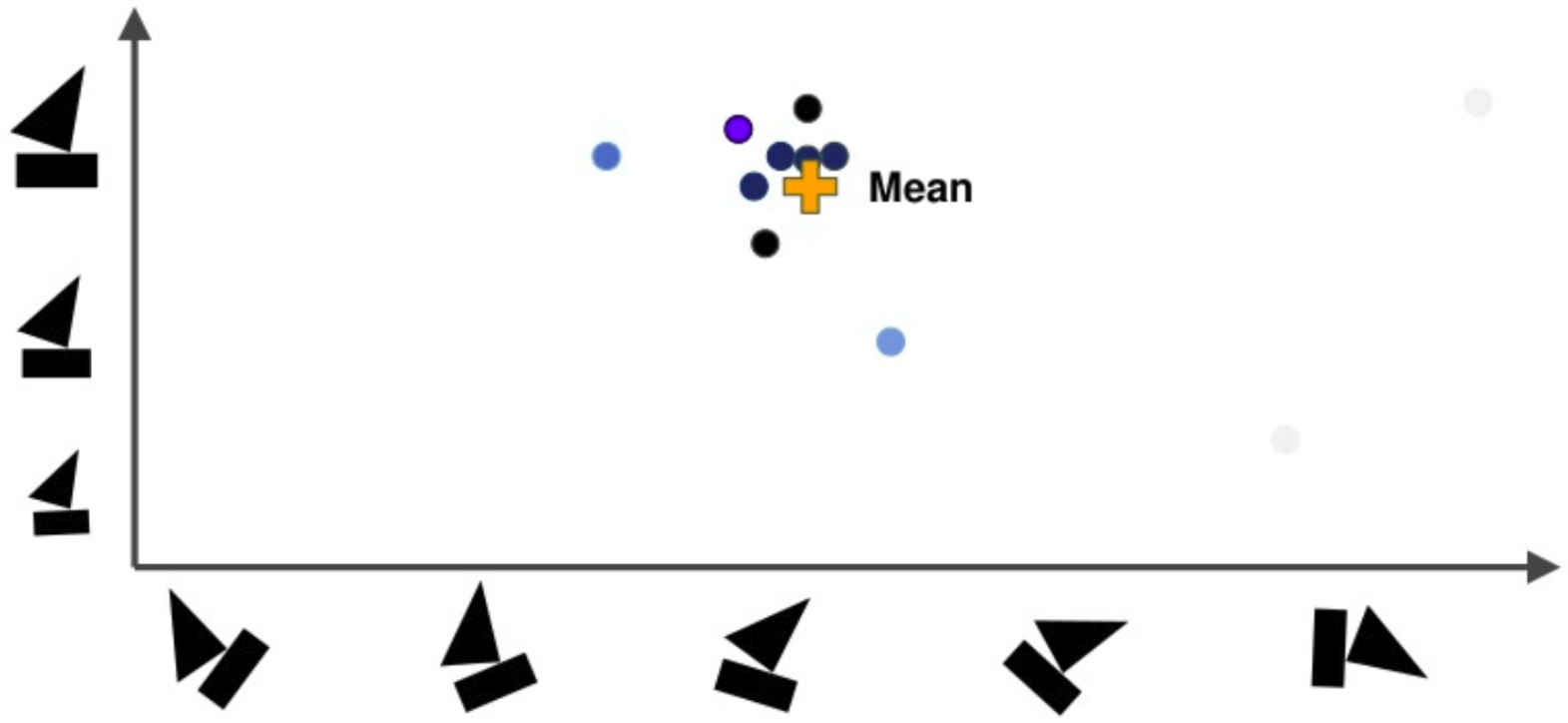
Slides heavily inspired by Aurélien Géron presentation

Clusters of Agreement



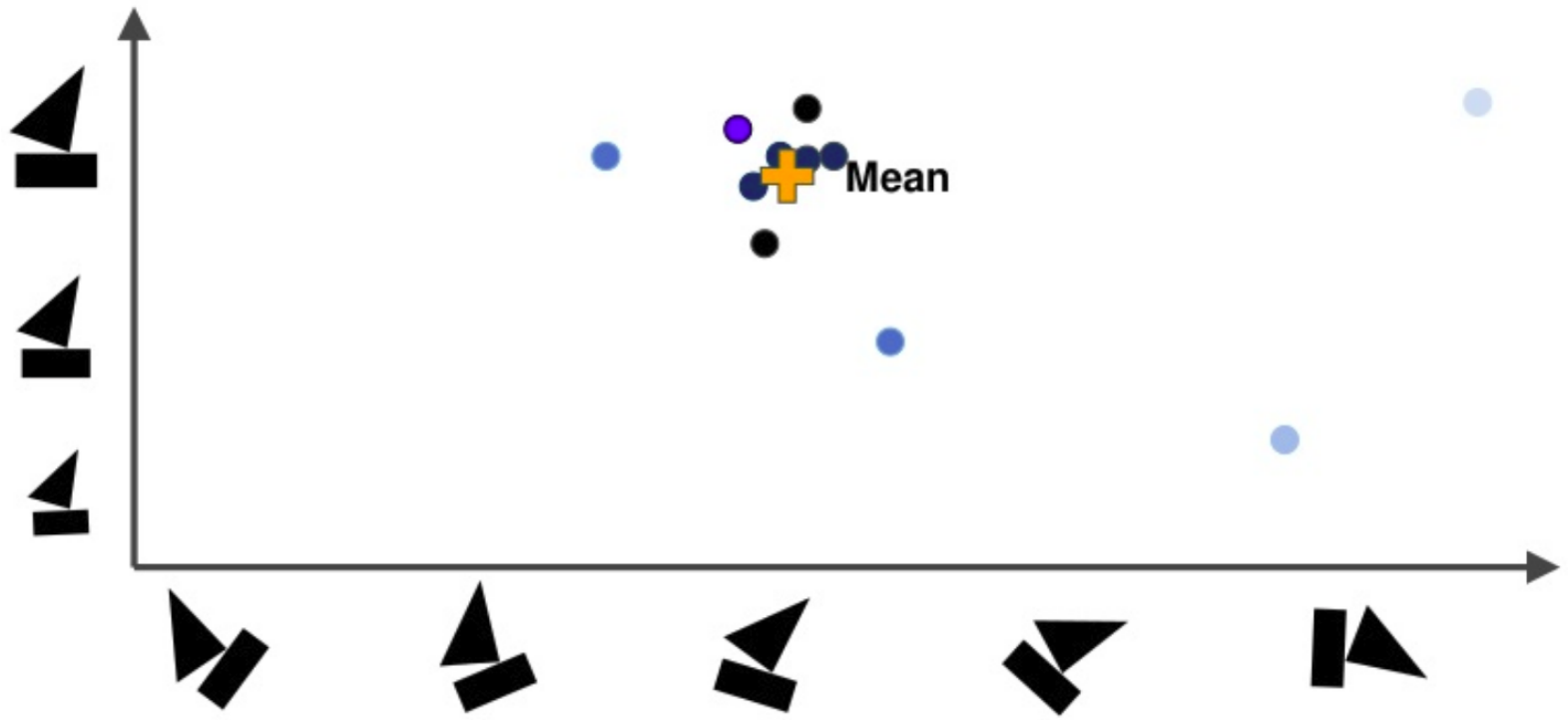
Slides heavily inspired by Aurélien Géron presentation

Clusters of Agreement



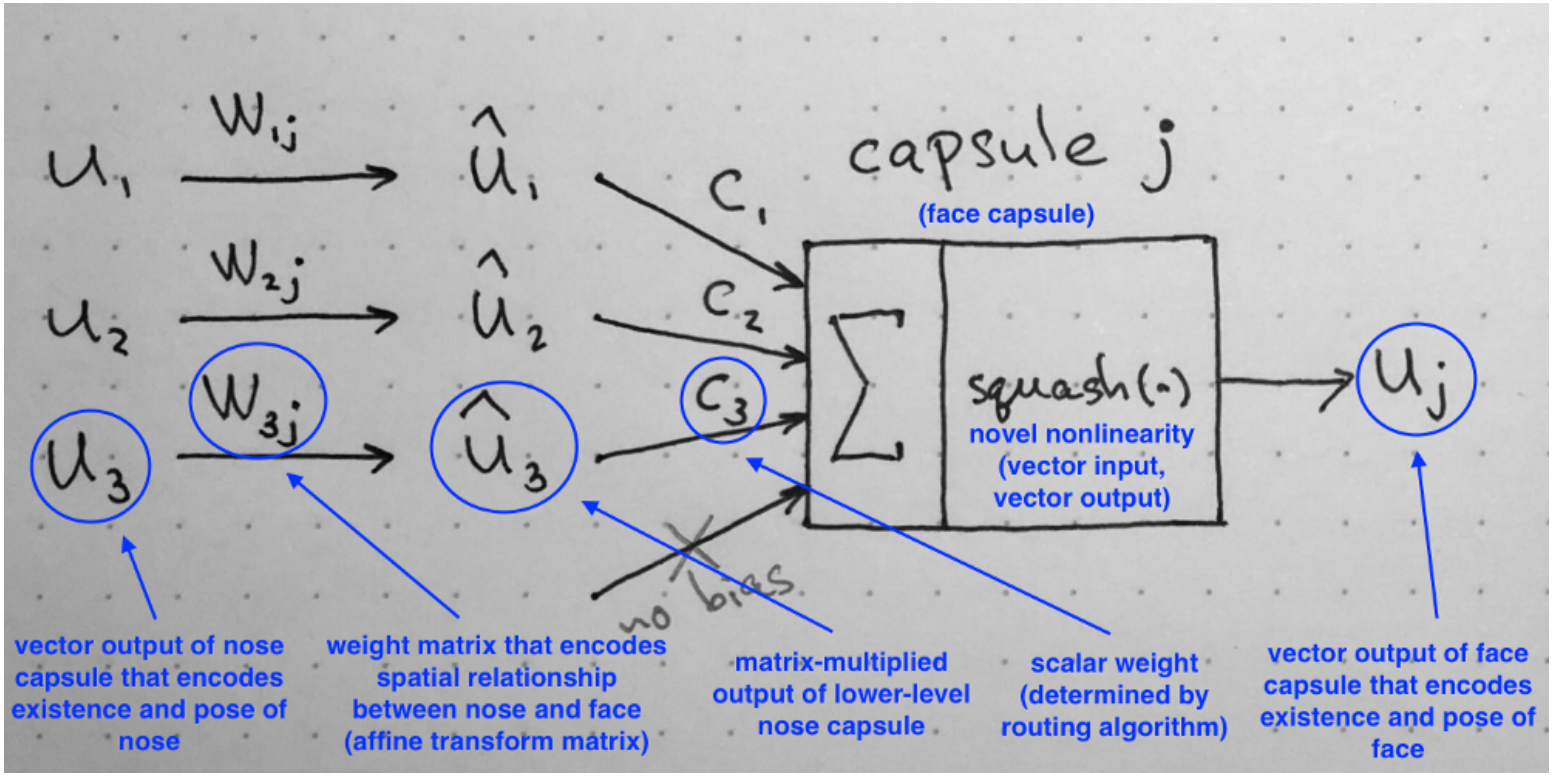
Slides heavily inspired by Aurélien Géron presentation

Clusters of Agreement



Slides heavily inspired by Aurélien Géron presentation

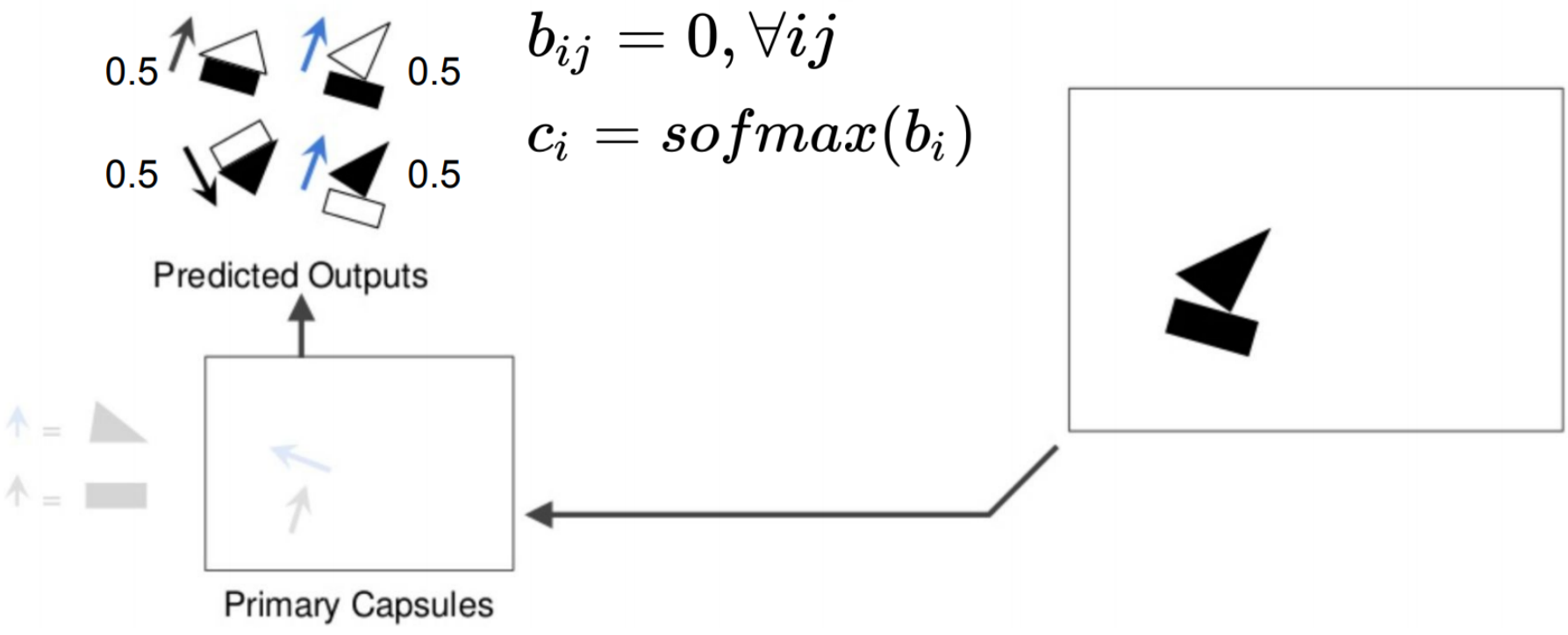
How does a capsule works?



- **W** encodes important spatial and other relationships between lower level features and higher level feature
- **Squash Function**: "Squash" vector to have length of no more than 1, without changing the direction

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

Routing Weights

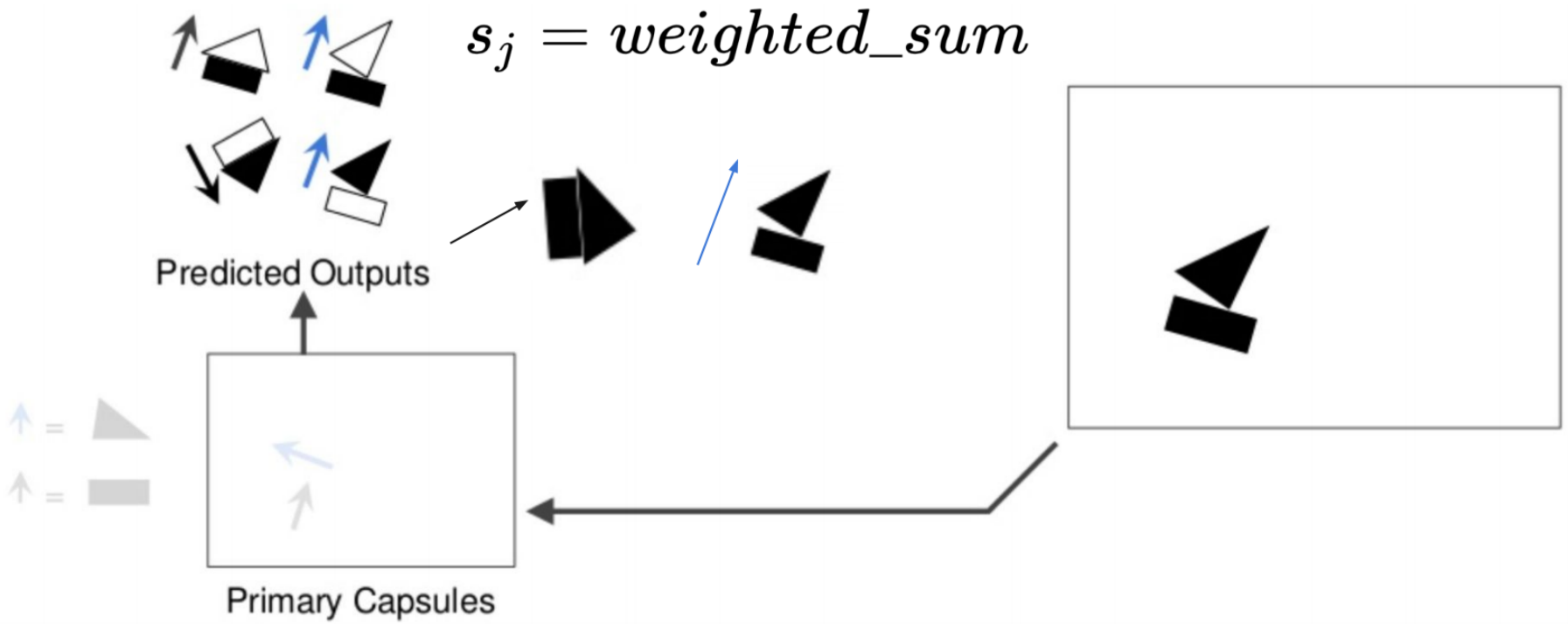


$$b_{ij} = 0, \forall ij$$

$$c_i = softmax(b_i)$$

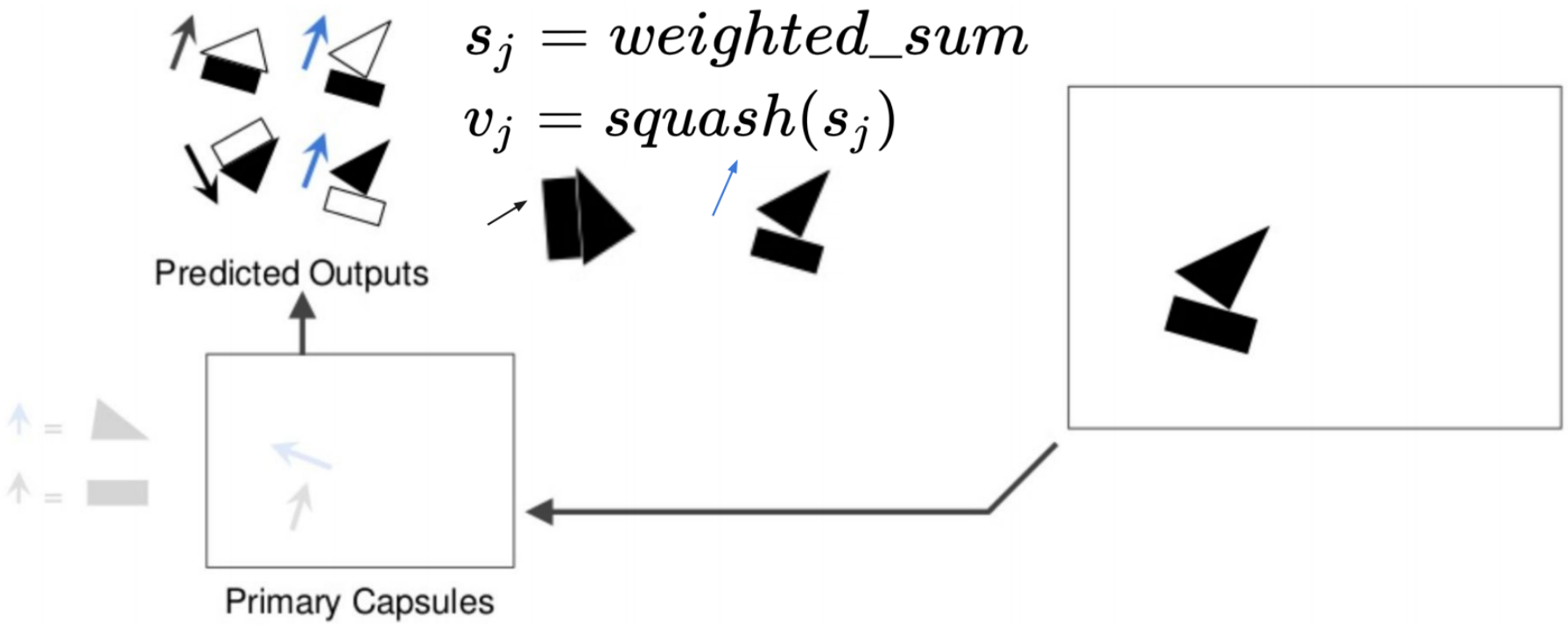
Slides heavily inspired by Aurélien Géron presentation

Compute Next Layer's Output



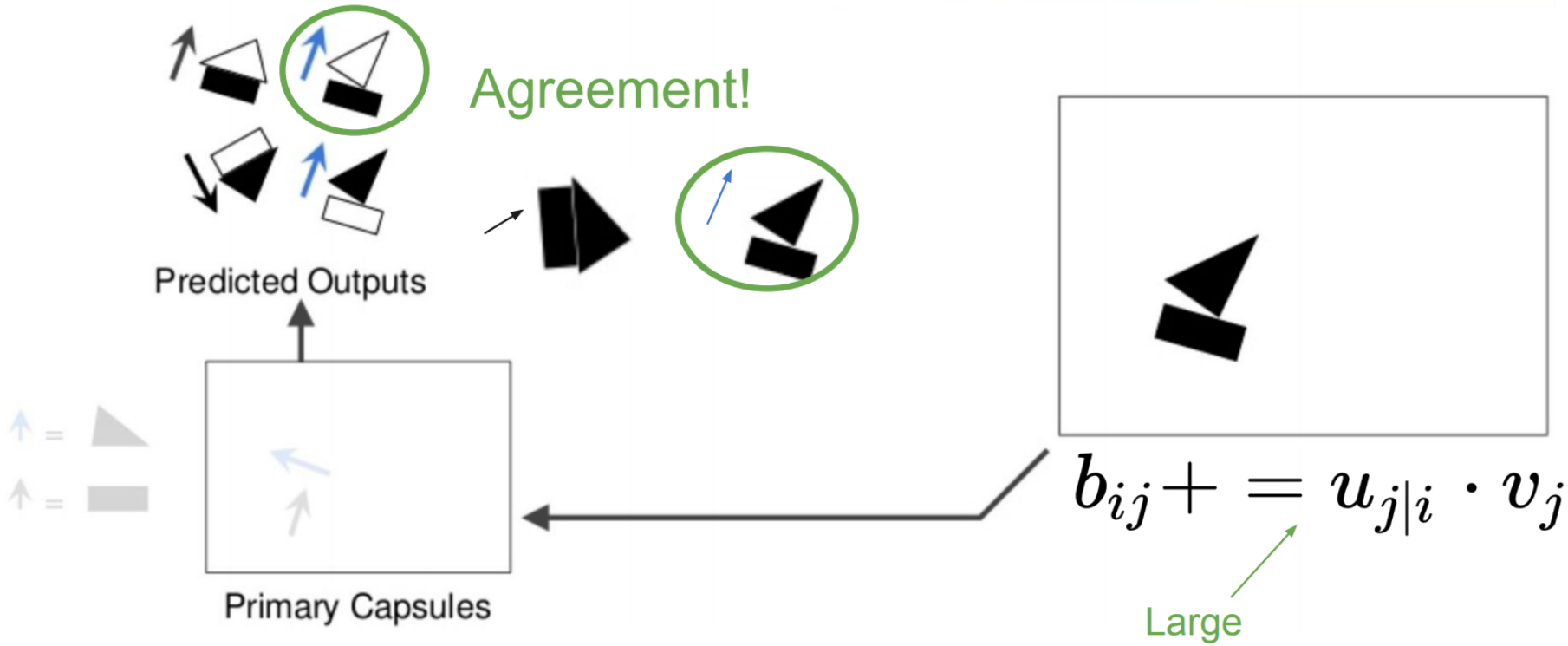
Slides heavily inspired by Aurélien Géron presentation

Compute Next Layer's Output



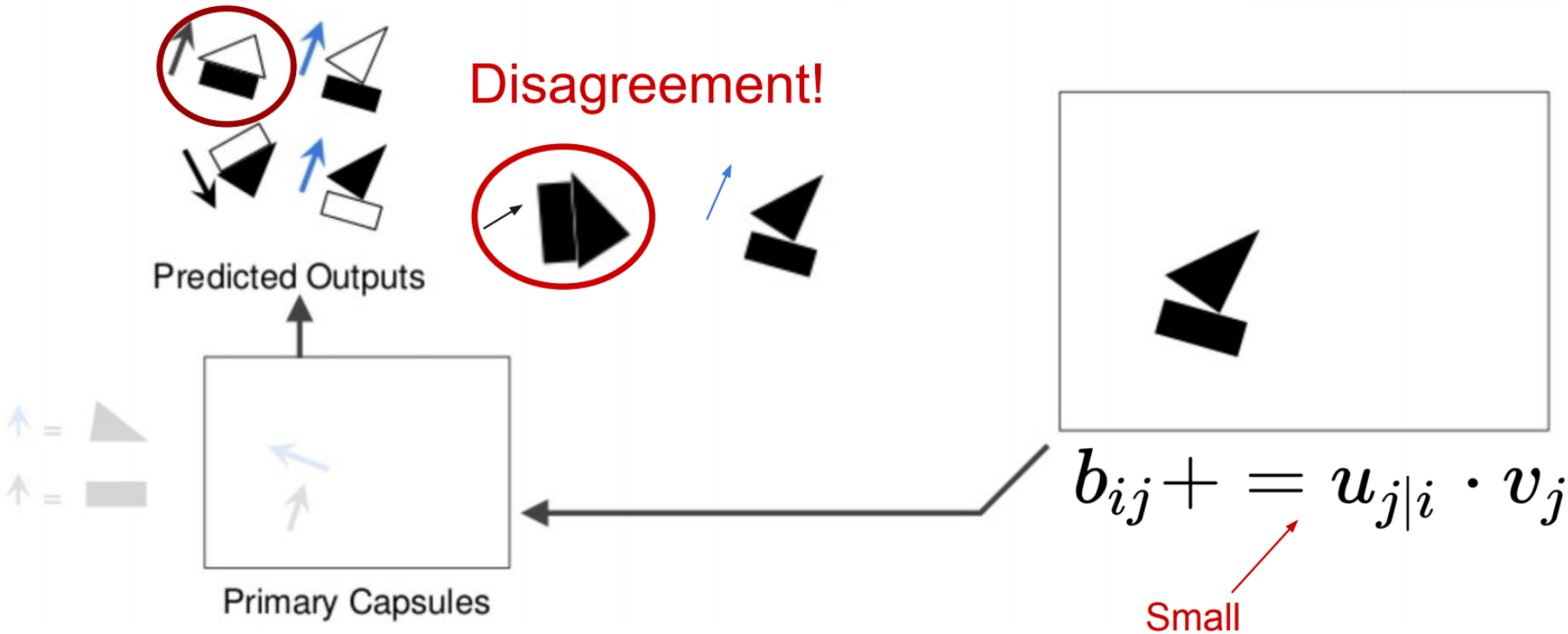
Slides heavily inspired by Aurélien Géron presentation

Update Routing Weights



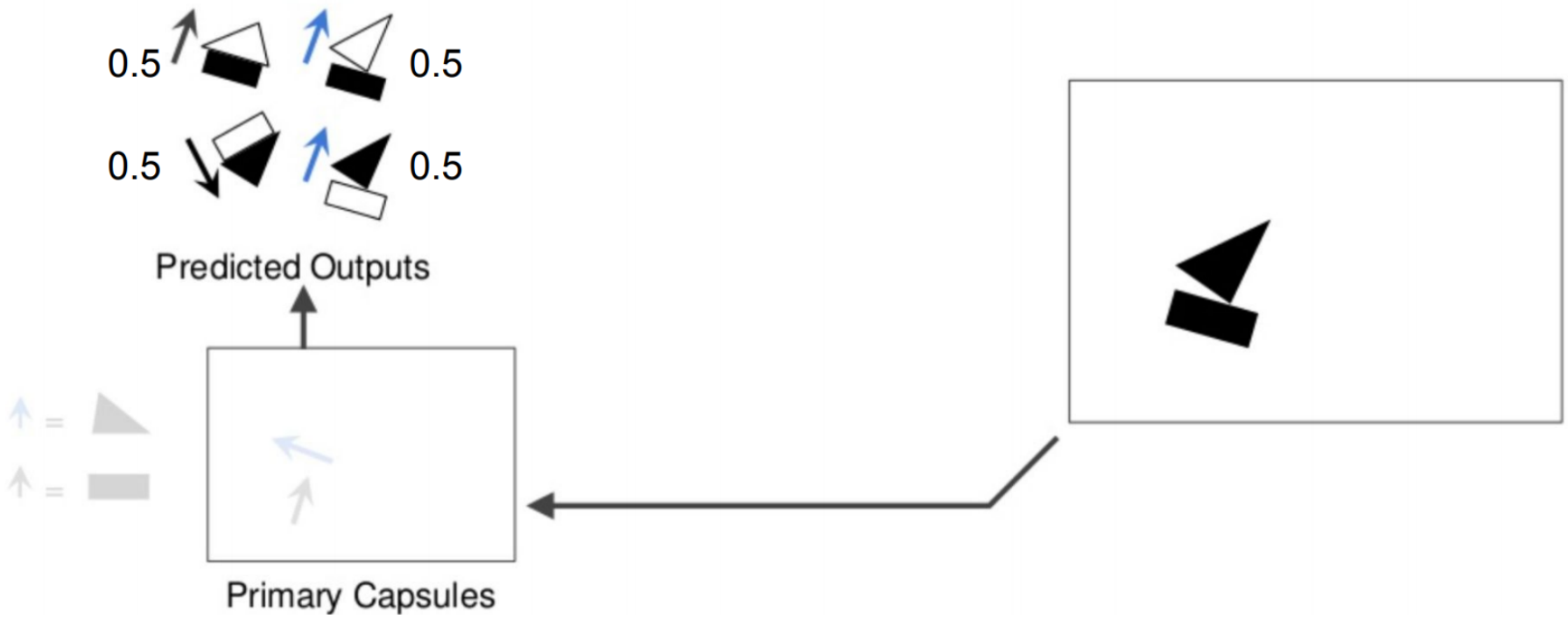
Slides heavily inspired by Aurélien Géron presentation

Update Routing Weights



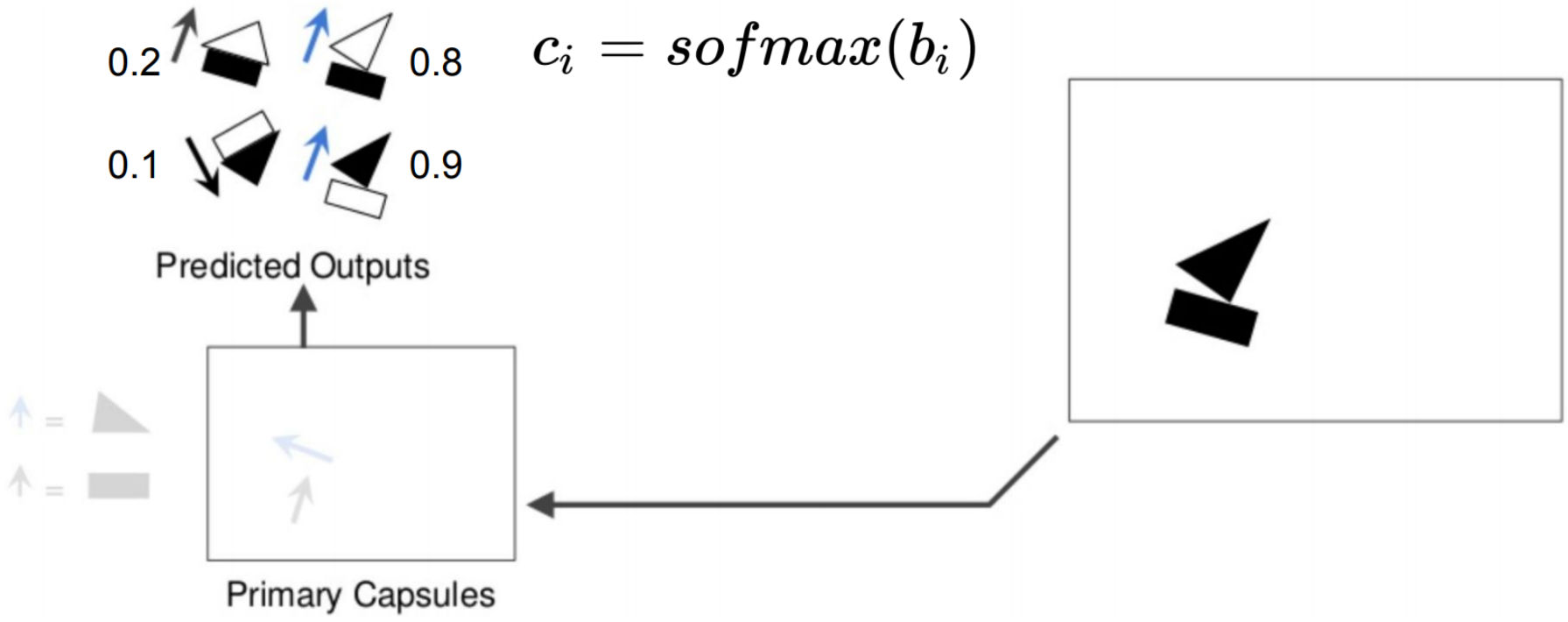
Slides heavily inspired by Aurélien Géron presentation

Routing Weights



Slides heavily inspired by Aurélien Géron presentation

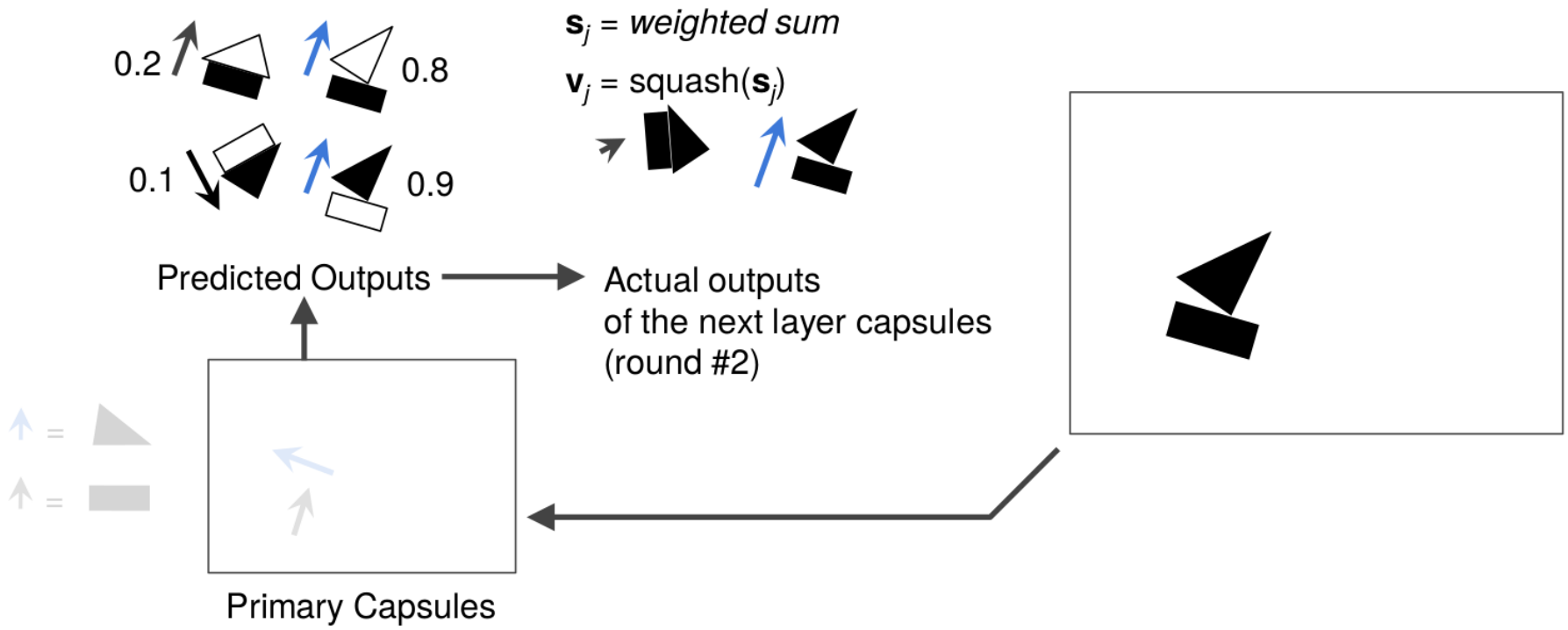
Routing Weights



$$c_i = softmax(b_i)$$

Slides heavily inspired by Aurélien Géron presentation

Compute Next Layer's Output



Slides heavily inspired by Aurélien Géron presentation

Dynamic Routing Between Capsules

Lower level capsule will send its input to the higher level capsule that "agrees" with its input. This is the essence of the dynamic routing algorithm.

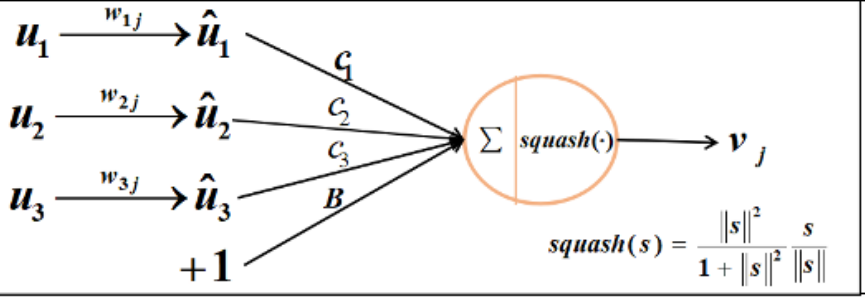
Procedure 1 Routing algorithm.

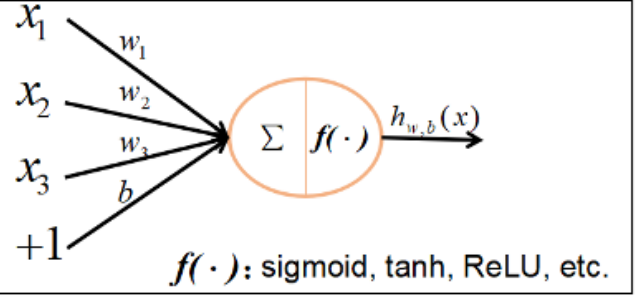
1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
5: for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

- Similar to k-means algorithm, the dynamic routing tries to find clusters of agreement between input capsules relative to each output capsule using the dot product similarity measure and updating the routing coefficients correspondingly
- More iterations tends to overfit the data
- It is recommended to use 3 routing iterations in practice

Capsule vs Traditional Neuron

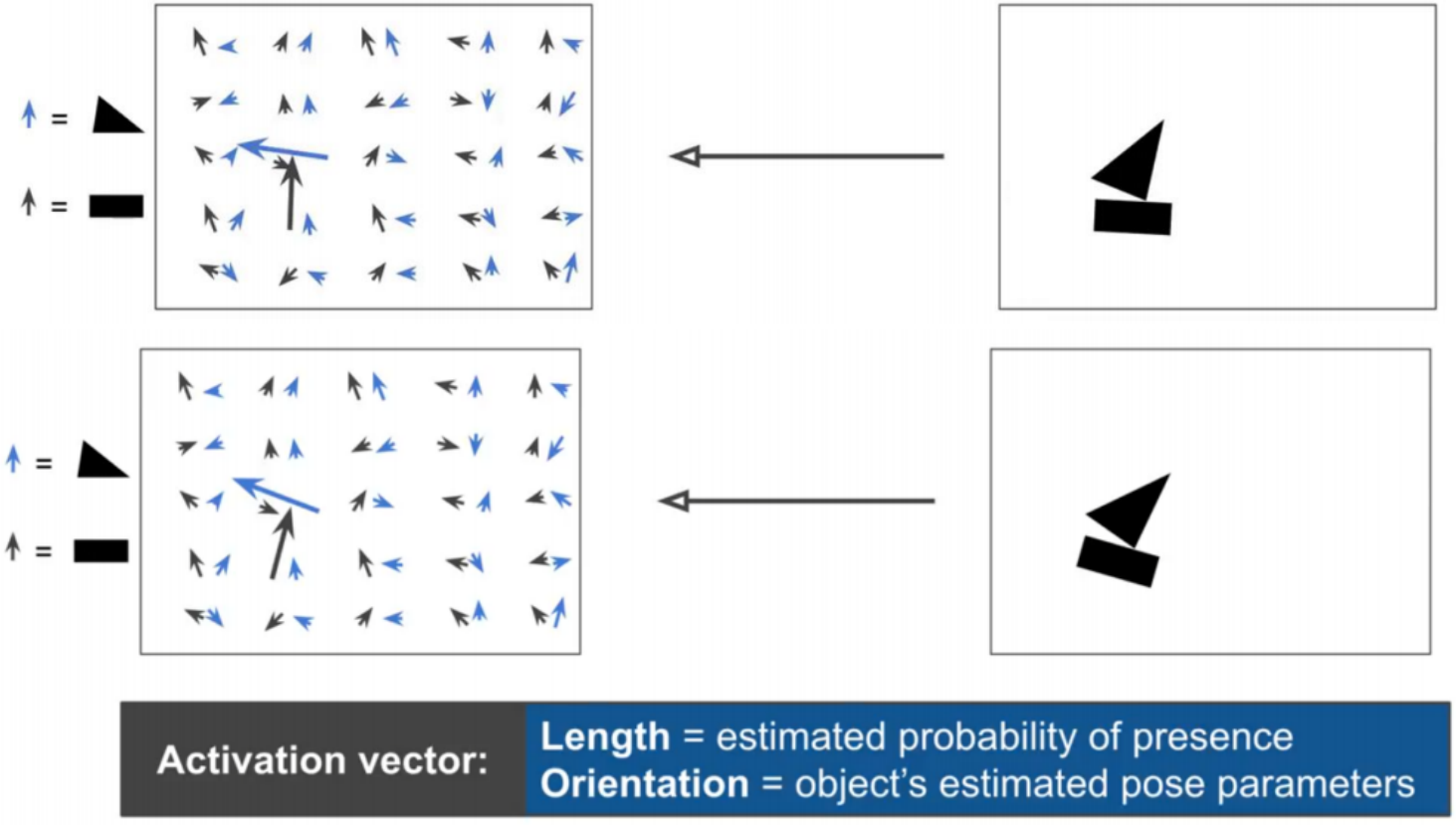
		capsule	VS.	traditional neuron
Input from low-level neurons/capsules		$\text{vector}(u_i)$		$\text{scalar}(x_i)$
Operations	Linear/Affine Transformation	$\hat{u}_{j\|i} = W_{ij} u_i + B_j$ (Eq. 2)		$a_{j\|i} = w_{ij} x_i + b_j$
	Weighting	$s_j = \sum_i c_{ij} \hat{u}_{j\|i}$ (Eq. 2)		$z_j = \sum_{i=1}^{3}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





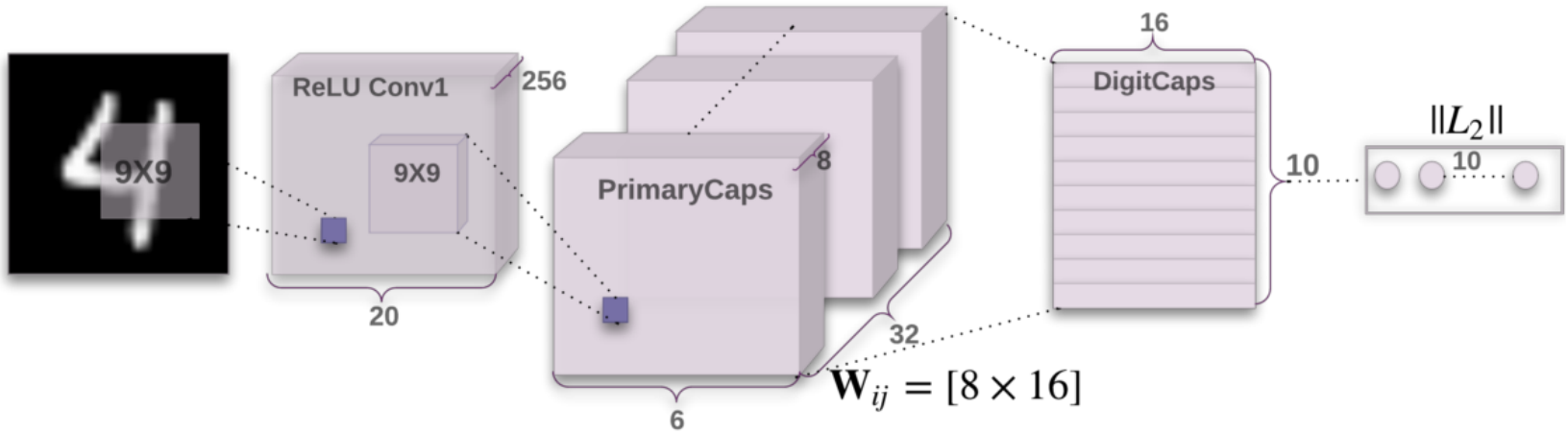
Capsule = New Version Neuron!
vector in, vector out VS. scalar in, scalar out

Capsule Equivariance



- If the detected feature moves around the image or its state somehow changes, **the probability still stays the same**
- This is what Hinton refers to as **activities equivariance**: neuronal activities will change when an object "moves over the manifold of possible appearances" in the picture. At the same time, the probabilities of detection remain constant, which is the form of invariance that we should aim at, and not the type offered by CNNs with max pooling.

CapsNet Architecture



CapsNet Architecture

1. **Layer 1 - Convolutional layer**: its job is to detect basic features in the 2D image. In the CapsNet, the convolutional layer has 256 kernels with size of $[9 \times 9 \times 1]$ and stride 1, followed by ReLU activation. The output of this network is $[20 \times 20 \times 256]$ features maps in MNIST.

2. **Layer 2 - PrimaryCaps layer**: this layer has 32 primary capsules whose job is to take basic features detected by the convolutional layer and produce combinations of the features. The layer has 32 "primary capsules" that are very similar to convolutional layer in their nature (with squash function at the end for non-linearity). Each capsule applies eight $[9 \times 9 \times 256]$ convolutional kernels (with stride 2) to the $[20 \times 20 \times 256]$ input volume and therefore produces $[6 \times 6 \times 8]$ output tensor. Since there are 32 such capsules, the output volume has shape of $[32 \times 6 \times 6 \times 8]$ or reshaped $[1152 \times 8]$.

3. **Layer 3 - DigitCaps layer**: this layer has 10 digit capsules, one for each digit. Each capsule takes as input a $[6 \times 6 \times 8 \times 32]$ tensor. You can think of it as $[6 \times 6 \times 32]$ 8-dimensional vectors, which is 1152 input vectors in total. As per the inner workings of the capsule, each of these input vectors gets their own $[8 \times 16]$ transformation matrix W_{ij} that maps 8-dimensional input space to the 16-dimensional capsule output space. So, there are 1152 matrices for each capsule, and also 1152 c coefficients and 1152 b coefficients used in the dynamic routing.

Margin Loss Function + Reconstruction as regularizer

CapsNet Loss Function

calculated for correct DigitCap

calculated for incorrect DigitCaps

loss term for one DigitCap

L2 norm

L2 norm

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

1 when correct DigitCap, 0 when incorrect

zero loss when correct prediction with probability greater than 0.9, non-zero otherwise

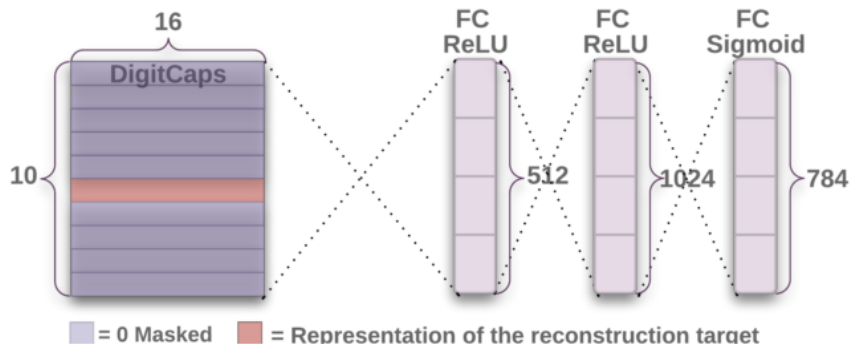
0.5 constant used for numerical stability

1 when incorrect DigitCap, 0 when correct

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

Note: correct DigitCap is one that matches training label, for each training example there will be 1 correct and 9 incorrect DigitCaps

The authors, along with the CapsNet loss they introduced reconstruction loss as a regularization method. The loss is defined as the MSE with the original image.



The total loss is defined as:

$$L_{total} = \sum_{c=1}^{C} L_c + 0.0005 * L_{reg}$$

Results

The authors argue that the capsules are successfully learn to span the space of variations, in the way digits of that class are instantiated. For example:

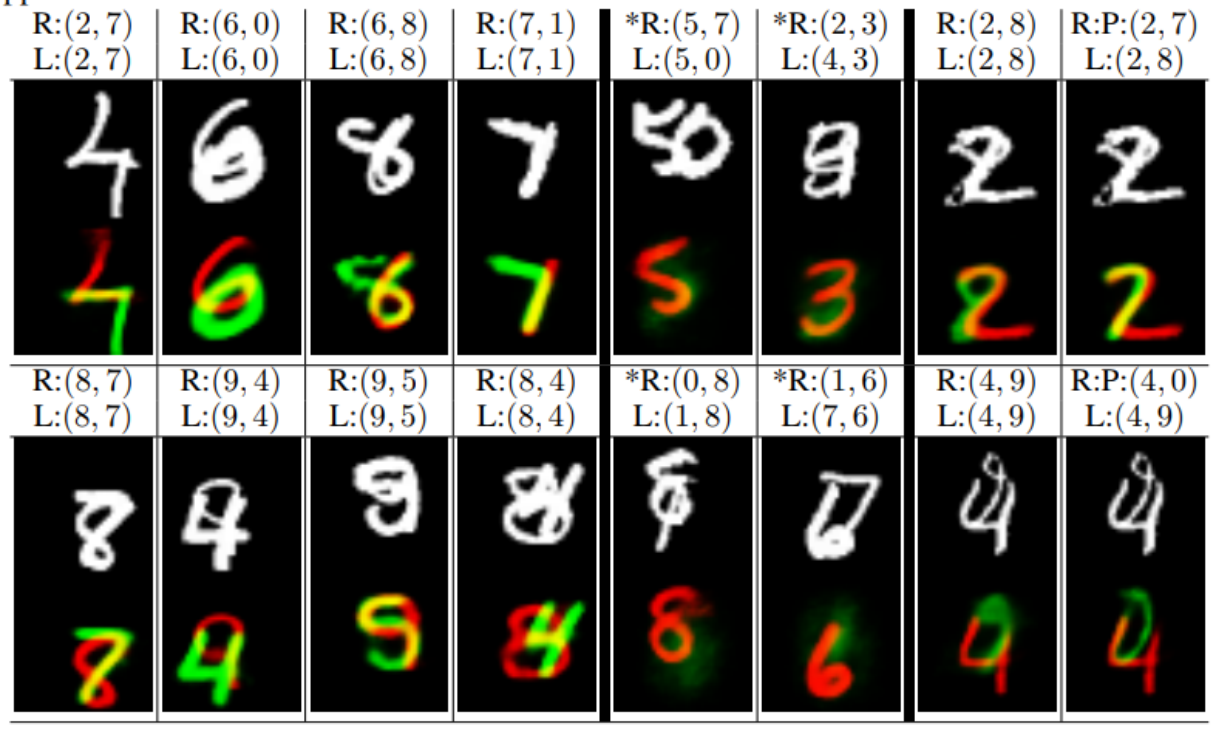
Figure 4: Dimension perturbations. Each row shows the reconstruction when one of the 16 dimensions in the DigitCaps representation is tweaked by intervals of 0.05 in the range $[-0.25, 0.25]$.

Scale and thickness	
Localized part	
Stroke thickness	
Localized skew	
Width and translation	
Localized part	

- MNIST: 0.25% test error
- CIFAR10: 10.6% test error (the authors state that is about what standard convolutional nets achieved when they were first applied to the dataset)

Segmenting highly overlapping digits

Figure 5: Sample reconstructions of a CapsNet with 3 routing iterations on MultiMNIST test dataset. The two reconstructed digits are overlayed in green and red as the lower image. The upper image shows the input image. L:(l_1, l_2) represents the label for the two digits in the image and R:(r_1, r_2) represents the two digits used for reconstruction. The two right most columns show two examples with wrong classification reconstructed from the label and from the prediction (P). In the $(2, 8)$ example the model confuses 8 with a 7 and in $(4, 9)$ it confuses 9 with 0. The other columns have correct classifications and show that the model accounts for all the pixels while being able to assign one pixel to two digits in extremely difficult scenarios (column $1 - 4$). Note that in dataset generation the pixel values are clipped at 1. The two columns with the (*) mark show reconstructions from a digit that is neither the label nor the prediction. These columns suggests that the model is not just finding the best fit for all the digits in the image including the ones that do not exist. Therefore in case of $(5, 0)$ it cannot reconstruct a 7 because it knows that there is a 5 and 0 that fit best and account for all the pixels. Also, in case of $(8, 1)$ the loop of 8 has not triggered 0 because it is already accounted for by 8. Therefore it will not assign one pixel to two digits if one of them does not have any other support.



Matrix Capsules With EM Routing

What's different

Recap:

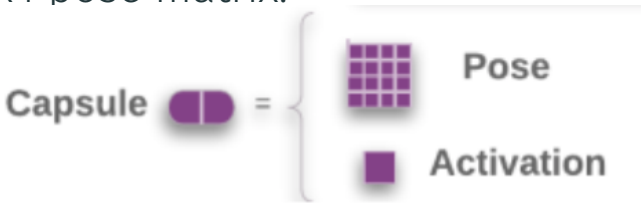
> *A* ***capsule*** *is* ***a group of neurons*** *whose* ***output represents*** *different properties of the* ***same entity****.*

General ideas differ from the original paper:

- Activity Vector $\rightarrow$ Pose Matrix + Activity Probability
- Dynamic Routing $\rightarrow$ EM Routing

Matrix Capsule

- A matrix capsule captures the activation (likeliness) similar to that of a neuron, but also captures a 4x4 pose matrix.



- In computer graphics, a pose matrix defines the translation and the rotation of an object which is equivalent to the change of the viewpoint of an object.
- An example:

- Of course, just like other deep learning methods, this is the intention and it is never guaranteed.

EM Routing By Agreement

- The objective of the EM (Expectation Maximization) routing is to group capsules to form a part-whole relationship using a clustering technique (EM).
- In machine learning, we use EM clustering to cluster datapoints into Gaussian distributions.
- A higher level feature is detected by looking for agreement between votes from the capsules one layer below. A **vote** v_{ij} for the parent capsule j from capsule i is computed by multipling the pose matrix M_i of capsule i with a **viewpoint invariant transformation** W_{ij}.

$$v_{ij} = M_i W_{ij}$$

- The probability that a capsule i is grouped into capsule j as a part-whole relationship is based on the proximity of the vote v_{ij} to other votes $(v_{o_1 j} \ldots v_{o_k j})$ from other capsules. W_{ij} is learned discriminatively through a cost function and the backpropagation.

Calculate capsule activation and pose matrix

In EM routing, each capsule in the higher-layer corresponds to a Gaussian and the pose of each active capsule in the lower-layer (converted to a vector) corresponds to a data-point (or a fraction of a data-point if the capsule is partially active). The pose matrix is a 4×4 matrix, i.e. 16 components. We model the pose matrix with a Gaussian having 16 μ and 16 σ and each μ represents a pose matrix's component.

Let v_{ij} be the vote from capsule i for the parent capsule j, and v_{ij}^h be its h-th component. We apply the probability density function of a Gaussian

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

to compute the probability of v_{ij}^h belonging to the capsule j's Gaussian model:

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

If we take the natural *log*:

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

Calculate capsule activation and pose matrix

Let's estimate the cost to activate a capsule. The lower the cost, the more likely a capsule will be activated. If cost is high, the votes do not match the parent Gaussian distribution and therefore a low chance to be activated.

Let $cost_{ij}$ be the cost to activate the parent capsule j by the capsule i. It is the negative of the log likelihood:

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

Since capsules are not equally linked with capsule j, we pro-rated the cost with the runtime assignment probabilities r_{ij}. The cost from all lower layer capsules is:

$$\begin{aligned}
cost_{j}^{h} &= \sum_{i} r_{ij} cost_{ij}^{h} \\
&= \sum_{i} -r_{ij} \ln(p_{i|j}^{h}) \\
&= \sum_{i} r_{ij} \left(\frac{(v_{ij}^{h} - \mu_{j}^{h})^2}{2(\sigma_{j}^{h})^2} + \ln(\sigma_{j}^{h}) + \frac{\ln(2\pi)}{2} \right) \\
&= \frac{\sum_{i} r_{ij} (\sigma_{j}^{h})^2}{2(\sigma_{j}^{h})^2} + \left(\ln(\sigma_{j}^{h}) + \frac{\ln(2\pi)}{2} \right) \sum_{i} r_{ij} \\
&= \left(\ln(\sigma_{j}^{h}) + \frac{1}{2} + \frac{\ln(2\pi)}{2} \right) \sum_{i} r_{ij} \\
&= \left(\ln(\sigma_{j}^{h}) + k \right) \sum_{i} r_{ij} \quad \text{which k is a constant}
\end{aligned}$$

Calculate capsule activation and pose matrix

Using the minimum description length (MDL) principle we have a choice when deciding whether or not to activate a higher-level capsule.

- **Choice 0**: if we do not activate it, we must pay a fixed cost of $-\beta_u$ per data-point for describing the poses of all the lower-level capsules that are assigned to the higher-level capsule. This cost is the negative log probability density of the data-point under an improper uniform prior. For fractional assignments we pay that fraction of the fixed cost.
- **Choice 1**: if we do activate the higher-level capsule we must pay a fixed cost of $-\beta_\alpha$ for coding its mean and variance and the fact that it is active and then pay additional costs, pro-rated by the assignment probabilities, for describing the discrepancies between the lower-level means and the values predicted for them when the mean of the higher-level capsule is used to predict them via the inverse of the transformation matrix.

A much simpler way to compute the cost of describing a datapoint is to use the negative log probability density of that datapoint's vote under the Gaussian distribution fitted by whatever higher-level capsule it gets assigned to. This is incorrect for reasons explained in the paper, but we use it because it requires much less computation.

Thus, to determine whether the capsule j will be activated, we use the following equation:

$$\alpha_j = sigmoid\big(\lambda\big(\beta_\alpha - \beta_u \sum_i r_{ij} - \sum_h cost_j^h\big)\big)$$

where λ is the inverse temperature parameter $\frac{1}{temperature}$.

The β_α and β_u are not computed analytically. Instead, they're approximated through training using the back-propagation and the cost function.

EM Routing Algorithm

Procedure 1 Routing algorithm returns **activation** and **pose** of the capsules in layer $L+1$ given the **activations** and **votes** of capsules in layer L. V_{ij}^h is the h^{th} dimension of the vote from capsule i with activation a_i in layer L to capsule j in layer $L+1$. β_a, β_u are learned discriminatively and the inverse temperature λ increases at each iteration with a fixed schedule.

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

procedure M-STEP($\boldsymbol{a}$, R, V, j) ▷ for one higher-level capsule, j
  $\forall i \in \Omega_L$: $R_{ij} \leftarrow R_{ij} * \boldsymbol{a}_i$
  $\forall h$: $\mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$
  $\forall h$: $(\sigma_j^h)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$
  $cost^h \leftarrow \left(\beta_u + log(\sigma_j^h)\right) \sum_i R_{ij}$
  $a_j \leftarrow logistic(\lambda(\beta_a - \sum_h cost^h))$

procedure E-STEP(μ, σ, $\boldsymbol{a}$, V, i) ▷ for one lower-level capsule, i
  $\forall j \in \Omega_{L+1}$: $\boldsymbol{p}_j \leftarrow \frac{1}{\sqrt{\prod_h^H 2\pi(\sigma_j^h)^2}} \exp\left(-\sum_h^H \frac{(V_{ij}^h - \boldsymbol{\mu}_j^h)^2}{2(\sigma_j^h)^2}\right)$
  $\forall j \in \Omega_{L+1}$: $\boldsymbol{R}_{ij} \leftarrow \frac{\boldsymbol{a}_j p_j}{\sum_{k \in \Omega_{L+1}} \boldsymbol{a}_k p_k}$

- The EM method fits data points into a a mixture of Gaussian models with alternative calls between an E-step and an M-step
- The E-step determines the assignment probability r_{ij} of each data point to a parent capsule
- The M-step re-calculate the Gaussian models' values based on r_{ij}
- The process is repeated 3 times
- The last a_j will be the parent capsule's output. The 16 μ from the last Gaussian model will be reshaped to form the 4×4 pose matrix of the parent capsule

Loss Function

The authors decided to use the ”spread loss” to directly maximize the gap between the activation of the target class (a_t) and the activation of the other classes.

The loss from the class i (other than the true label t) is defined as:

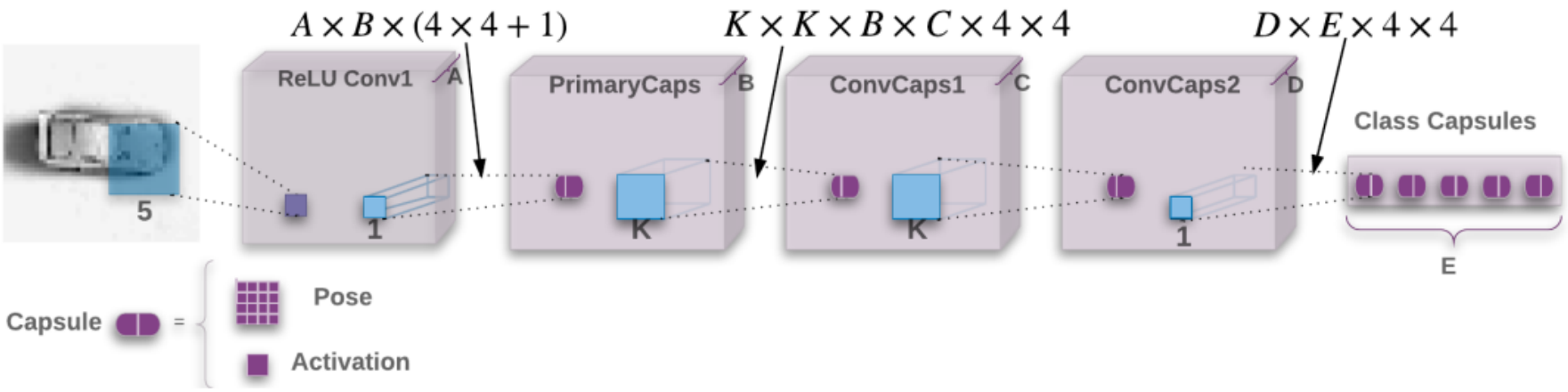
$$L_i = (\max(0, m - (a_t - a_i)))^2$$

which a_t is the activation of the target class (true label) and a_i is the activation for class i. The total cost is:

$$L = \sum_{i \neq t} L_i$$

If the margin between the true label and the wrong class is smaller than m, we penalize it by the square of $m - (a_t - a_i)$. m is initially start as 0.2 and linearly increased by 0.1 after each epoch training. m will stop growing after reaching the maximum 0.9. Starting at a lower margin helps the training to avoid too many dead capsules during the early phase.

CapsNet Architecture



Below is the summary of each layer and the shape of their outputs (for MNIST dataset):

Layer Name	Apply	Output shape
MNist image		28, 28, 1
ReLU Conv1	Regular Convolution (CNN) layer using 5x5 kernels with 32 output channels, stride 2 and padding	14, 14, 32
PrimaryCaps	Modified convolution layer with 1x1 kernels, strides 1 with padding and outputing 32 capsules. Requiring 32x32x(4x4+1) parameters.	pose (14, 14, 32, 4, 4), activations (14, 14, 32)
ConvCaps1	Capsule convolution with 3x3 kernels, strides 2 and no padding. Requiring 3x3x32x32x4x4 parameters.	poses (6, 6, 32, 4, 4), activations (6, 6, 32)
ConvCaps2	Capsule convolution with 3x3 kernels, strides 1 and no padding	poses (4, 4, 32, 4, 4), activations (4, 4, 32)
Class Capsules	Capsule with 1x1 kernel. Requiring 32x10x4x4 parameters.	poses (10, 4, 4), activations (10)

Results on smallNORB dataset

- The smallNORB dataset has gray-level stereo images of 5 classes of toys: airplanes, cars, trucks, humans and animals.

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

Routing iterations	Pose structure	Loss	Coordinate Addition	Test error rate
1	Matrix	Spread	Yes	9.7%
2	Matrix	Spread	Yes	2.2%
3	Matrix	Spread	Yes	**1.8%**
5	Matrix	Spread	Yes	3.9%
3	Vector	Spread	Yes	2.9%
3	Matrix	Spread	No	2.6%
3	Vector	Spread	No	3.2%
3	Matrix	Margin[1]	Yes	3.2%
3	Matrix	CrossEnt	Yes	5.8%
Baseline CNN with 4.2M parameters				5.2%
CNN of Cireşan et al. (2011) with extra input images & deformations				2.56%
Our Best model (third row), with multiple crops during testing				**1.4%**

- The error rate for the Capsule network is generally lower than a CNN model with similar number of layers as shown below.

Table 2: A comparison of the smallNORB test error rate of the baseline CNN and the capsules model on novel viewpoints when both models are matched on error rate for familiar viewpoints.

Test set	Azimuth		Elevation	
	CNN	Capsules	CNN	Capsules
Novel viewpoints	20%	13.5%	17.8%	12.3%
Familiar viewpoints	3.7%	3.7%	4.3%	4.3%

Adversarial Robustness

- The core idea of FGSM (Fast Gradient Sign Method) adversary is to add some noise on every step of optimization to drift the classification away from the target class.
- Compute gradient of output w.r.t. change in pixel intensity, then slightly modifies each pixel by small ϵ in direction that either (1) maximizes loss, or (2) maximizes classification probability of wrong class.
- The authors also tested the model on the slightly more sophisticated adversarial attack of the Basic Iterative Method (BIM), which is simply the aforementioned attack except it takes multiple smaller steps when creating the adversarial image.

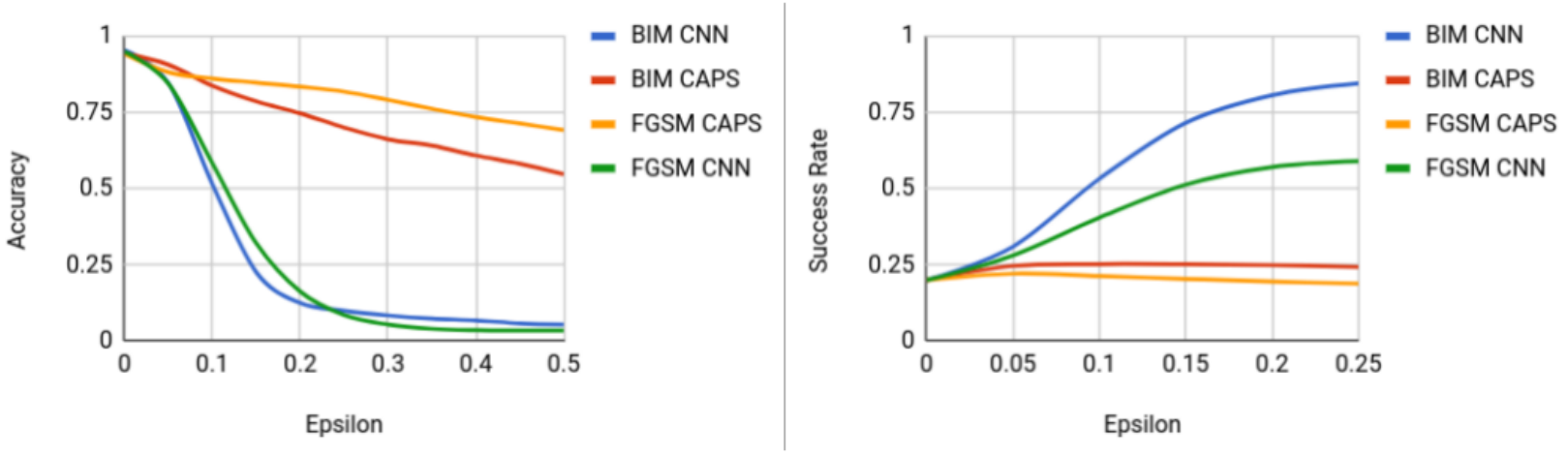


Figure 3: Accuracy against ϵ after an adversarial attack (left) and Success Rate after a targeted adversarial attack (right). The targeted attack results were evaluated by averaging the success rate after the attack for each of the 5 possible classes.

Improvements compared to the original paper

The authors argue that this paper compared to the original Capsules paper overcomes the following deficiencies:

1. The original paper uses the length of the pose vector to represent the probability that the entity represented by a capsule is present. To keep the length less than 1, requires an unprincipled non-linearity and this prevents the existence of any sensible objective function that is minimized by the iterative routing procedure.
2. The original paper also uses the cosine of the angle between two pose vectors to measure their agreement. Unlike the negative log variance of a Gaussian cluster, the cosine saturates at 1, which makes it insensitive to the difference between a quite good agreement and a very good agreement.
3. Finally, the original paper uses a vector of length n rather than a matrix with n elements to represent a pose, so its transformation matrices have n^2 parameters rather than just n.

Conclusion

Discussion

Pros:

- Requires less training data
- Position and pose is preserved (Equivariance)
- Robust affine transformations
- Activation vector is easy (?) to interpret
- Less trainable parameters required (77% less for MNIST)
- Great for overlapping objects
- Good for dealing with segmentation

Discussion

Cons:

- Poor performance (11% error) on CIFAR10; generally bad at complex images.
- Still use regural conv layer at first for local feature extraction (Capsules cannot extract local features?)
- Slow training, due to inner loop (routing by agreement)
- CapsNet does not allow two instances of the same class at the same location
    - The is called ”crowding”, and it has been observed as well in human vision
- Likes to account for everything in the image
- How to restrict to get certain feature? (Disentagling features)
- Requires a lot of further research (Is there any science in Capsule Theory?)

Capsule Networks So Far

There have been around 20 works using capsules in the literature.

- K. Qiao, C. Zhang, L. Wang, B. Yan, J. Chen, L. Zeng, and L. Tong, "Accurate reconstruction of image stimuli from human fMRI based on the decoding model with capsule network architecture," *CoRR*, vol. abs/1801.00602, 2018.
- D. Wang and Q. Liu, "An Optimization View on Dynamic Routing Between Capsules," 2018.
- P. Afshar, A. Mohammadi, and K. N. Plataniotis, "Brain Tumor Type Classification via Capsule Networks," *CoRR*, vol. abs/1802.10200, 2018.
- L. Zhang, M. Edraki, and G. Qi, "CapProNet: Deep Feature Learning via Orthogonal Projections onto Capsule Subspaces," *CoRR*, vol. abs/1805.07621, 2018.
- A. Jaiswal, W. AbdAlmageed, and P. Natarajan, "CapsuleGAN: Generative Adversarial Capsule Network," *arXiv preprint arXiv:1802.06167*, 2018.
- E. Xi, S. Bing, and Y. Jin, "Capsule network performance on complex data," *arXiv preprint arXiv:1712.03480*, 2017.
- R. LaLonde and U. Bagci, "Capsules for Object Segmentation," *arXiv preprint arXiv:1804.04241*, 2018.

Capsule Networks So Far (cont.)

- S. S. R. Phaye, A. Sikka, A. Dhall, and D. R. Bathula, "Dense and Diverse Capsule Networks: Making the Capsules Learn Better," *CoRR*, vol. abs/1805.04001, 2018.
- A. Mobiny and H. Van Nguyen, "Fast CapsNet for Lung Cancer Screening," *arXiv preprint arXiv:1806.07416*, 2018.
- Y. Upadhyay and P. Schrater, "Generative Adversarial Network Architectures For Image Synthesis Using Capsule Networks," *arXiv preprint arXiv:1806.03796*, 2018.
- M. T. Bahadori, "Spectral Capsule Networks," *International Conference on Learning Representations (ICLR Workshop)*, 2018.
- Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu, "Sentiment Analysis by Capsules," in *Proceedings of the 2018 World Wide Web Conference*, WWW '18, (Republic and Canton of Geneva, Switzerland), pp. 1165–1174, International World Wide Web Conferences Steering Committee, 2018.
- K. Duarte, Y. S. Rawat, and M. Shah, "VideoCapsuleNet: A Simplified Network for Action Detection," *arXiv preprint arXiv:1805.08162*, 2018.
- F. Deng, S. Pu, X. Chen, Y. Shi, T. Yuan, and S. Pu, "Hyperspectral Image Classification with Capsule Network Using Limited Training Samples," *Sensors (Basel)*, vol. 18, Sep 2018.
- H. Li, X. Guo, B. Dai, W. Ouyang, and X. Wang, "Neural Network Encapsulation," 08 2018.

Capsule Networks So Far (cont.)

- C. Xiang, L. Zhang, W. Zou, Y. Tang, and C. Xu, "MS-CapsNet: A Novel Multi-Scale Capsule Network," *IEEE Signal Processing Letters*, pp. 1–1, 2018.
- A. Deliège, A. Cioppa, and M. Van Droogenbroeck, "HitNet: a neural network with capsules embedded in a Hit-or-Miss layer, extended with hybrid data augmentation and ghost capsules," *arXiv preprint arXiv:1806.06519*, 2018.
- J. O. Neill, "Siamese capsule networks," *arXiv preprint arXiv:1805.07242*, 2018.
- Z. Chen and D. Crandall, "Generalized Capsule Networks with Trainable Routing Procedure," *arXiv preprint arXiv:1808.08692*, 2018.
- W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, and Z. Zhao, "Investigating Capsule Networks with Dynamic Routing for Text Classification," *arXiv preprint arXiv:1804.00538*, 2018.

Questions?