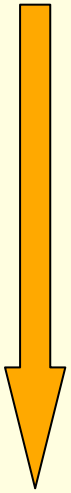
Does the Brain do Inverse Graphics?

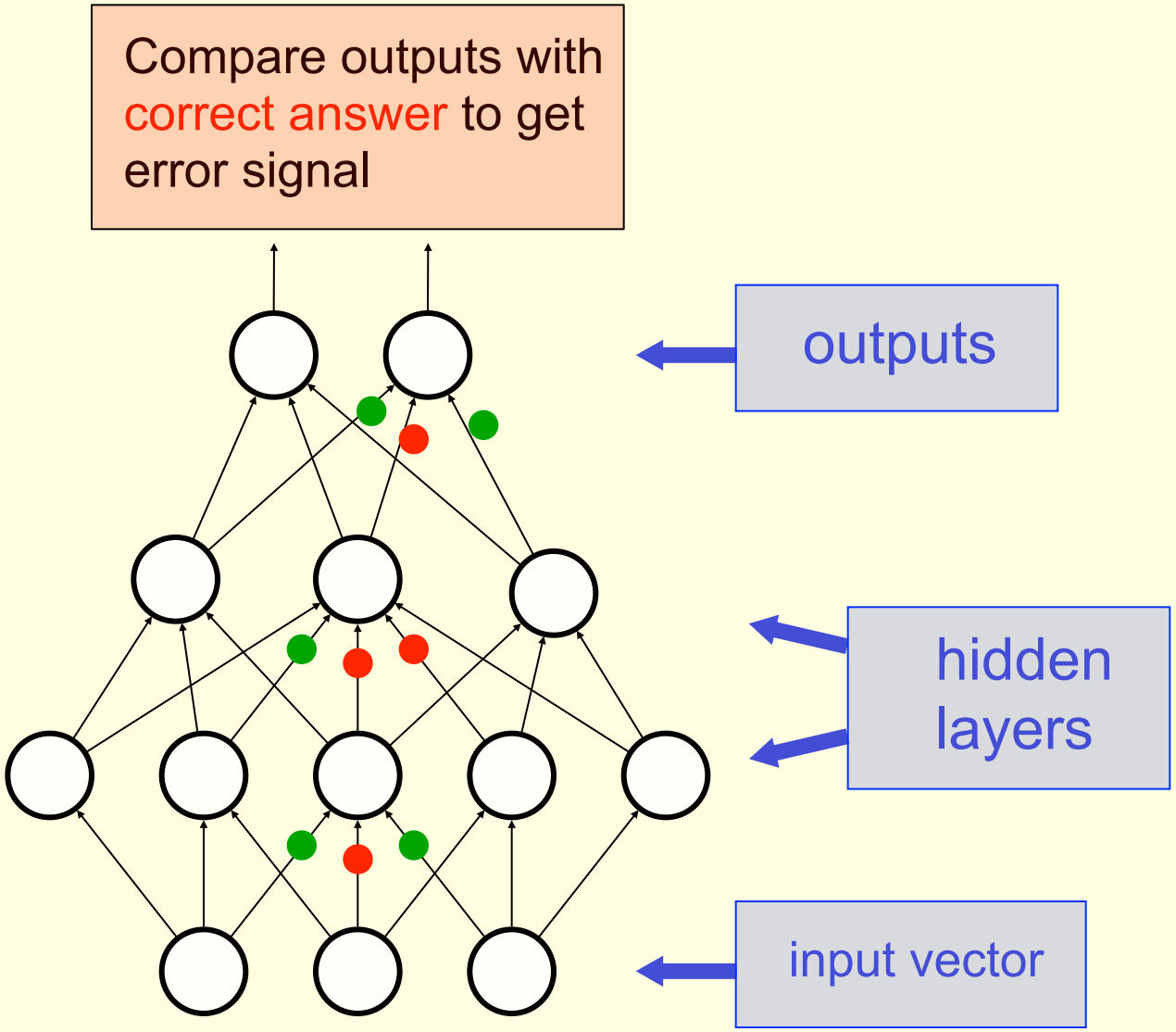
Geoffrey Hinton, Alex Krizhevsky, Navdeep Jaitly,
Tijmen Tieleman & Yichuan Tang

Department of Computer Science
University of Toronto

How to learn many layers of features (~1985)

Back-propagate error signal to get derivatives for learning





The label poverty problem

- Its hard to get a large number of accurately labeled examples.
  - Also, each label typically only contains a few bits of information to constrain the mapping from inputs to outputs.
- Solution 1: Work very hard to get lots of labels.
- Solution 2: Learn features from unlabeled data by building a generative model of the images.

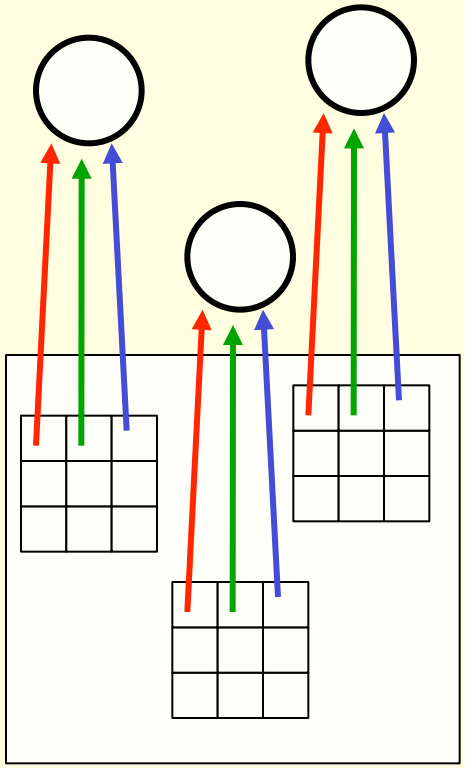
The representation used by the neural nets that work best for recognition (Yann LeCun)

- Convolutional neural nets use multiple layers of feature detectors that have local receptive fields and shared weights.
- The feature extraction layers are interleaved with sub-sampling layers that throw away information about precise position in order to achieve some translation invariance.

The replicated feature approach

- Use many different copies of the same feature detector.
  - The copies all have slightly different positions.
  - Replication reduces the number of free parameters to be learned.
- Use several different feature types, each with its own replicated pool of detectors.
  - Allows each patch of image to be represented in several ways.

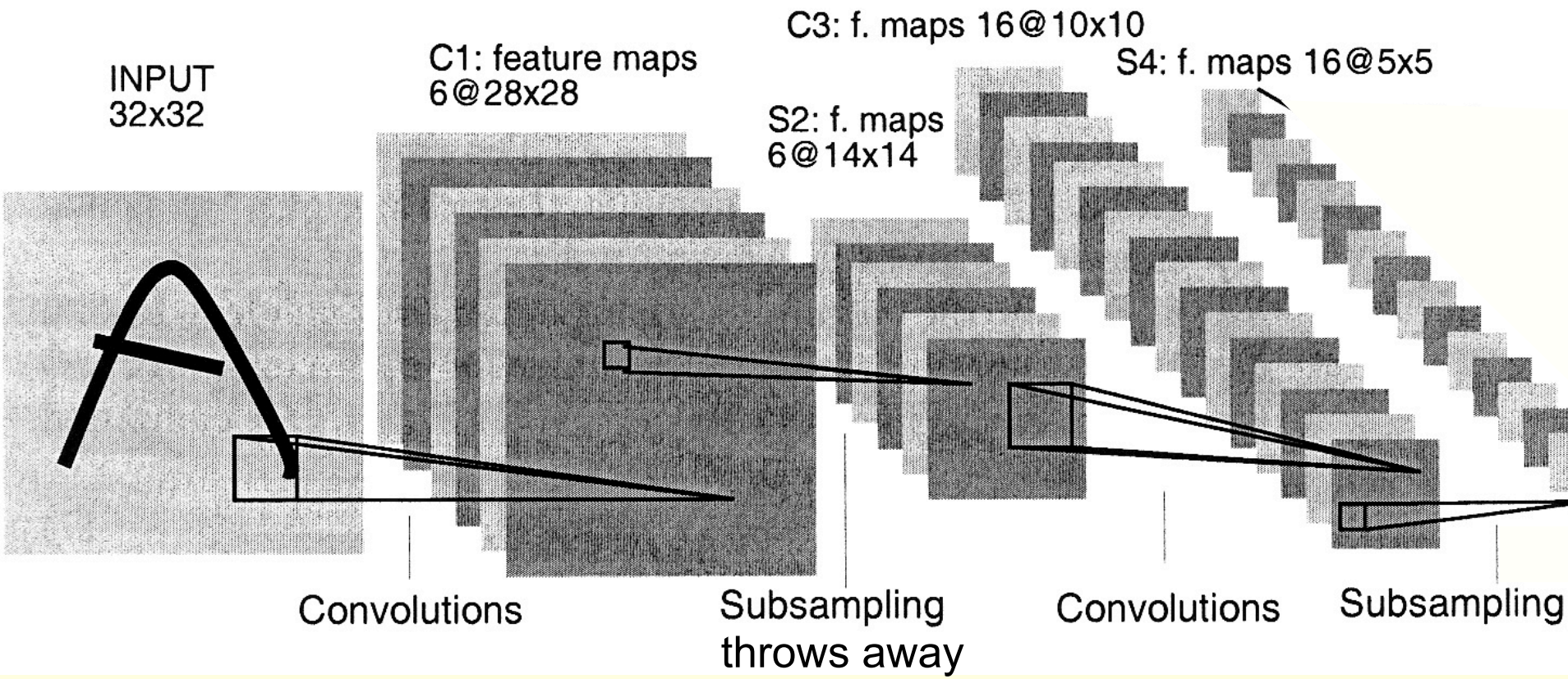
The red connections all have the same weight.



Combining the outputs of replicated features

- Get a small amount of translational invariance at each level by averaging some neighboring replicated detectors to give a single output to the next level.
  - This reduces the number of inputs to the next layer of feature extraction, thus allowing us to have many more different feature pools.
  - Taking the maximum of the four works slightly better.

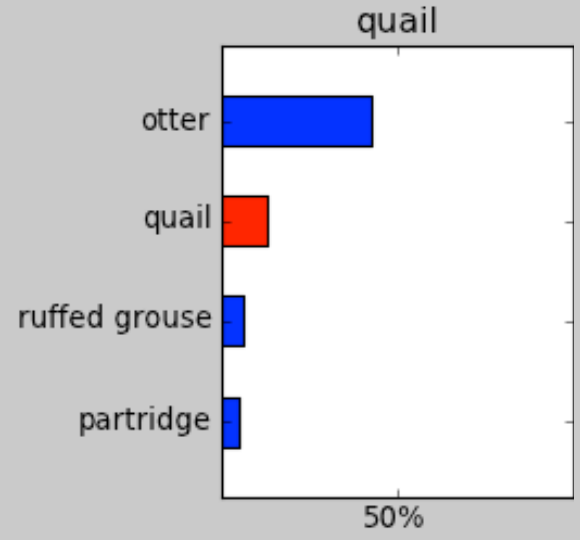
The architecture of LeNet5

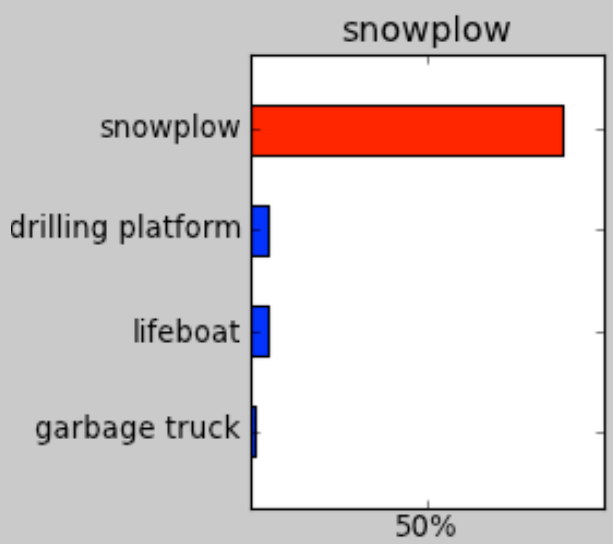


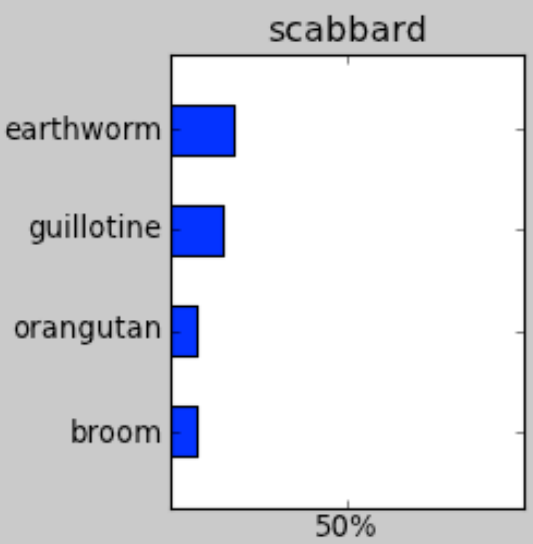
Recognizing objects in images: The Imagenet benchmark
(recent work by Alex Krizhevsky)

- Scaling up to large datasets is much easier with neural networks than with other methods.
  - They can learn to deal with a huge range of variation.
- Recognizing a thousand different types of object only requires about a million training images.
  - That is about one week of visual experience at four fixations per second for 12 hrs per day.
- The neural net requires quite a lot of engineering, but then it gives 10% less errors than the very best hand-tailored computer vision system.

Some examples from an earlier version of the net







Why convolutional neural networks are doomed

- Convolutional nets are doomed for two reasons:
  - Sub-sampling loses the precise spatial relationships between higher-level parts such as a nose and a mouth. The precise spatial relationships are needed for identity recognition
    - But overlapping the sub-sampling pools mitigates this.
  - They cannot extrapolate their understanding of geometric relationships to radically new viewpoints.

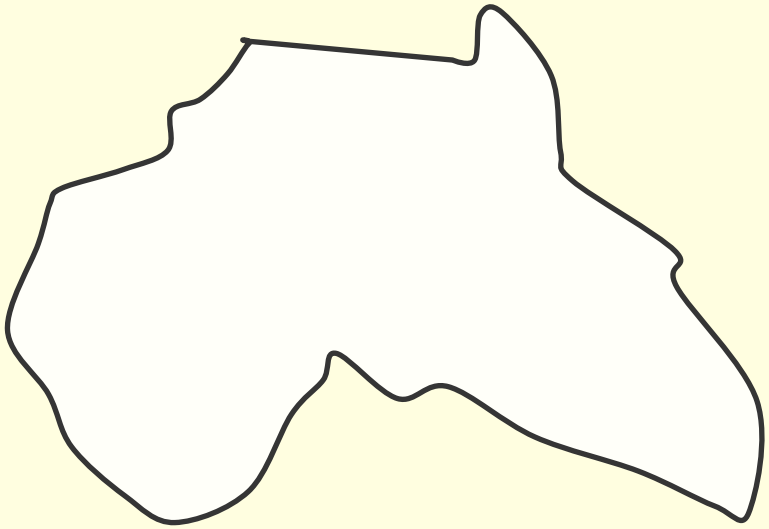
Equivariance vs Invariance

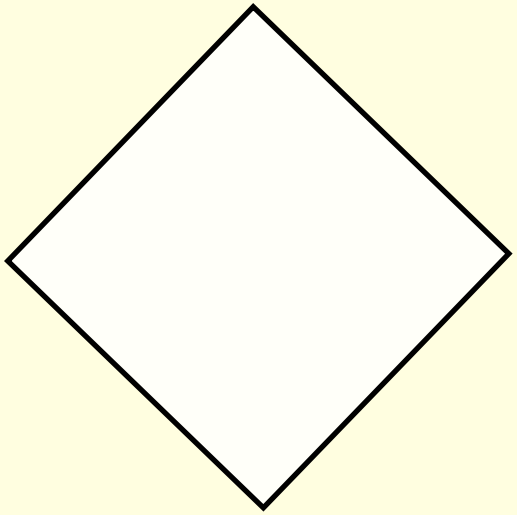
- Sub-sampling tries to make the neural activities invariant for small changes in viewpoint.
  - This is a silly goal, motivated by the fact that the final label needs to be viewpoint-invariant.
- Its better to aim for equivariance: Changes in viewpoint lead to corresponding changes in neural activities.
  - In the perceptual system, its the weights that code viewpoint-invariant knowledge, not the neural activities.

What is the right representation of images?

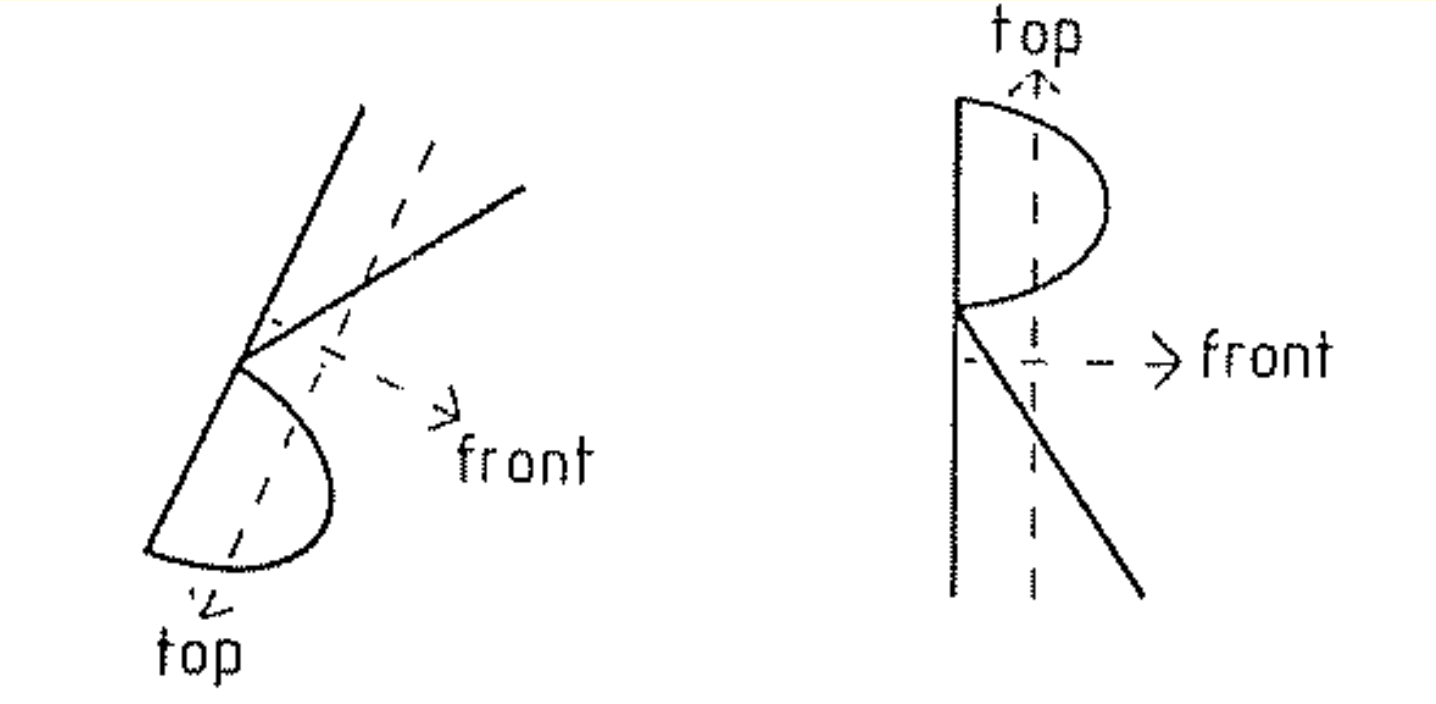
- Computer vision is inverse computer graphics, so the higher levels of a vision system should look like the representations used in graphics.
- Graphics programs use hierarchical models in which spatial structure is modeled by matrices that represent the transformation from a coordinate frame embedded in the whole to a coordinate frame embedded in each part.
  - These matrices are totally viewpoint invariant.
  - This representation makes it easy to compute the relationship between a part and the retina from the relationship between a whole and the retina.

Some psychological evidence that our visual systems impose coordinate frames in order





Mental rotation:
More evidence for coordinate frames



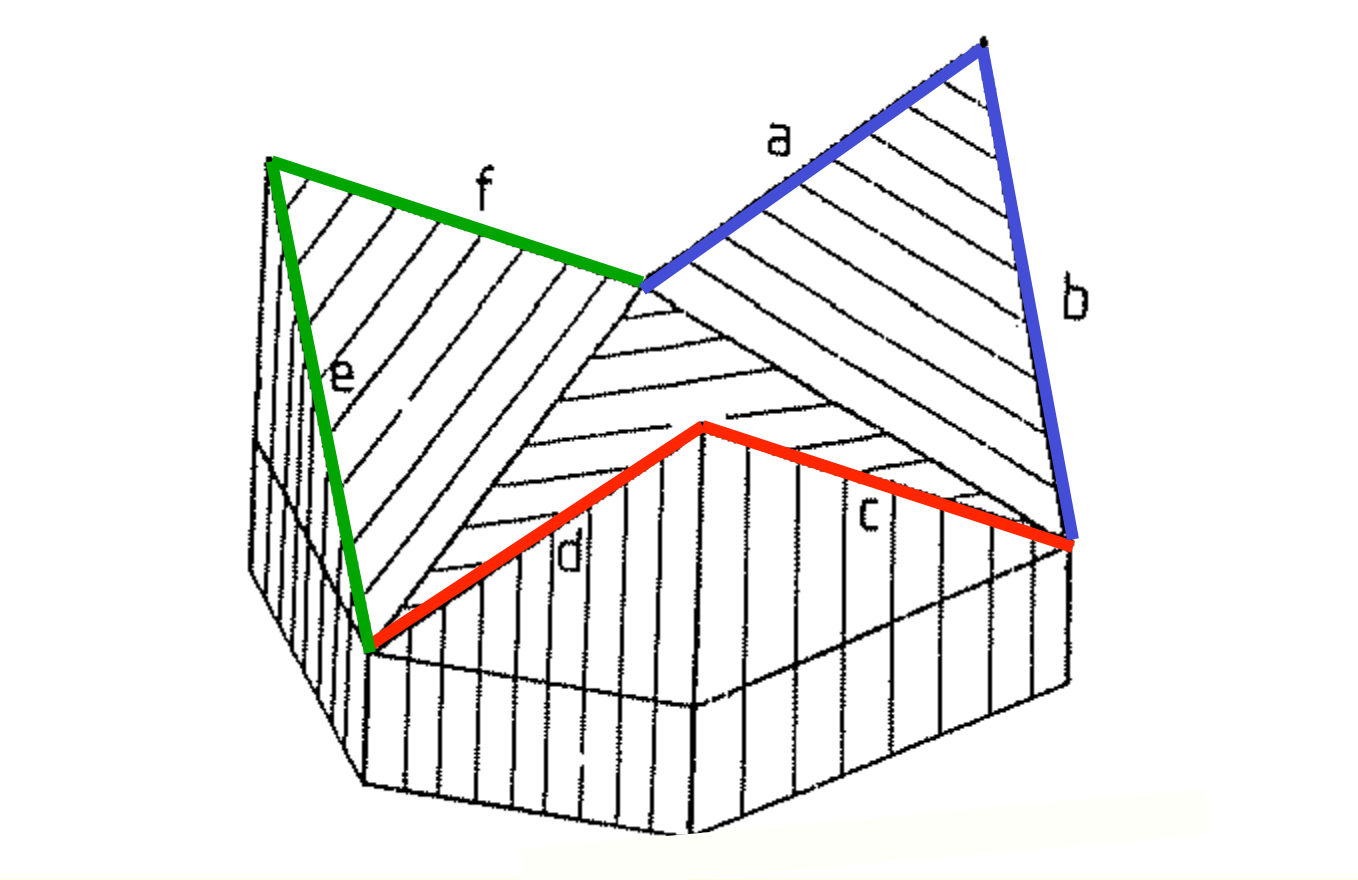
We perform mental rotation to decide if the tilted R has the correct handedness, not to recognize that it is an R.

But why do we need to do mental rotation to decide handedness?

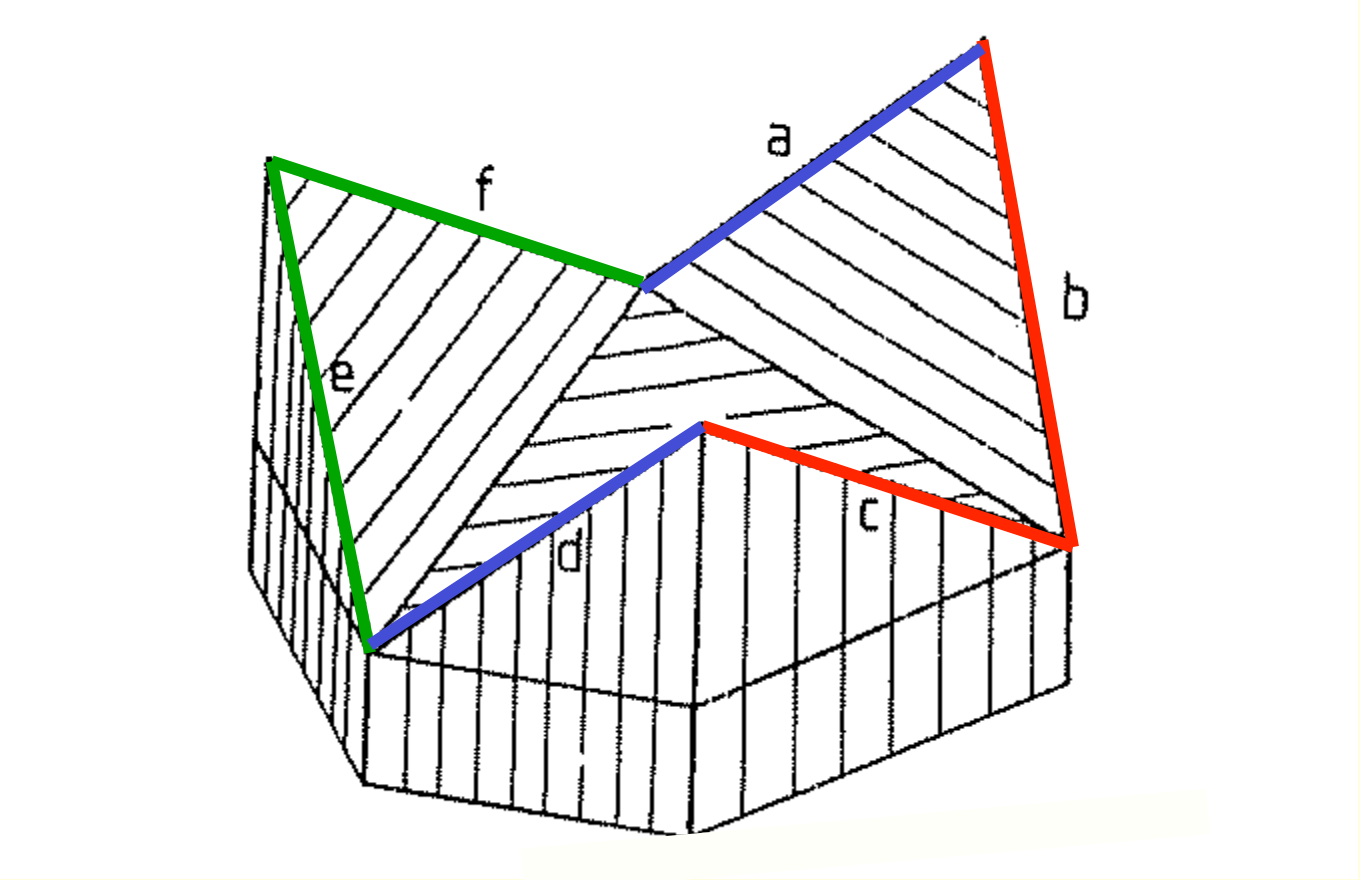
What mental rotation achieves

- It is very difficult to compute the handedness of a coordinate transformation by looking at the individual elements of the matrix.
  - The handedness is the sign of the determinant of the matrix relating the object to the viewer.
  - This is a high-order parity problem.
- To avoid this computation, we rotate to upright *preserving handedness*, then we look to see which way it faces when it is in its familiar orientation.
- If we had individual neurons that represented a whole pose, we would not have a problem with identifying handedness, because these neurons would have a handedness.

An arrangement of 6 rods



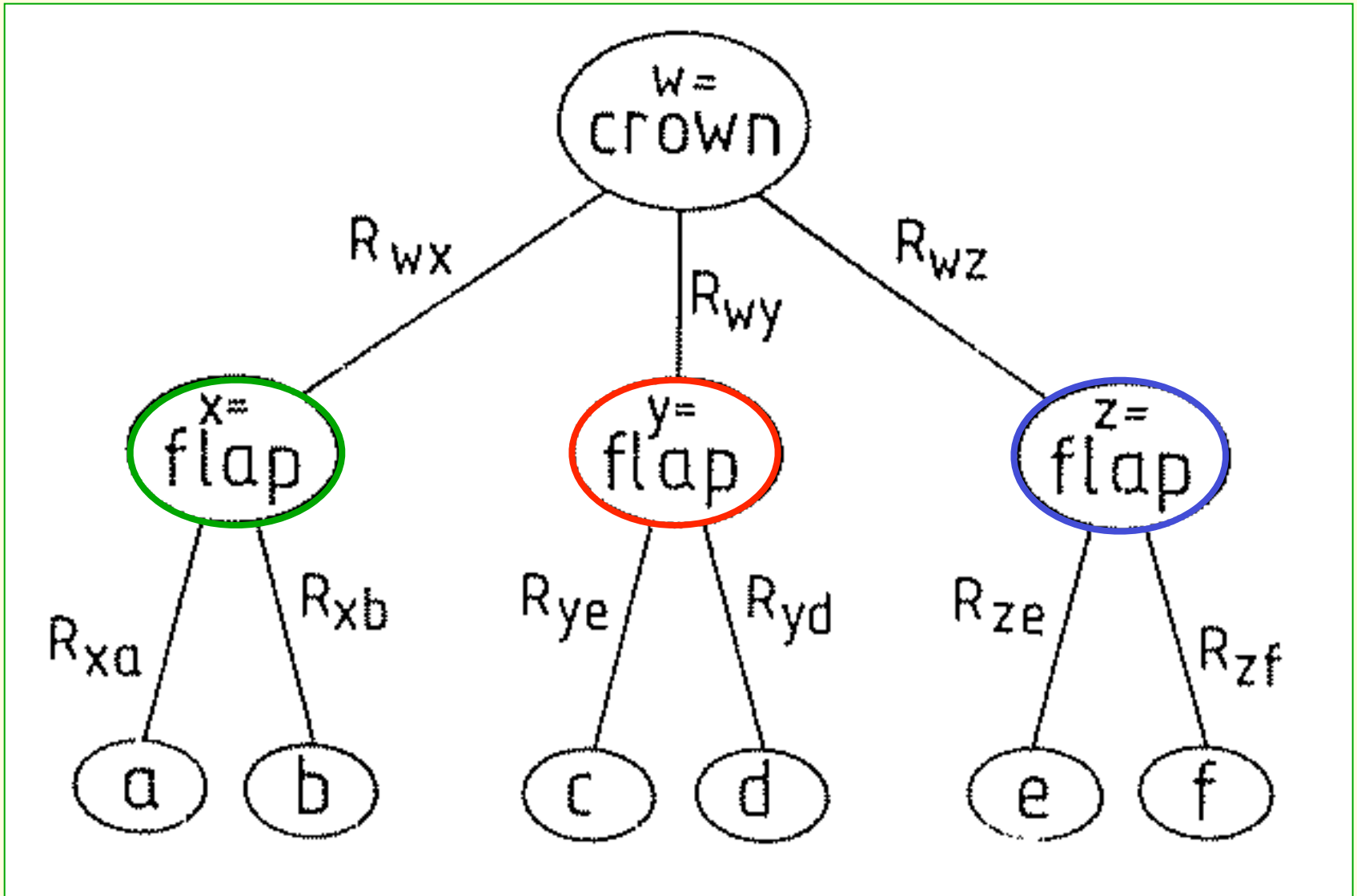
A different percept of the 6 rods



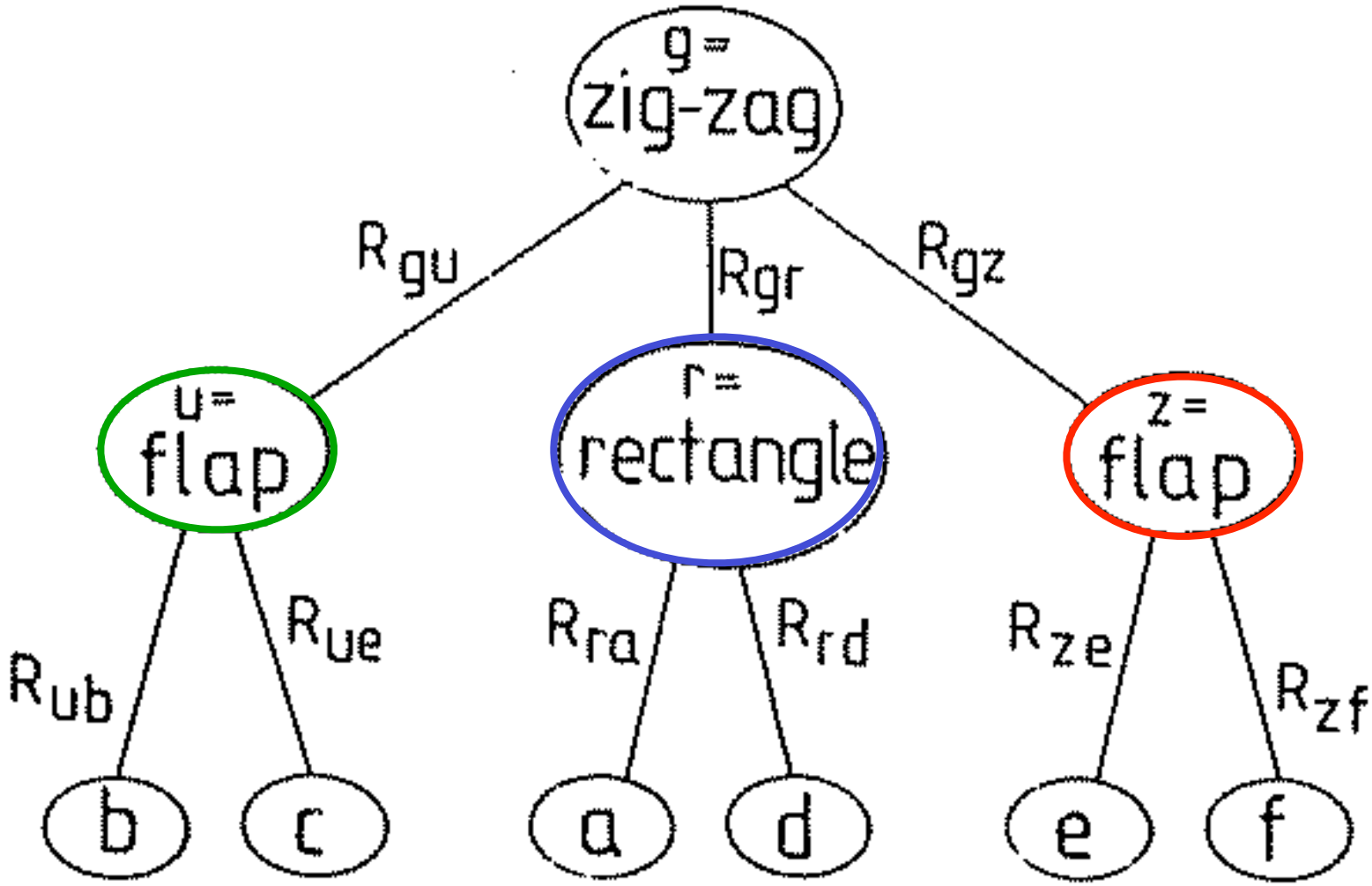
Alternative representations

- The very same arrangement of rods can be represented in quite different ways.
  - Its not like the Necker cube where the alternative percepts disagree on depth.
- The alternative percepts do not disagree, but they make different facts obvious.
  - In the zig-zag representation it is obvious that there is one pair of parallel edges.
  - In the crown representation there are no obvious pairs of parallel edges because the edges do not

A structural description of the “crown” formed by the six rods

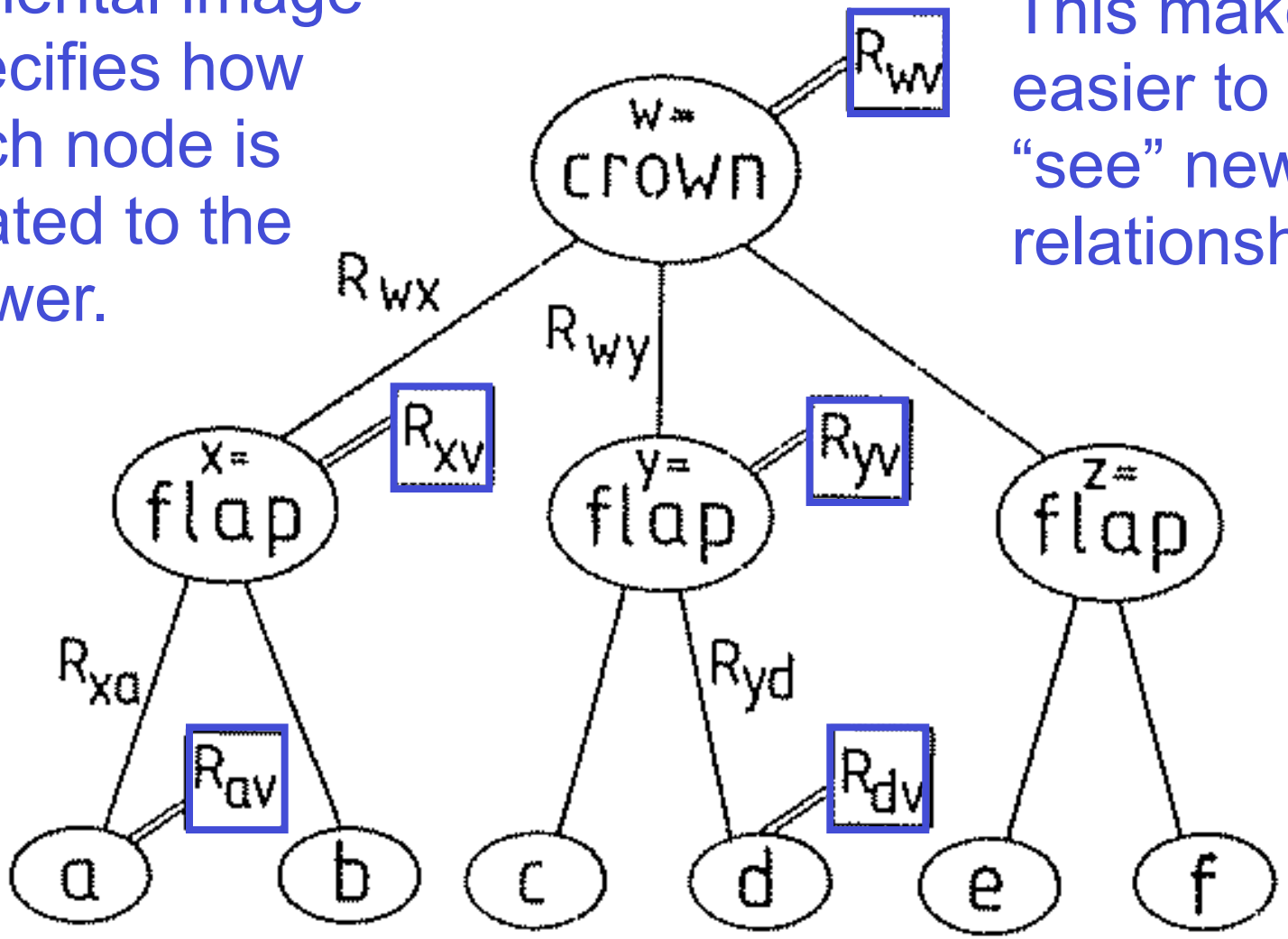


A structural description of the “zig-zag”



A mental image of the crown

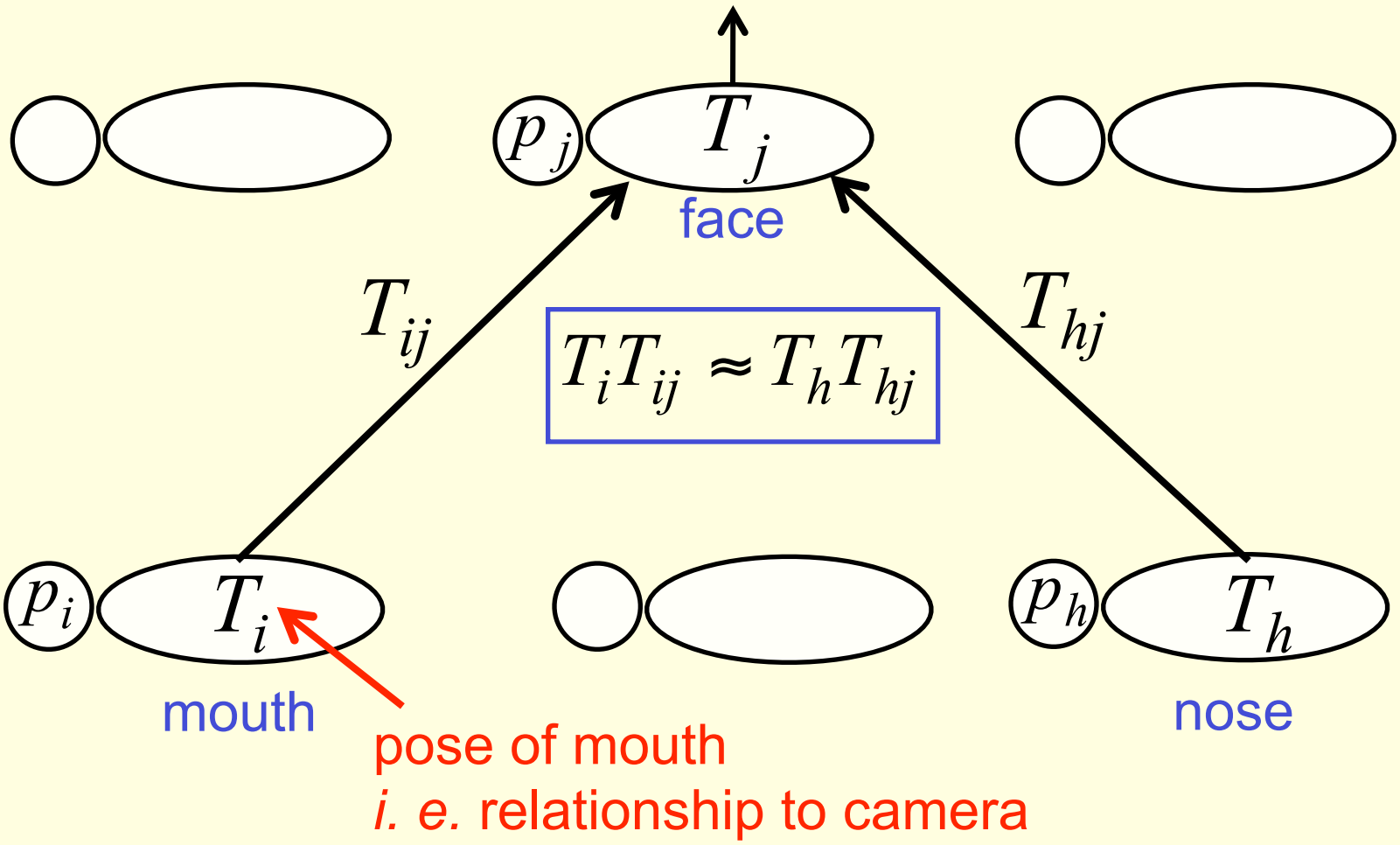
A mental image specifies how each node is related to the viewer.



This makes it easier to "see" new relationships

Two layers in a hierarchy of parts

- A higher level visual entity is present if several lower level visual entities can agree on their predictions for its pose.

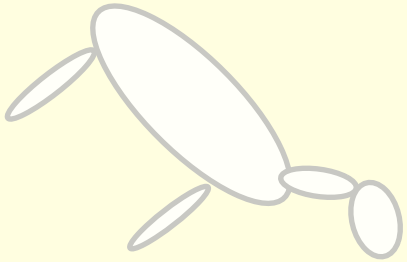


A crucial property of the pose vectors

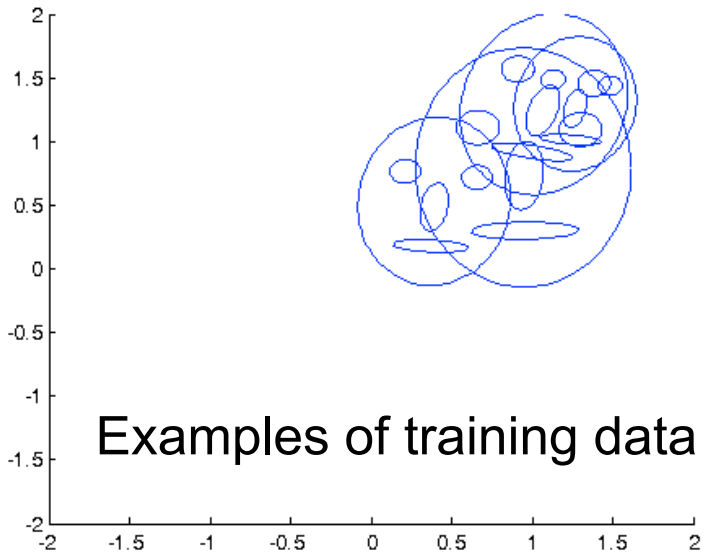
- They allow spatial transformations to be modeled by linear operations.
  - This makes it easy to learn a hierarchy of visual entities.
  - It makes it easy to generalize across viewpoints.

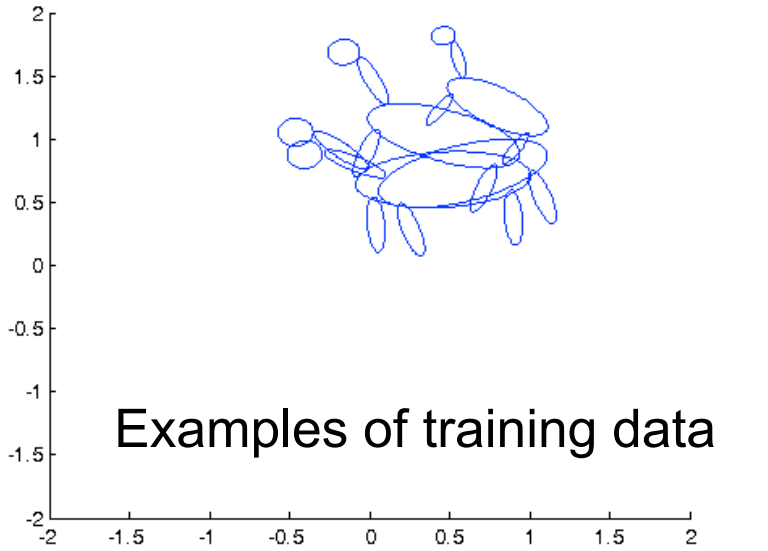
A toy example of what we could do if we could reliably extract the poses of parts of objects

- Consider images composed of five ellipses.
  - The shape is determined entirely by the spatial relations between the ellipses because all parts have the same shape.
- Can we sets of spatial relationships from data that contains several different shapes?
  - Can we generalize far beyond the range of variation in the training examples?
  - Can we learn without being told which ellipse corresponds to which part of a shape?

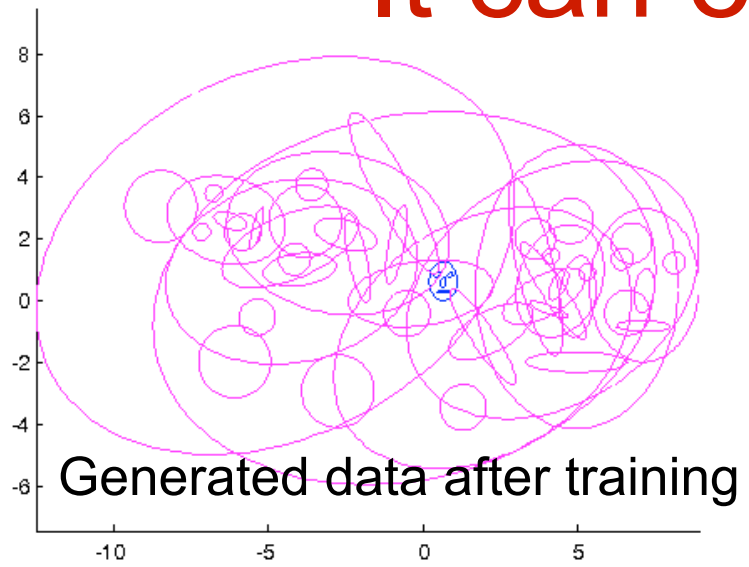


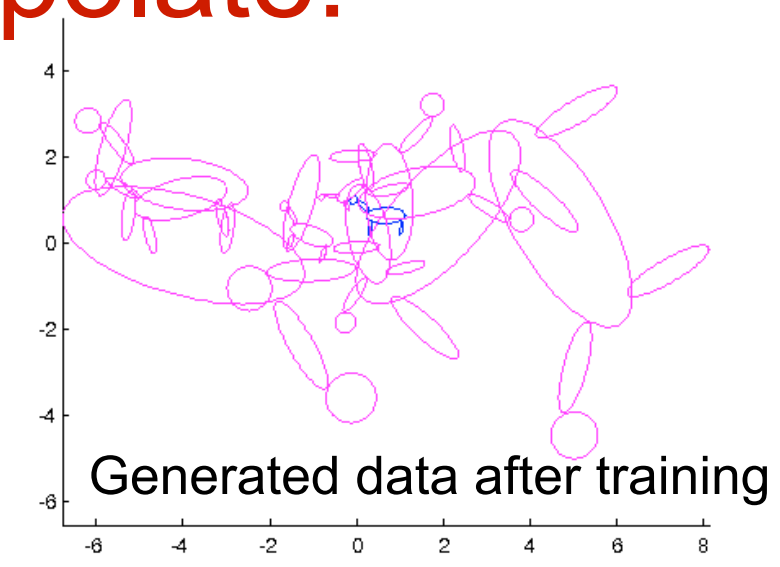
Examples of two shapes (Yichuan Tang)





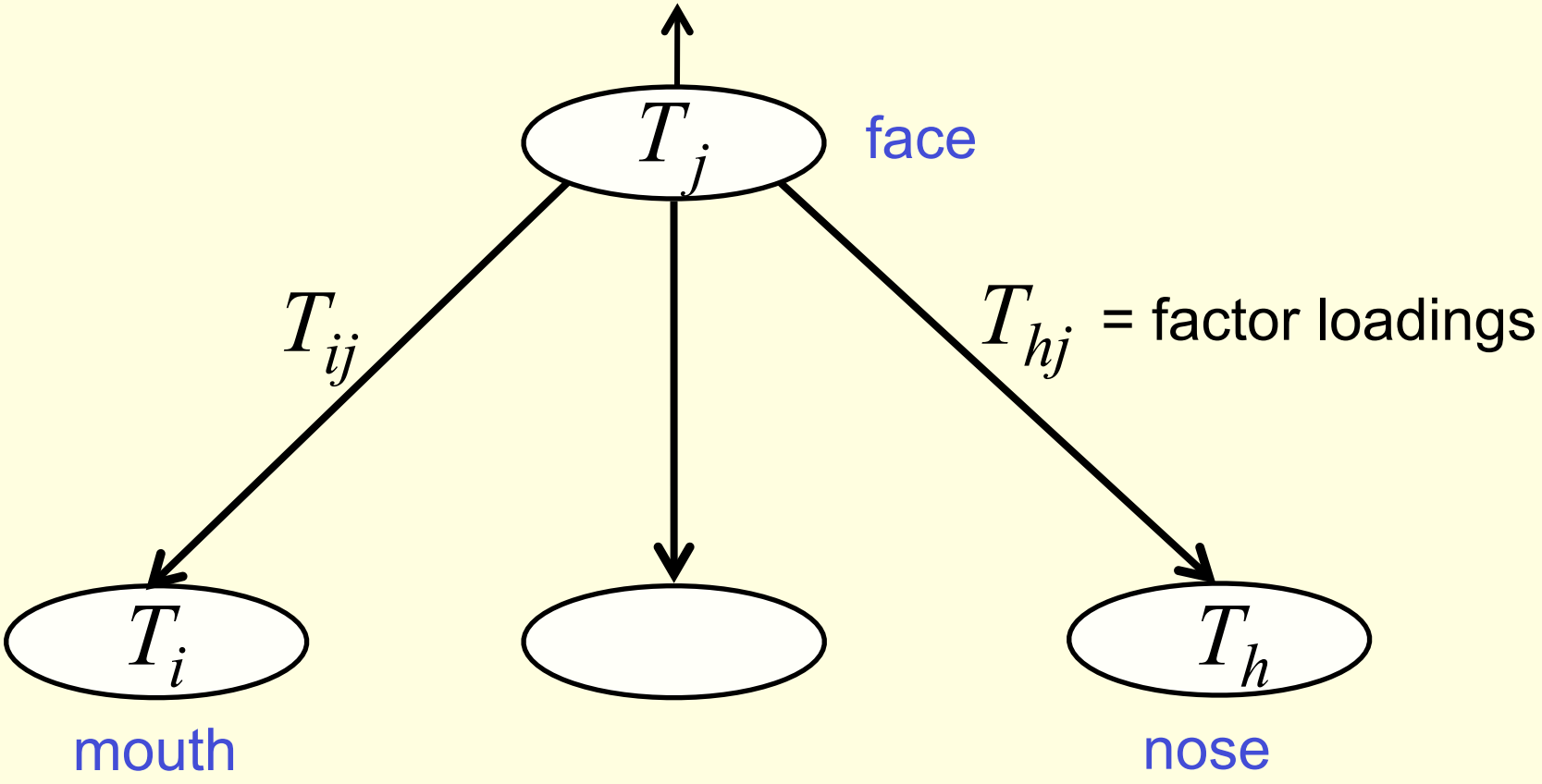
It can extrapolate!





Learning one shape using factor analysis

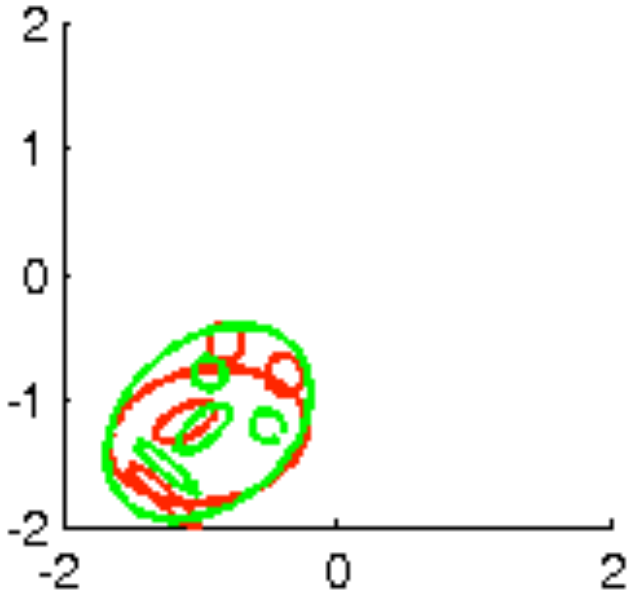
- The pose of the whole can predict the poses of all the parts. This generative model is easy to fit using the EM algorithm for factor analysis.

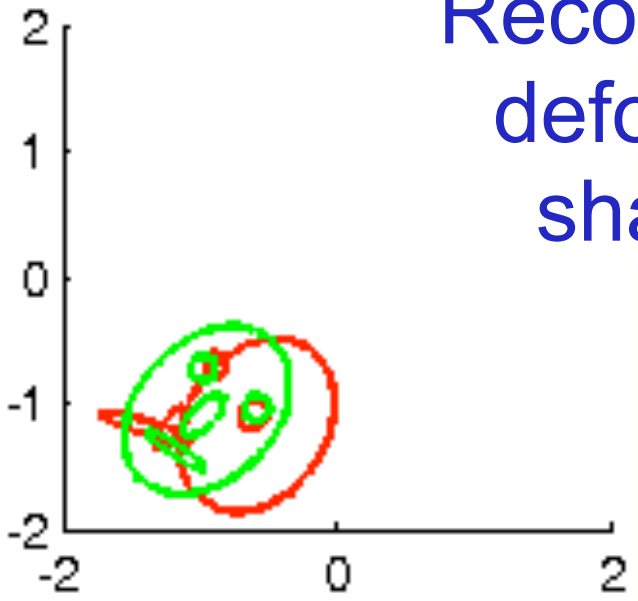


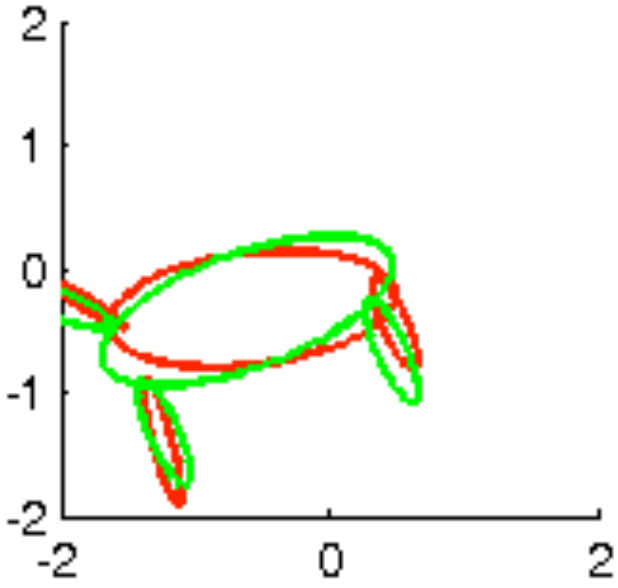
Fitting a mixture of shapes

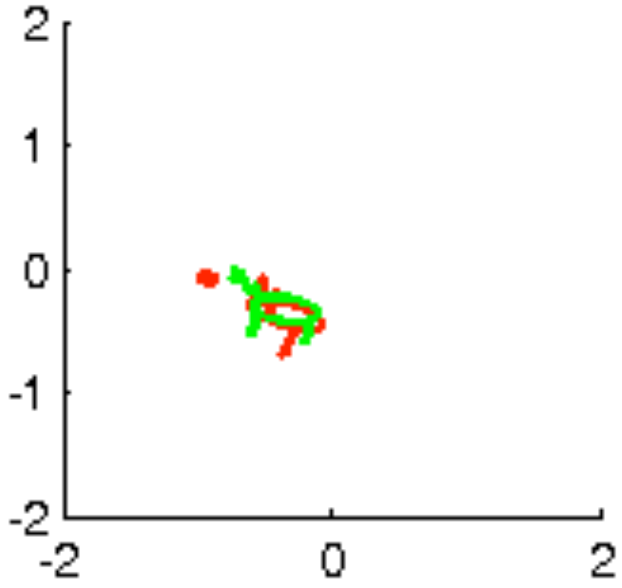
- We can use a mixture of factor analysers (MFA) to fit images that contain many different shapes so long as each image only contains one example of one shape.
- If we do not know how to assign the ellipses to the parts of a known shape we can try many different assignments and pick the one that gives the best reconstruction.
  - Then we assume that is the correct assignment and use that assignment for learning.

Recognizing deformed shapes







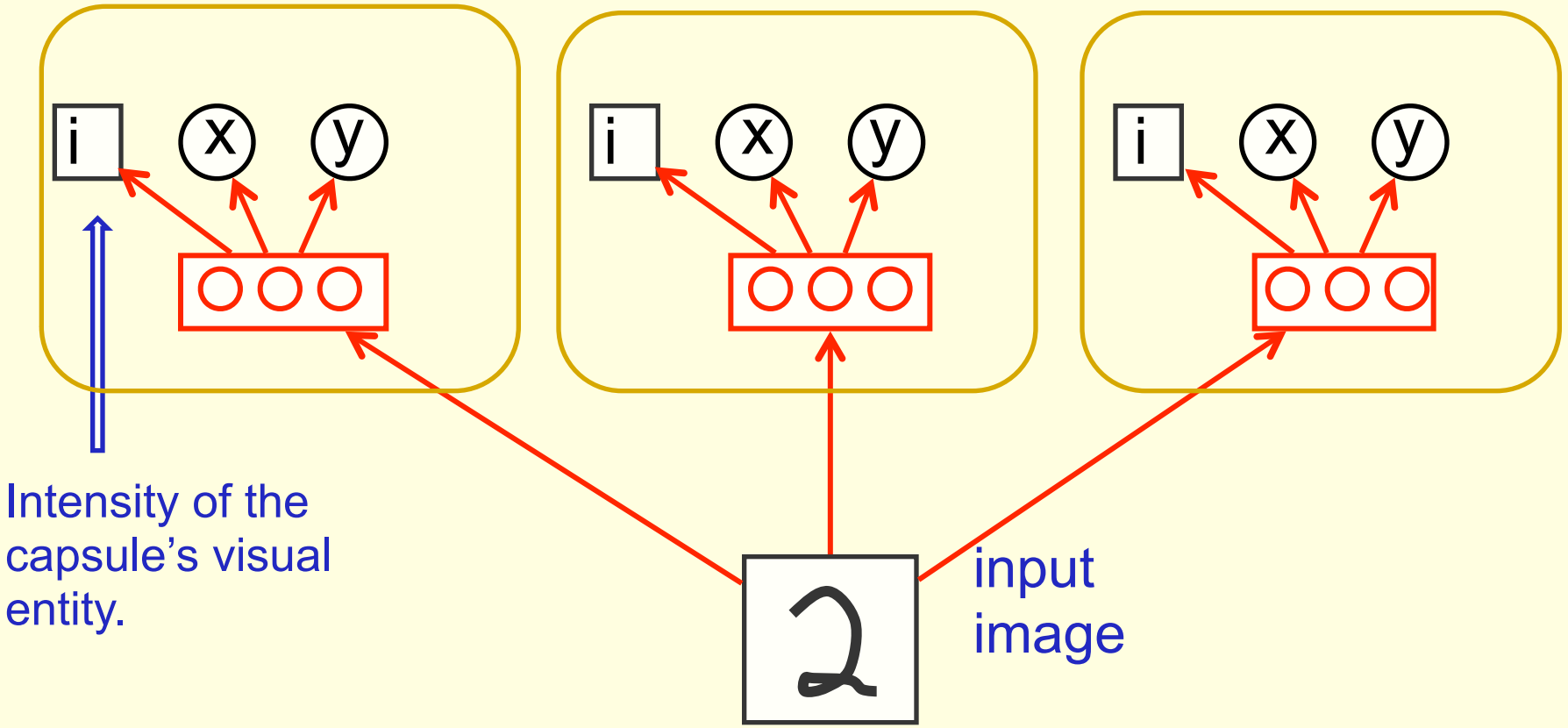


Two big problems

- How do we get from pixels to the first level parts that output explicit pose parameters?
  - We do not have any labeled data.
  - This stage has to be very non-linear.

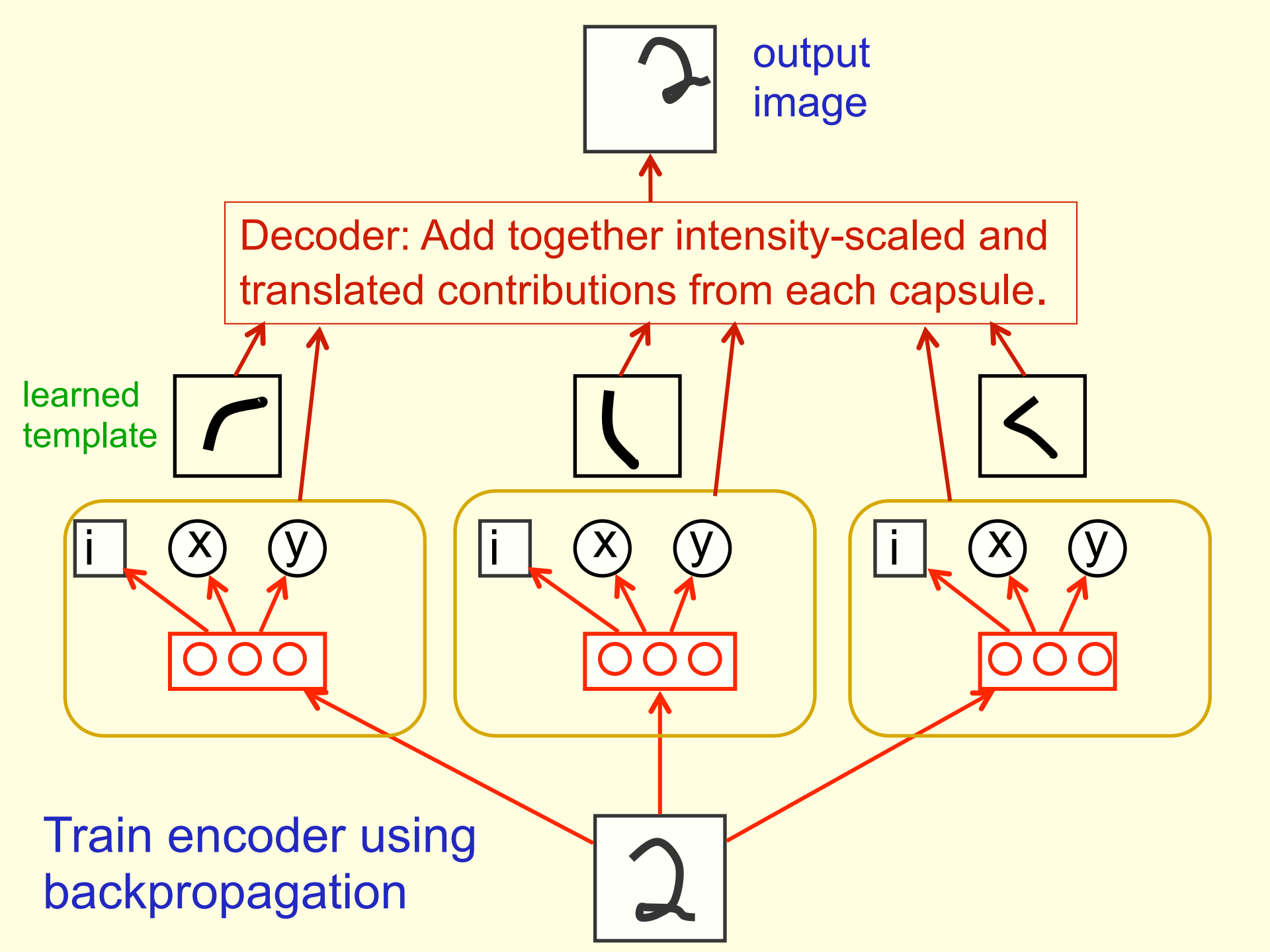
- How does the brain do all this linear algebra rapidly and accurately?
  - Can the brain really do the linear algebra?
  - Can it even communicate an approximate real number?

A picture of three capsules



Extracting pose information by using a domain specific decoder (Navdeep Jaitly & Tijmen Tieleman)

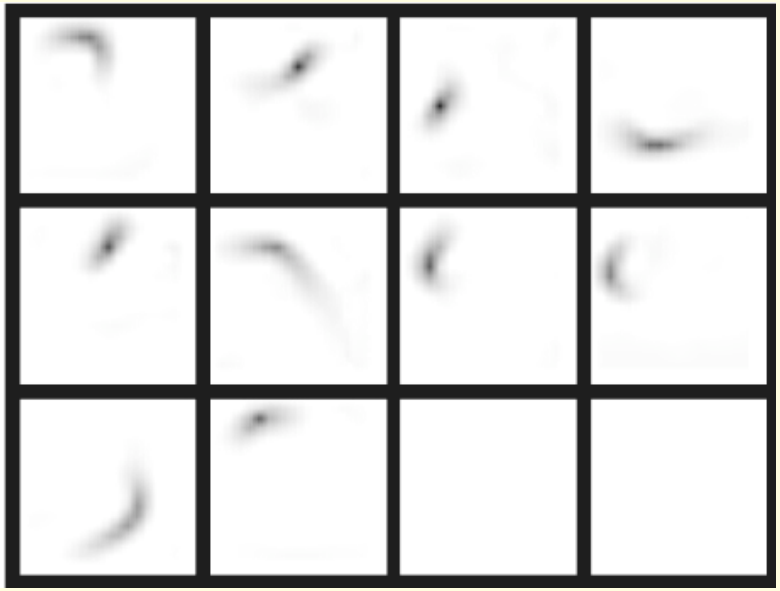
- The idea is to define a simple decoder that produces an image by adding together contributions from each capsule.
- Each capsule learns a fixed “template” that gets intensity-scaled and translated differently for reconstruction each image.
- The encoder must learn to extract the appropriate intensity and translation for each capsule from the input image.


output
image
Decoder: Add together intensity-scaled and
translated contributions from each capsule.
learned
template
i
x
y
i
x
y
i
x
y
Train encoder using
backpropagation

The templates learned by the autoencoder

Each template is multiplied by a case-specific intensity and translated by a case-specific $\Delta x,\ \Delta y$

Then it is added to the output image.



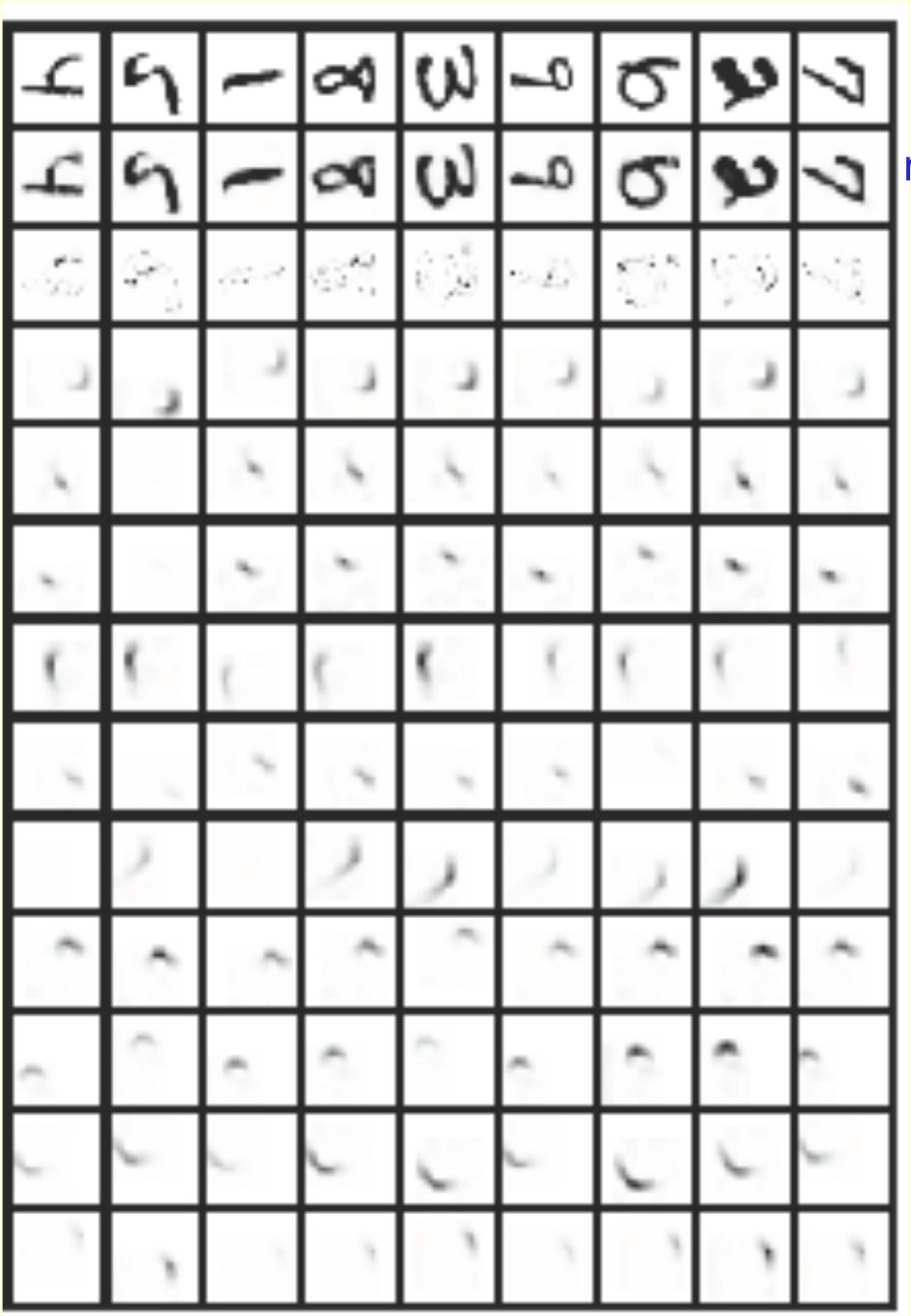
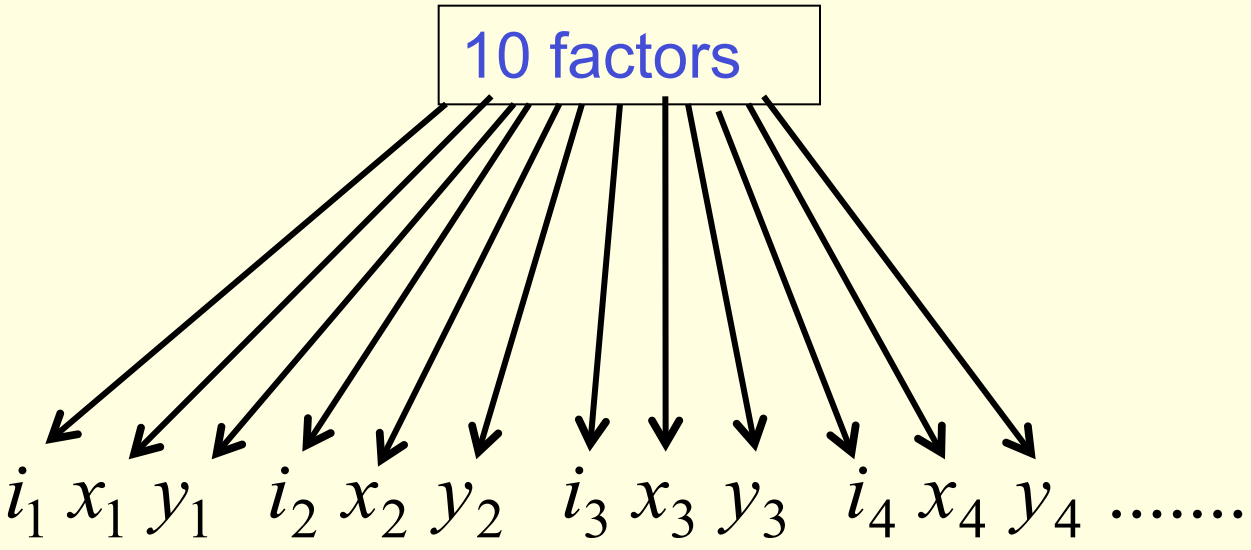


image
reconstruction
recon. error
capsule 1
capsule 2
capsule 3
capsule 4

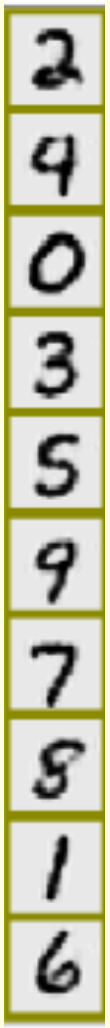
Modeling the capsule outputs with a factor analyser



We learn a mixture of 10 factor analysers on unlabeled data. What do the means look like?

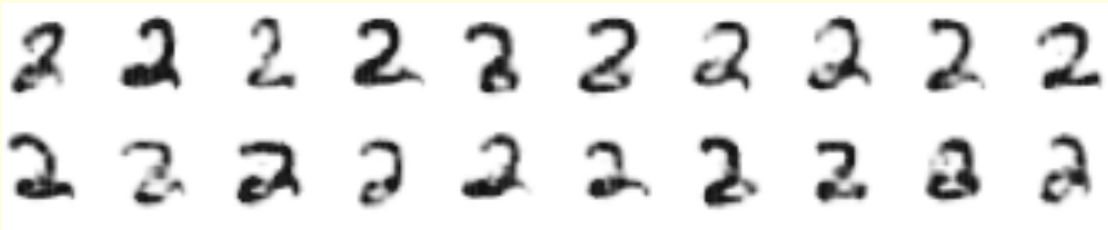
Means discovered by a mixture of factor analysers

directly on pixels



on outputs of capsules

mean - 2 σ of each factor



mean + 2σ of each factor

Another simple way to learn the lowest level parts

- Use pairs of images that are related by a known coordinate transformation
  - *e.g.* a small translation of the image.
- We often have non-visual access to image transformations
  - *e.g.* When we make an eye-movement.
- Cats learn to see much more easily if they control the image transformations (Held & Hein)

Learning the lowest level capsules

- We are given a pair of images related by a known translation.

Step 1: Compute the capsule outputs for the first image.

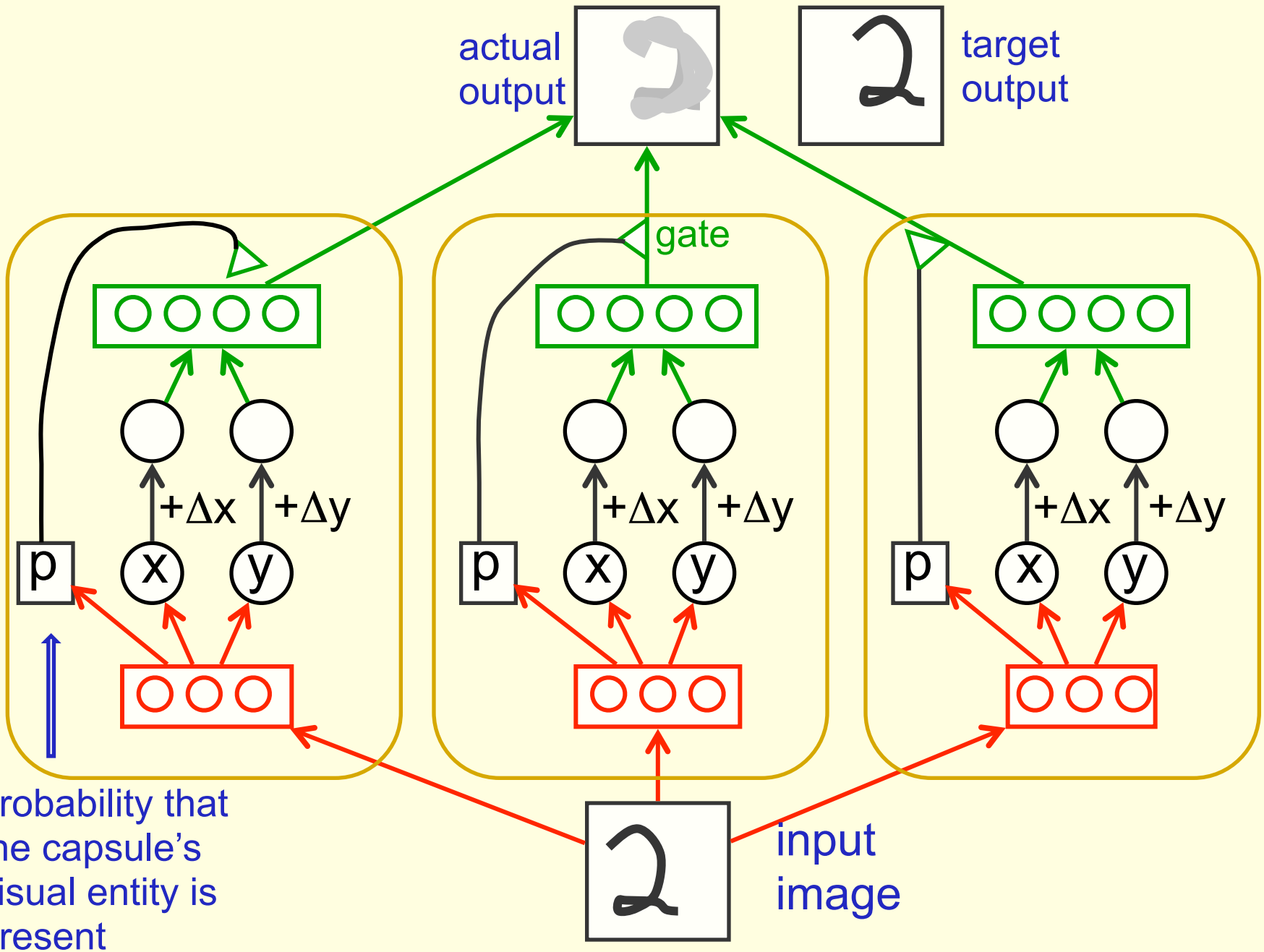
- Each capsule uses its own set of "recognition" hidden units to extract the x and y coordinates of the visual entity it represents (and also the probability of existence)

Step 2: Apply the transformation to the outputs of each capsule

- Just add Δx to each x output and Δy to each y output

Step 3: Predict the transformed image from the transformed outputs of the capsules

- Each capsule uses its own set of "generative" hidden units to compute its contribution to the prediction.


actual
output
target
output
gate
+Δx
+Δy
p
x
y
probability that
the capsule's
visual entity is
present
input
image

Why it has to work

- When the net is trained with back-propagation, the only way it can get the transformations right is by using x and y in a way that is consistent with the way we are using Δx and Δy.

- This allows us to force the capsules to extract the coordinates of visual entities without having to decide what the entities are or where they are.

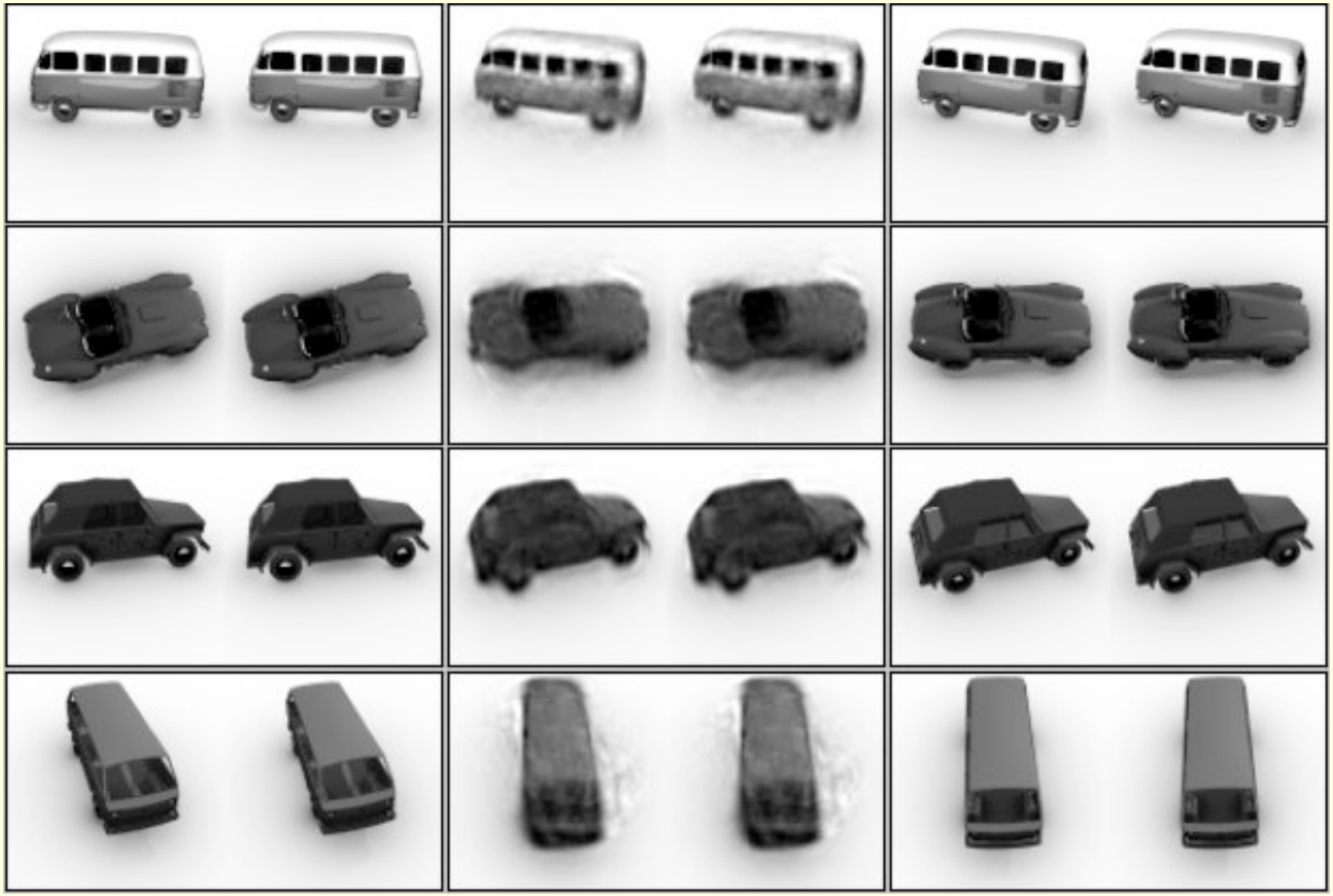
Dealing with the three–dimensional world

- Use stereo images to allow the encoder to extract 3-D poses.
  - Using capsules, 3-D would not be much harder than 2-D if we started with 3-D pixels.
  - The loss of the depth coordinate is a separate problem from the complexity of 3-D geometry.

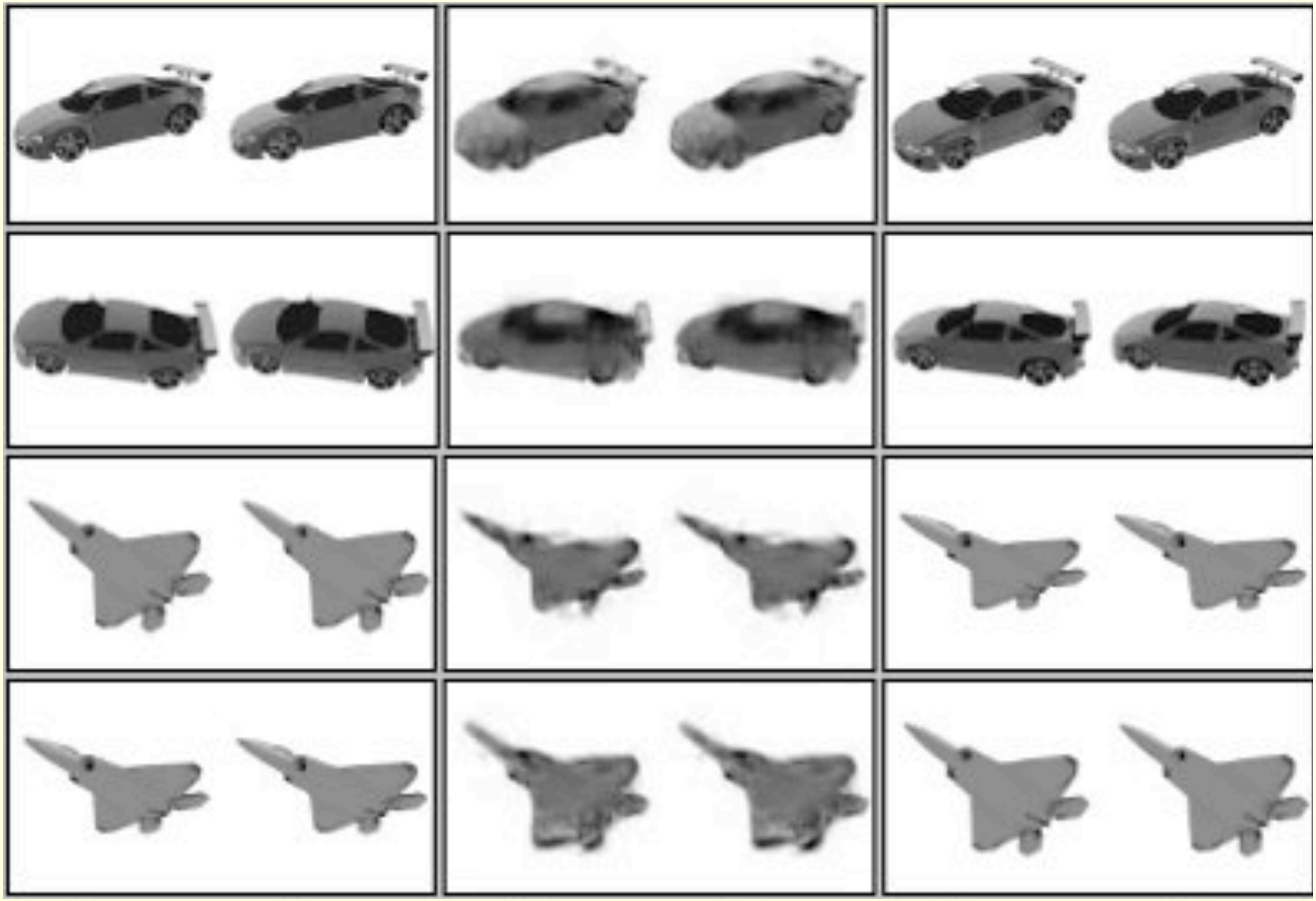
- At least capsules stand a chance of dealing with the 3-D geometry properly.
  - Convolutional nets in 3-D are a nightmare.

Dealing with 3-D viewpoint (Alex Krizhevsky)

Input stereo pair | Output stereo pair | Correct output



It also works on test data



But can the brain do the linear algebra required for inverse graphics?

- If neural activities are used to represent and communicate poses, surely the brain needs to communicate approximate real numbers?
  - But neurons only send a 1 or a 0 on the 10ms time-scale that we need.
- For a device that does massive amounts of signal processing, it seems crazy not to use real numbers.

Cut?

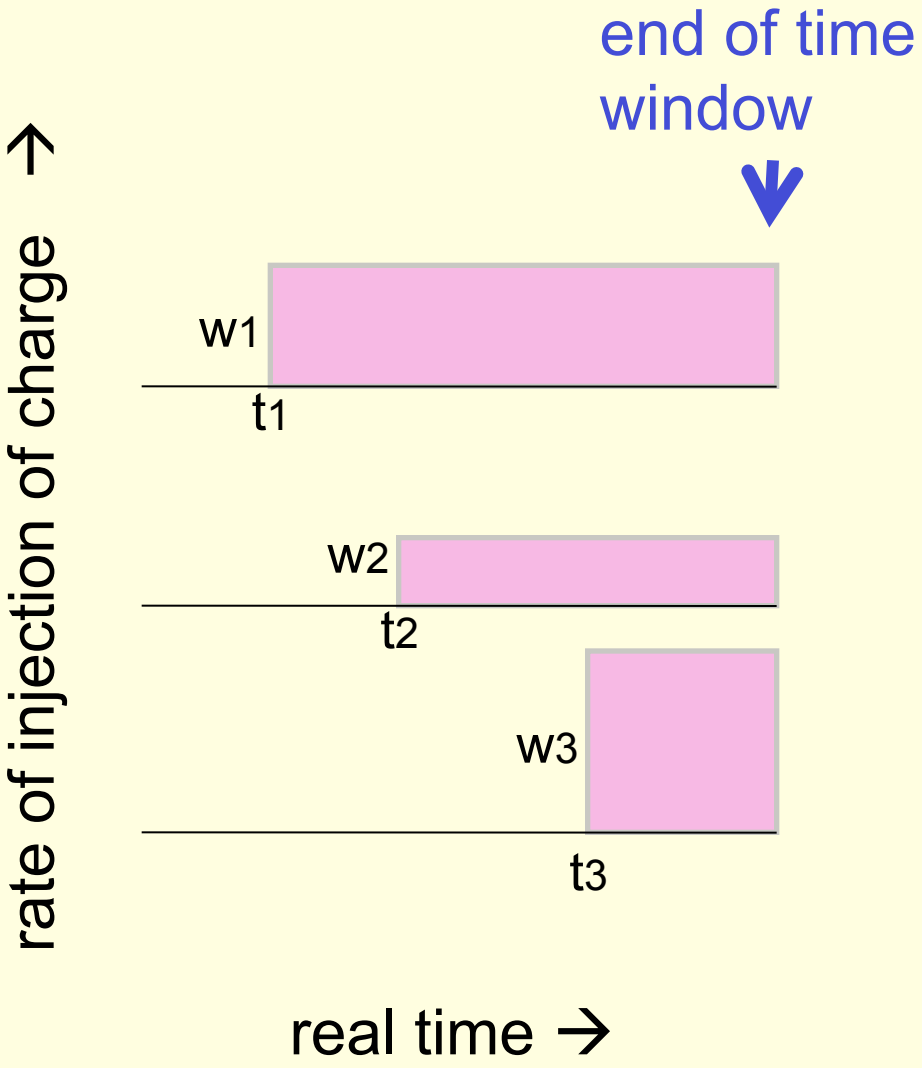
How spike timing could be used to compute scalar products

- To compute a pose parameter after a coordinate transform, we need to take the scalar product of a vector of pose parameters (coded as spike times) with a vector of transformation parameters (coded as synaptic weights).

- The answer must then be converted back to a spike time.

Cut?

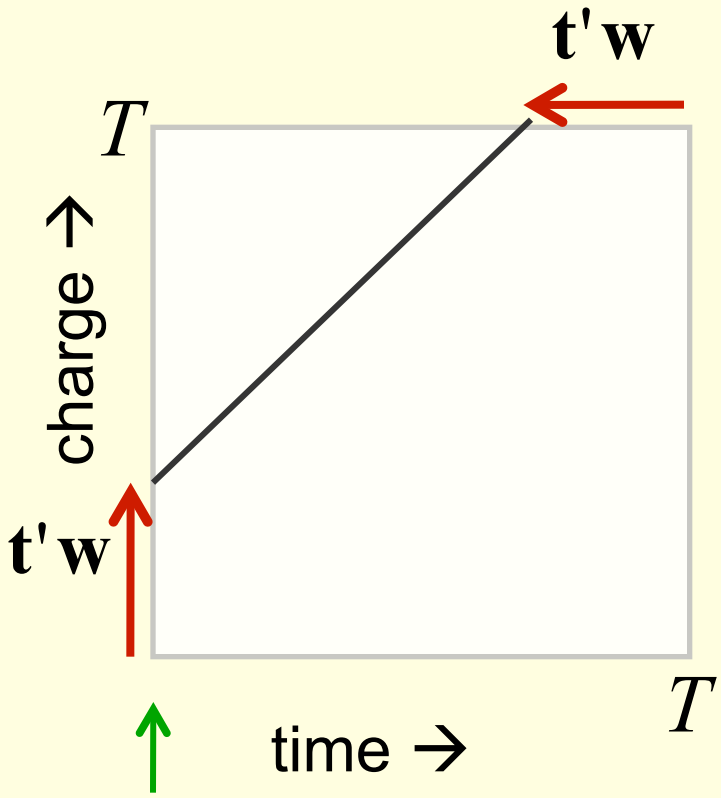
A picture of how an integrate-and-fire neuron could compute a scalar product



- At the end of the time window, the total injected charge is the scalar product of the time advances and the weights.
- But how do we convert this back into a spike time?

Cut?

Converting accumulated charge into a spike time



end of first time window

- At the end of the time window, add an additional input that injects charge at the rate:

$$1 - \sum_i w_i$$

- The total rate of injection of charge is then 1 and so the additional time taken to reach a threshold of T is: $T - \mathbf{t'w}$
- So in the next window, the advance of the outgoing spike represents the scalar product $\mathbf{t'w}$

Cut?

Does the brain really make use of spike times to communicate real numbers?

- In the hippocampus of a rat, the location of the rat is represented by place cells.
  - The precise time at which a place cell fires probably indicates where the rat is within that place field.
- But there is not much evidence that spike times are used this way in sensory cortex. This leaves two main possibilities:
  - Evolution failed to discover an obvious trick.
  - Our ideas about signal processing are hopelessly wrong.

the end