Why equivariance is better than premature invariance

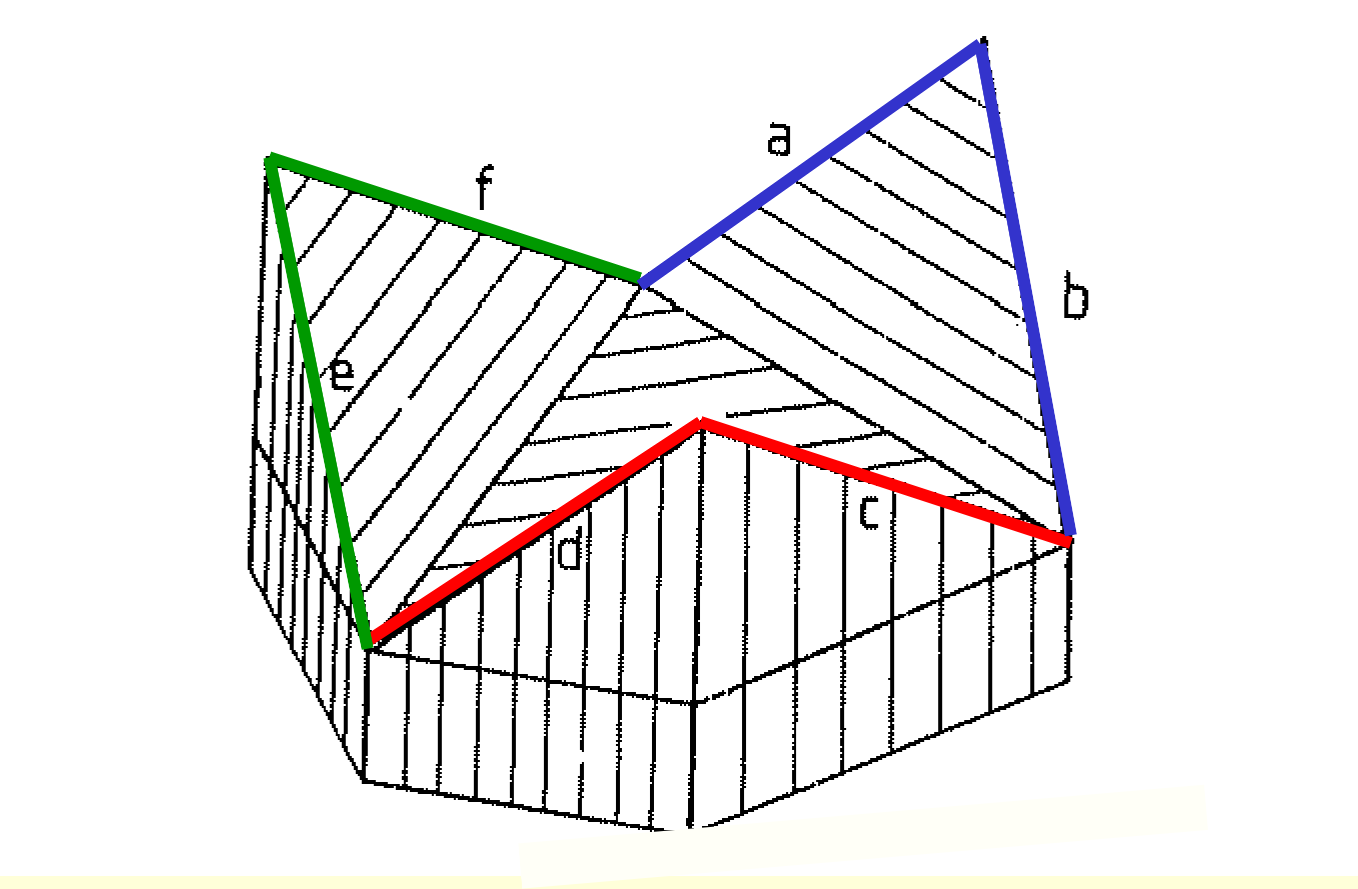
Geoffrey Hinton
Canadian Institute for Advanced Research
&
Department of Computer Science
University of Toronto

with contributions from
Sida Wang and Alex Krizhevsky

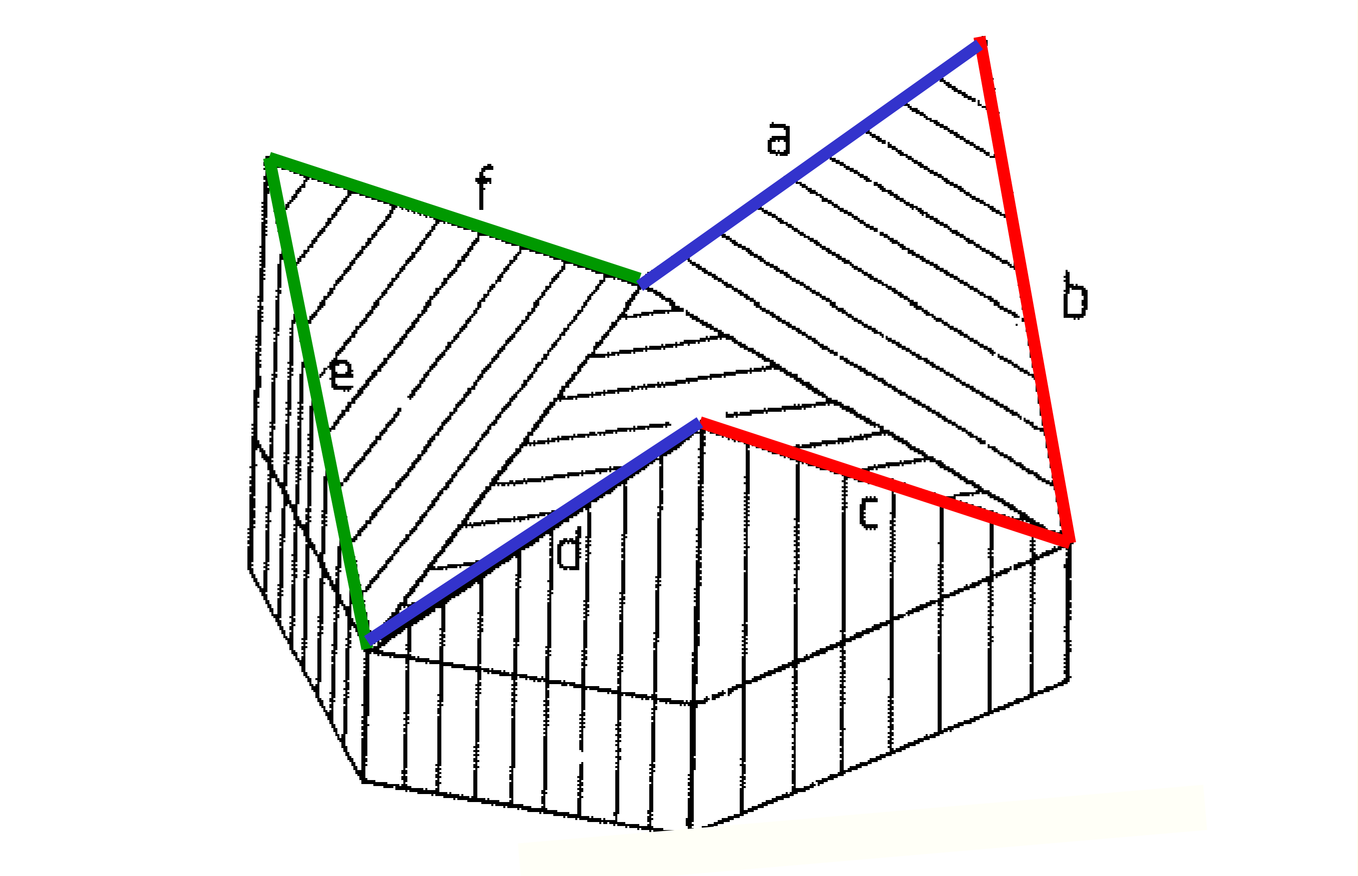
What is the right representation of images?

- Computer vision is inverse graphics, so the higher levels should look like the representations used in graphics.
  - Graphics programs use hierarchical models in which matrices are used to represent the spatial relationships between wholes and parts.
  - The generative models of images that are currently used by neural networks researchers do not look like graphics programs.
- There is a lot of psychological evidence that people use hierarchical structural descriptions to represent images.

An arrangement of 6 rods



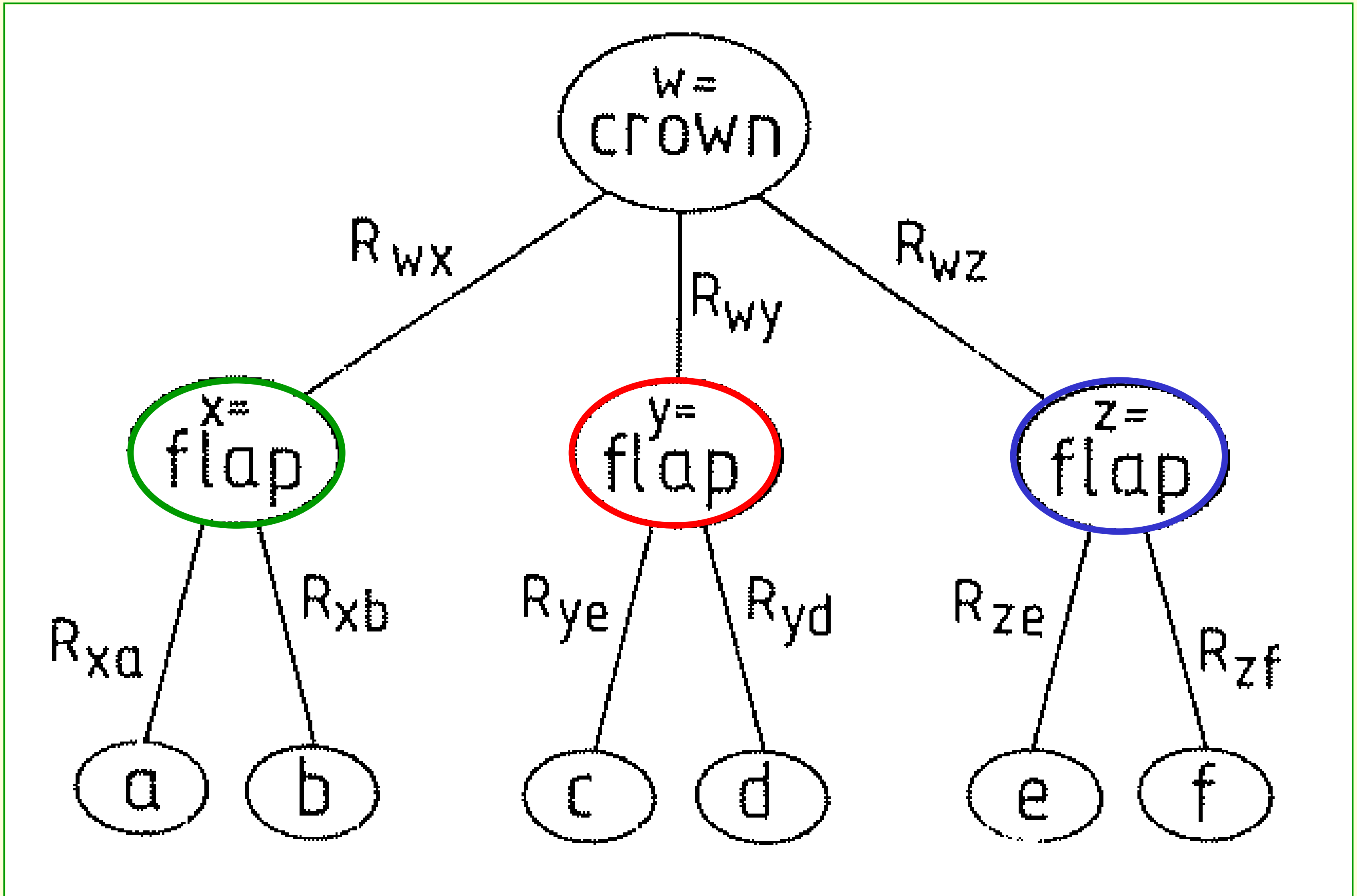
A different percept of the 6 rods



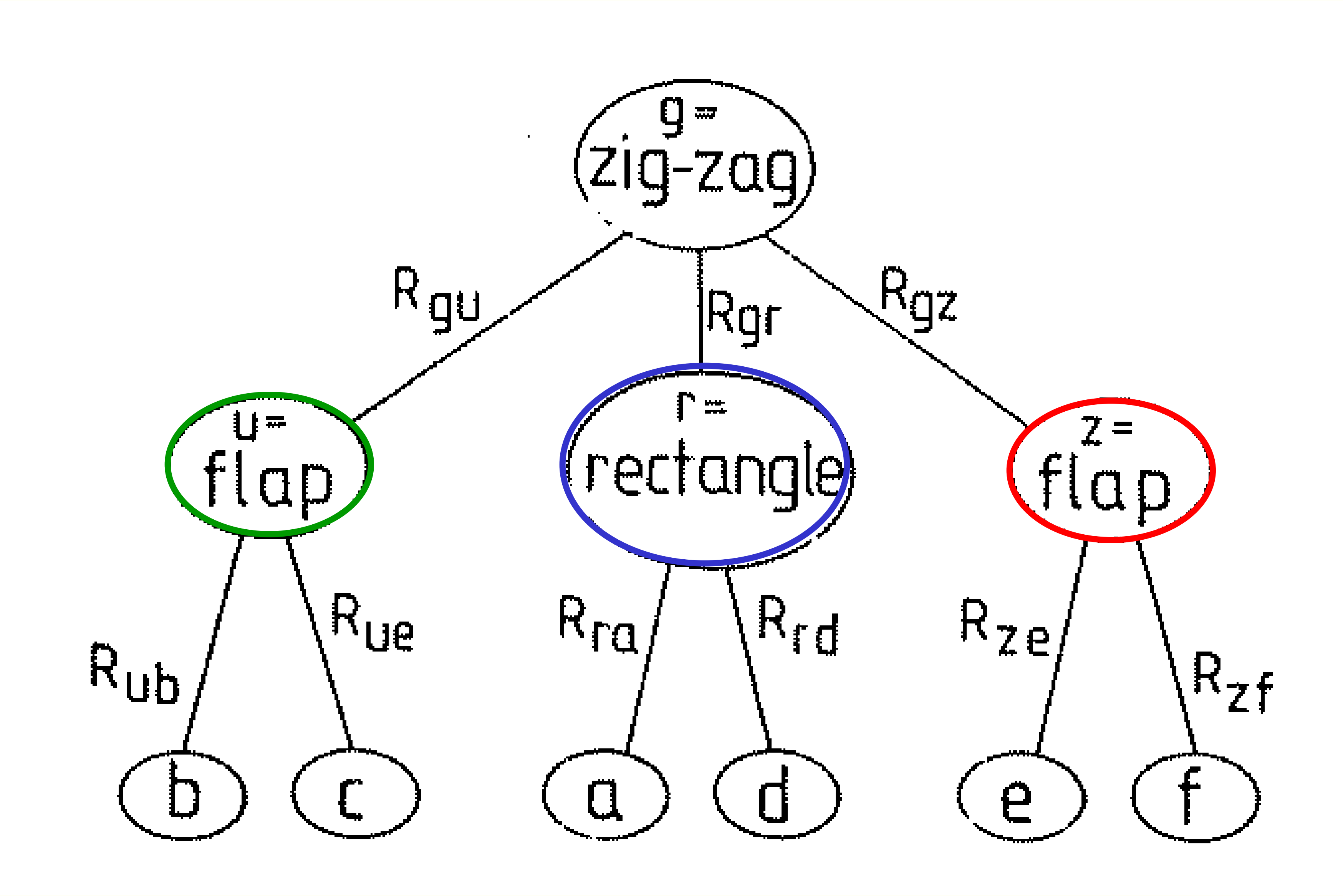
Alternative representations

- 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 align with the intrinsic frame of any of the parts.

A structural description of the "crown" formed by the six rods

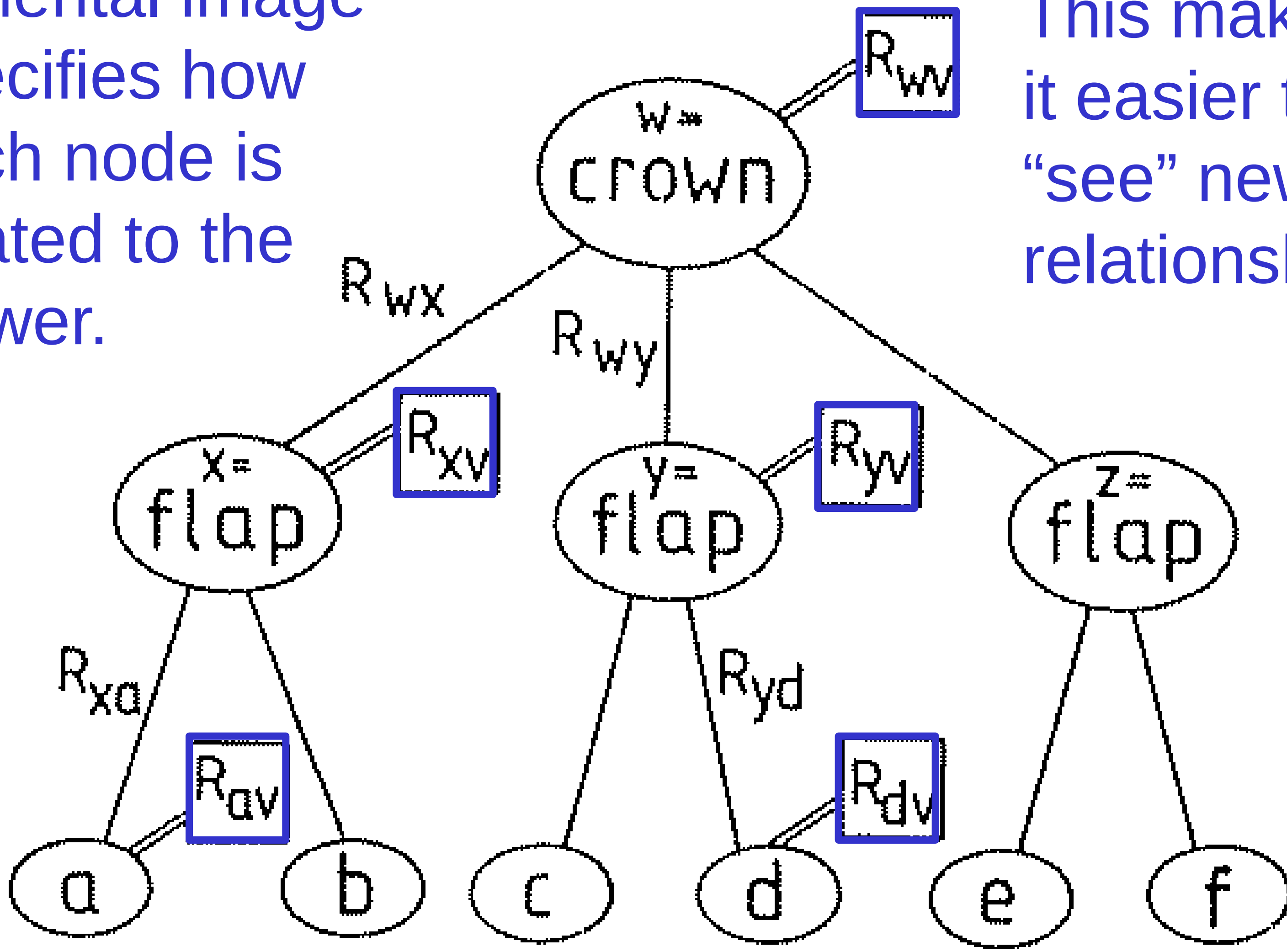


A structural description of the “zig-zag”



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

A psychological theory of the right representation of images

- The representation should be a tree-structured structural description.
  - Knowledge of the viewpoint-invariant relationship between a part and a whole should be stored as a weight matrix.
  - Knowledge of the varying relationship of each node to the viewer should be in the neural activities.
- Mental imagery accesses stored knowledge of spatial relationships by propagating viewpoint information over a structural description.

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

- This is nothing like a structural description.
- It uses multiple layers of convolutional 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.

Why convolutional neural networks are doomed

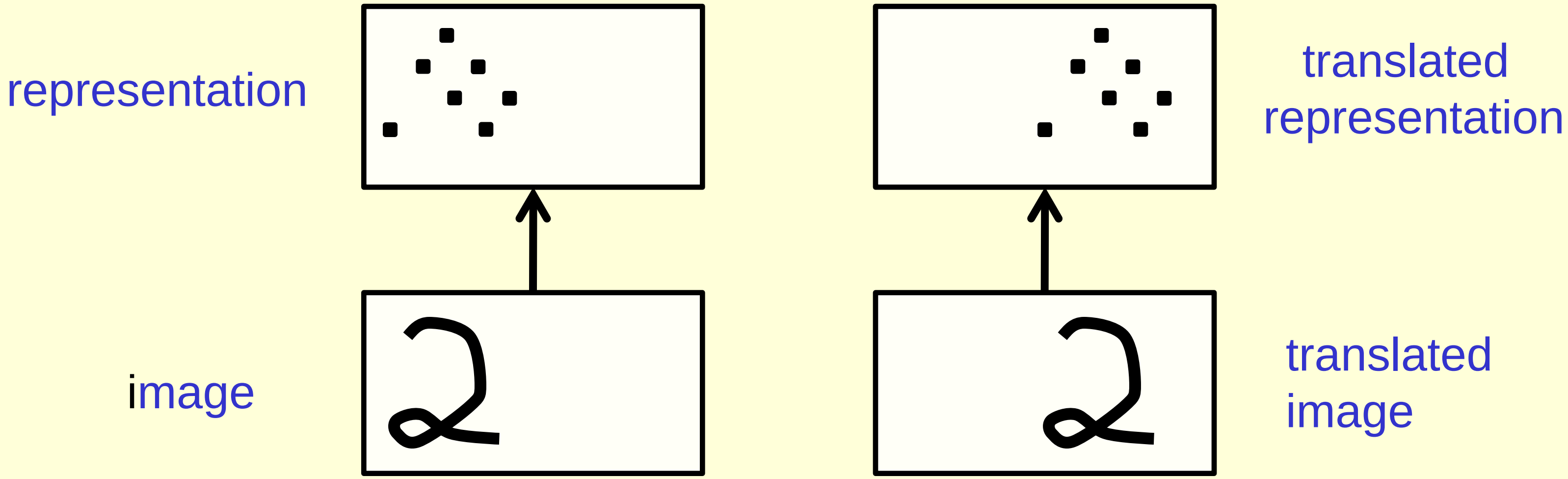
- This architecture is doomed because the sub-sampling loses the precise spatial relationships between higher-level parts such as a nose and a mouth.
  - The precise spatial relationships are needed for recognizing whose face it is..

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.

Equivariance

- Without the sub-sampling, convolutional neural nets give "place-coded" equivariance for discrete translations.



- A small amount of translational invariance can be achieved at each layer by using local averaging or maxing.

Two types of equivariance

- In “place-coded” equivariance, a discrete change in a property of a visual entity leads to a discrete change in which neurons are used for encoding that visual entity.
  - This is what happens in convolutional nets.
- In “rate-coded” equivariance, a real-valued change in a property of a visual entity leads to a real-valued change in the output of some of the neurons used for coding that visual entity, but there is no change in which neurons are used.
  - Our visual systems may use both types.

A way to achieve rate-coded equivariance for small translations

Use a "capsule" that uses quite a lot of internal computation (using non-linear "recognition units") and encapsulates the results of this computation into a low dimensional output:

$$x = \sum_j u_j \Phi_j(\text{image}); \quad y = \sum_j v_j \Phi_j(\text{image});$$

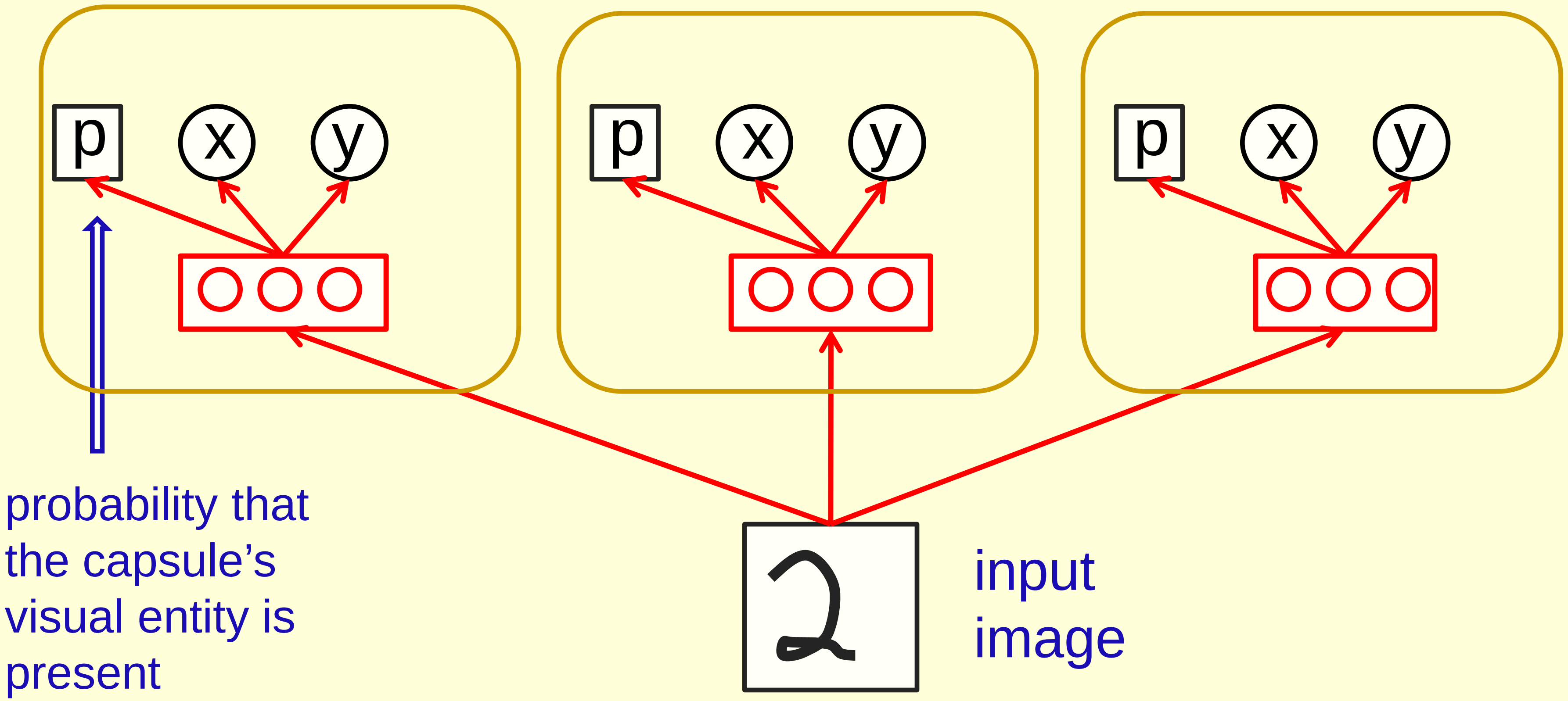
$$p = \sigma\left(\sum_j w_j \Phi_j(\text{image}) \right)$$

probability the visual entity is present

learned weights

learned non-linear recognition units

A picture of three capsules



The real difference between rate-coded equivariance and convolutional nets.

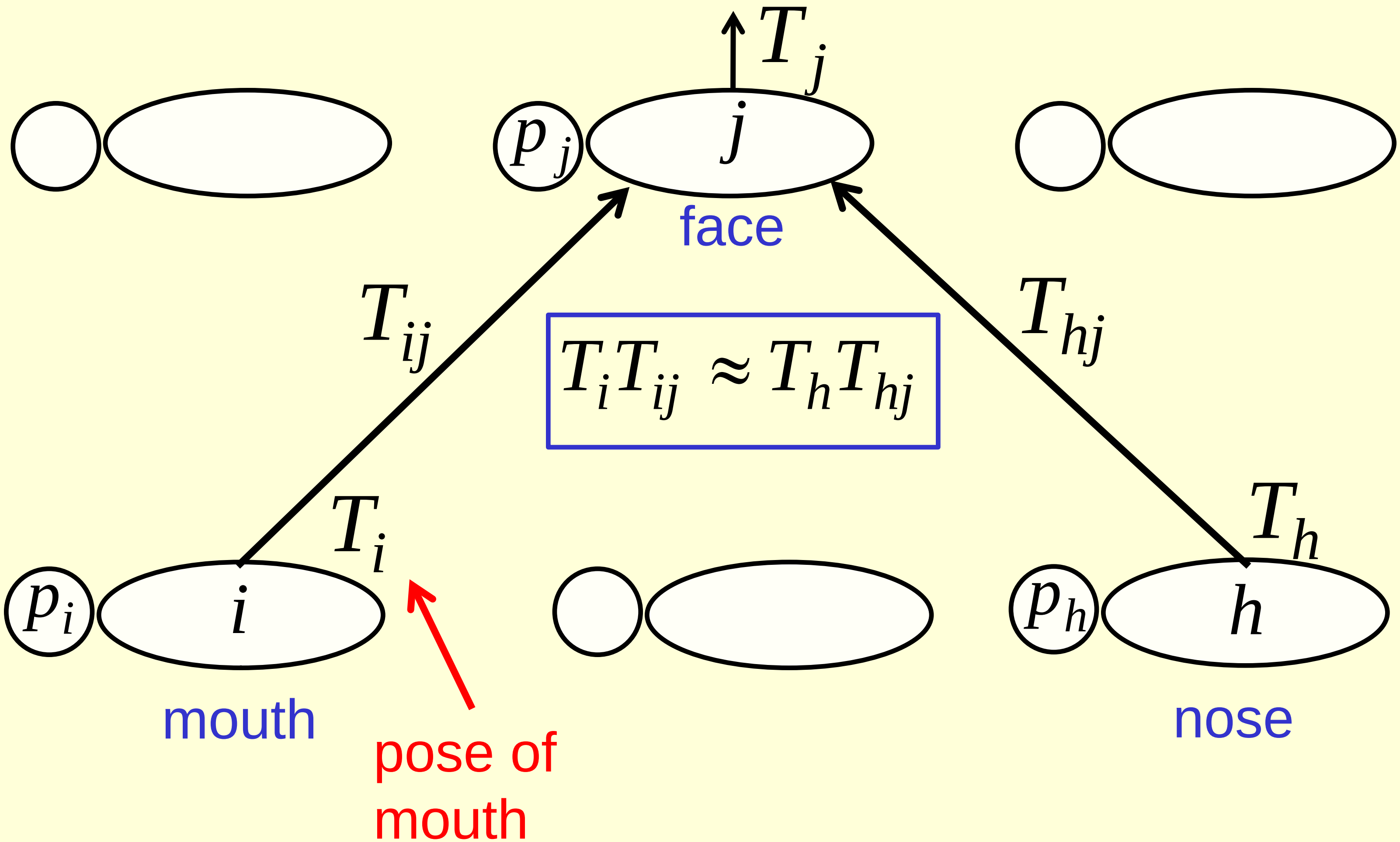
- Sub-sampling compresses the outputs of a pool of convolutional units into the activity level of the most active unit.
  - It may also use the integer location of the winner.
- A capsule encapsulates all of the information provided by the recognition units into two kinds of information:
  - The first is the probability that the visual entity represented by the capsule is present.
  - The second is a set of real-valued outputs that represent the pose of the entity very accurately (and possibly other properties to do with deformation, lighting etc.)

A crucial property of the pose outputs

- 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.

Two layers in a hierarchy of capsules

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



A simple way to learn the lowest level capsules

- 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.

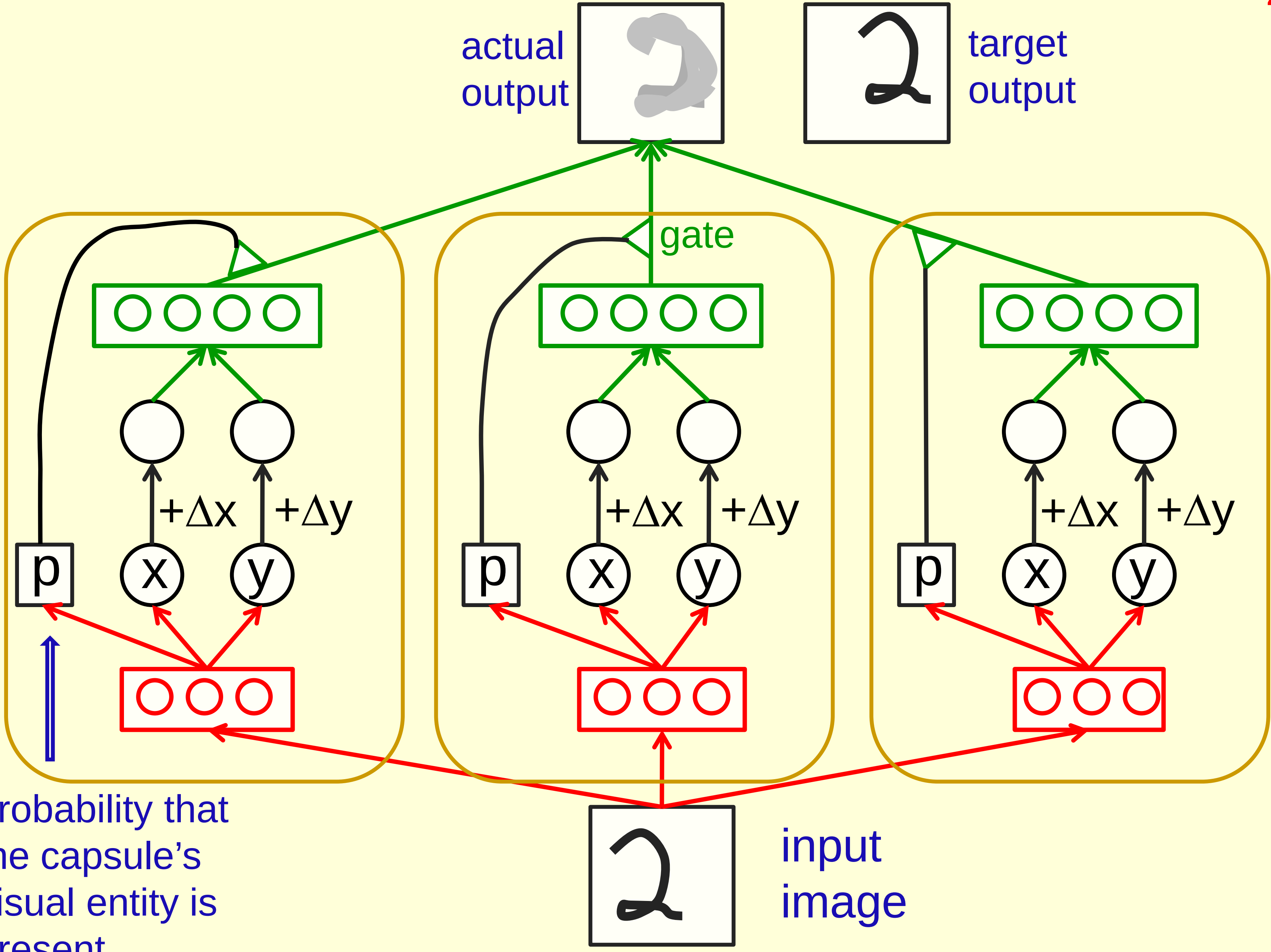
- Each capsule uses its own set of “recognition” hidden units to extract the x and y coordinates of the visual entity it represents (and also the probability of existence)

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.



probability that
the capsule's
visual entity is
present

Why it has to work

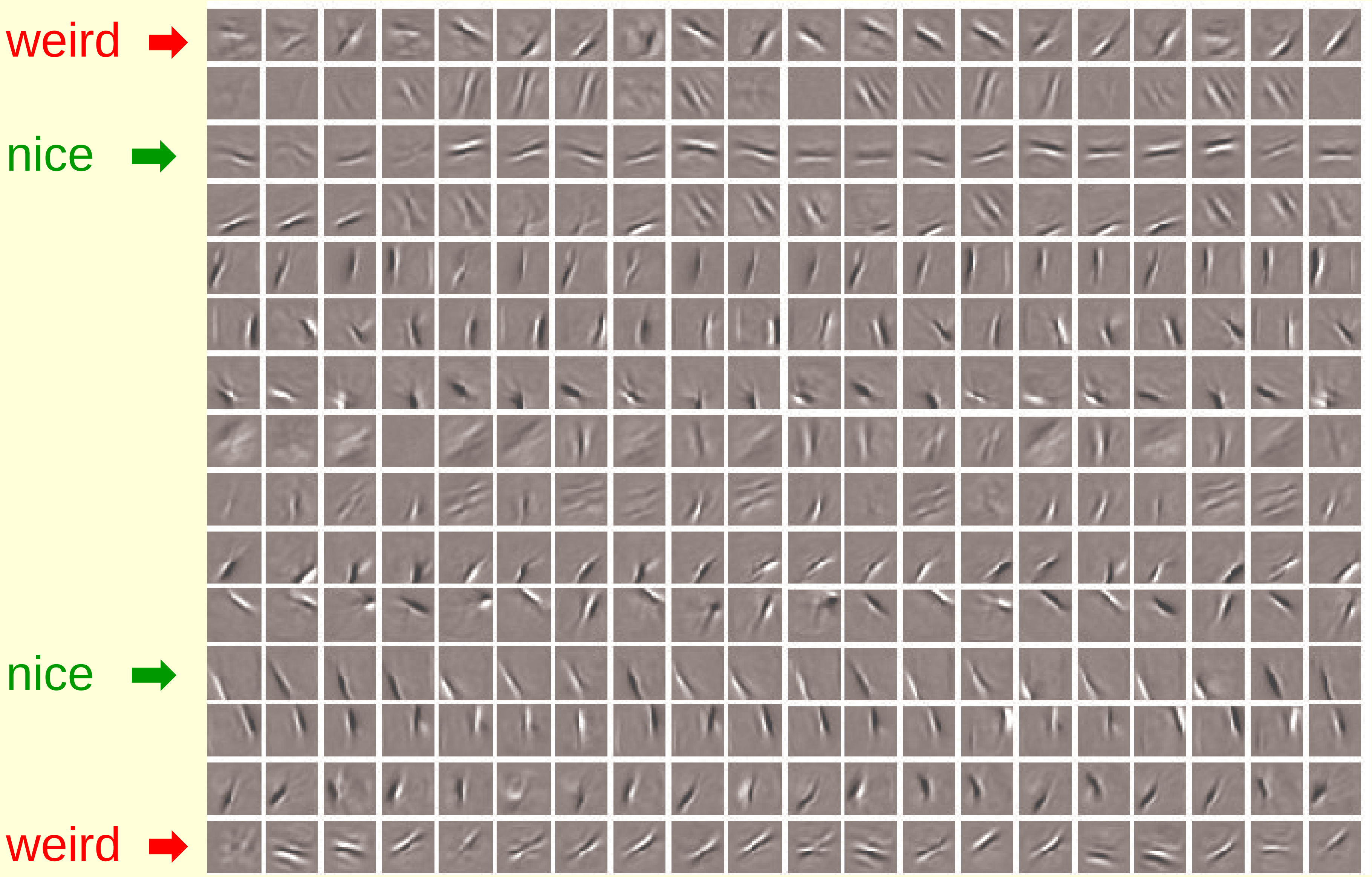
- When the net is trained with back-propagation, the only way it can get the transformations right is by using x and y in a way that is consistent with the way we are using Δx and Δy.

- 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.

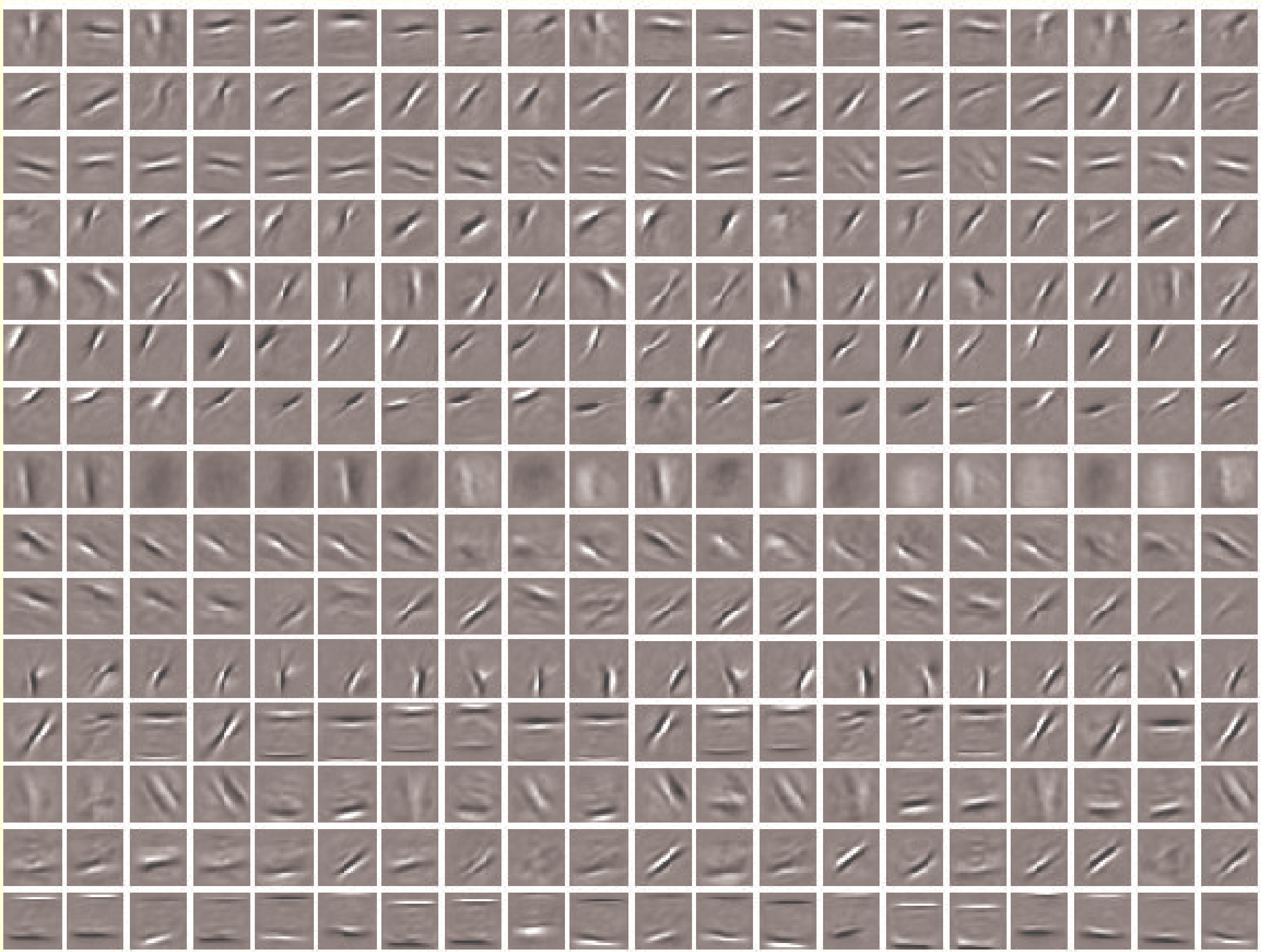
How many capsules do we need?

- Surprisingly few.
  - Each capsule is worth a large number of standard logistic dumb features.
  - 30 capsules is more than enough for representing an MNIST digit image.
- This is very good news for the communication bandwidth that is required to higher levels of analysis.
  - Encapsulation is helpful for parallel distributed computing

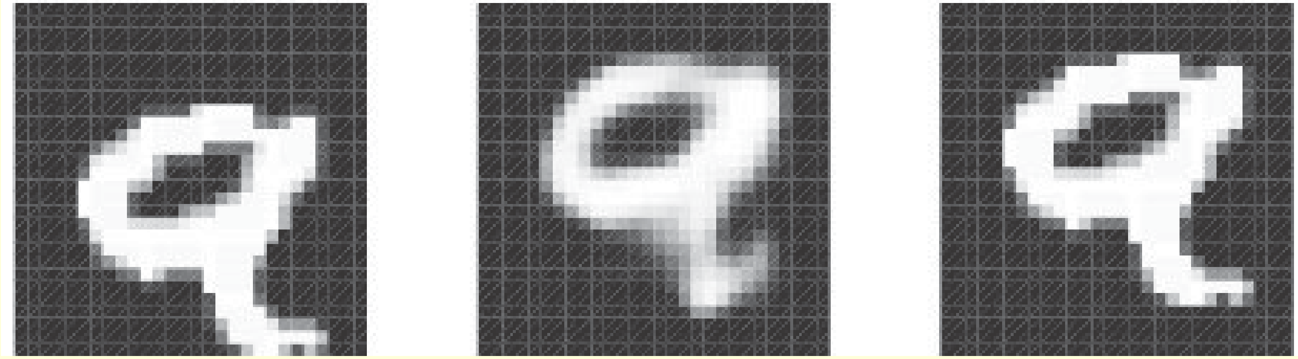
The output fields of the 20 generative hidden units in the first fifteen capsules



The output fields of the 20 generative hidden units in the second fifteen capsules



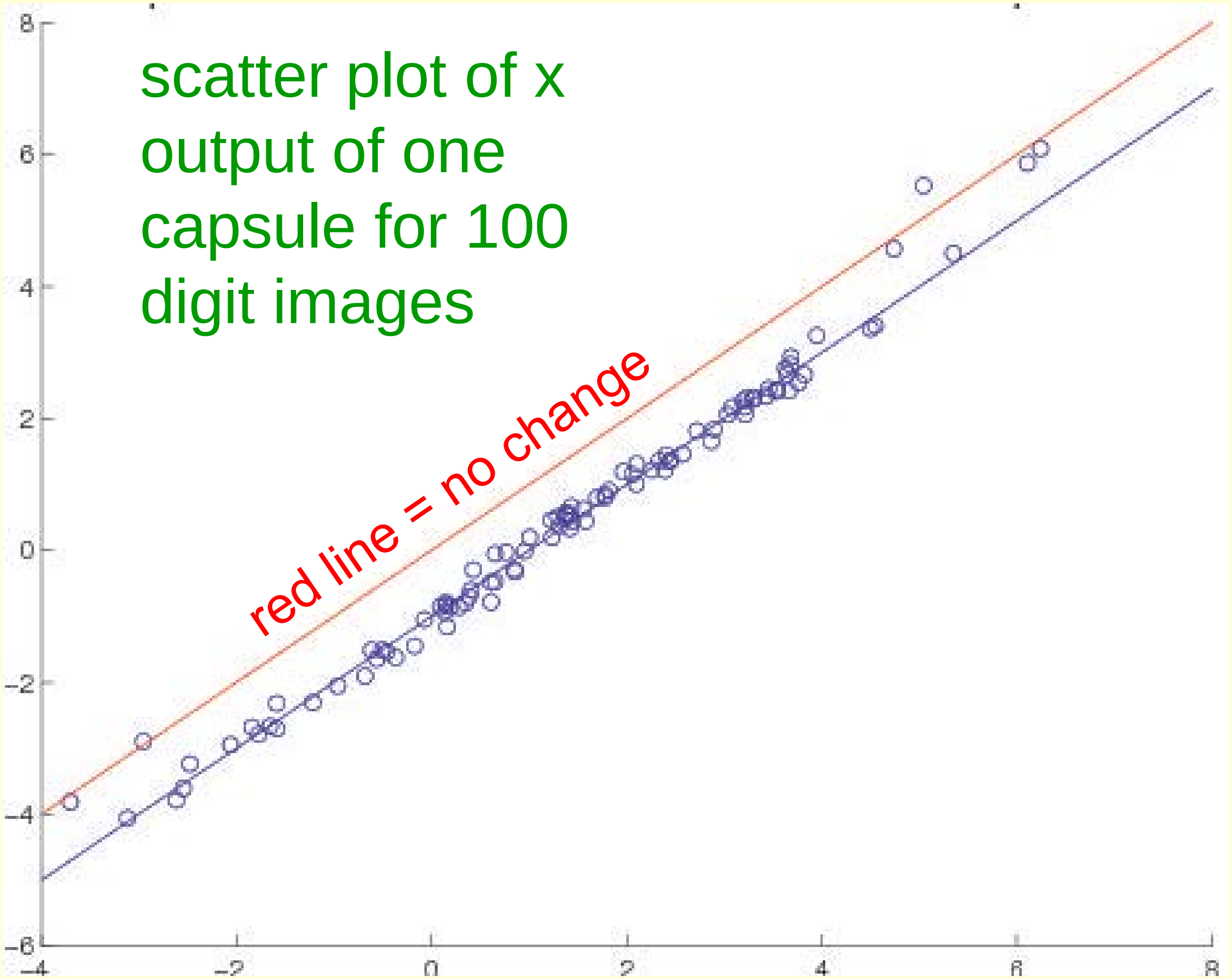
The prediction of the transformed image



input image | predicted image | shifted image

What happens to the coordinates that a capsule outputs when we translate the input image?

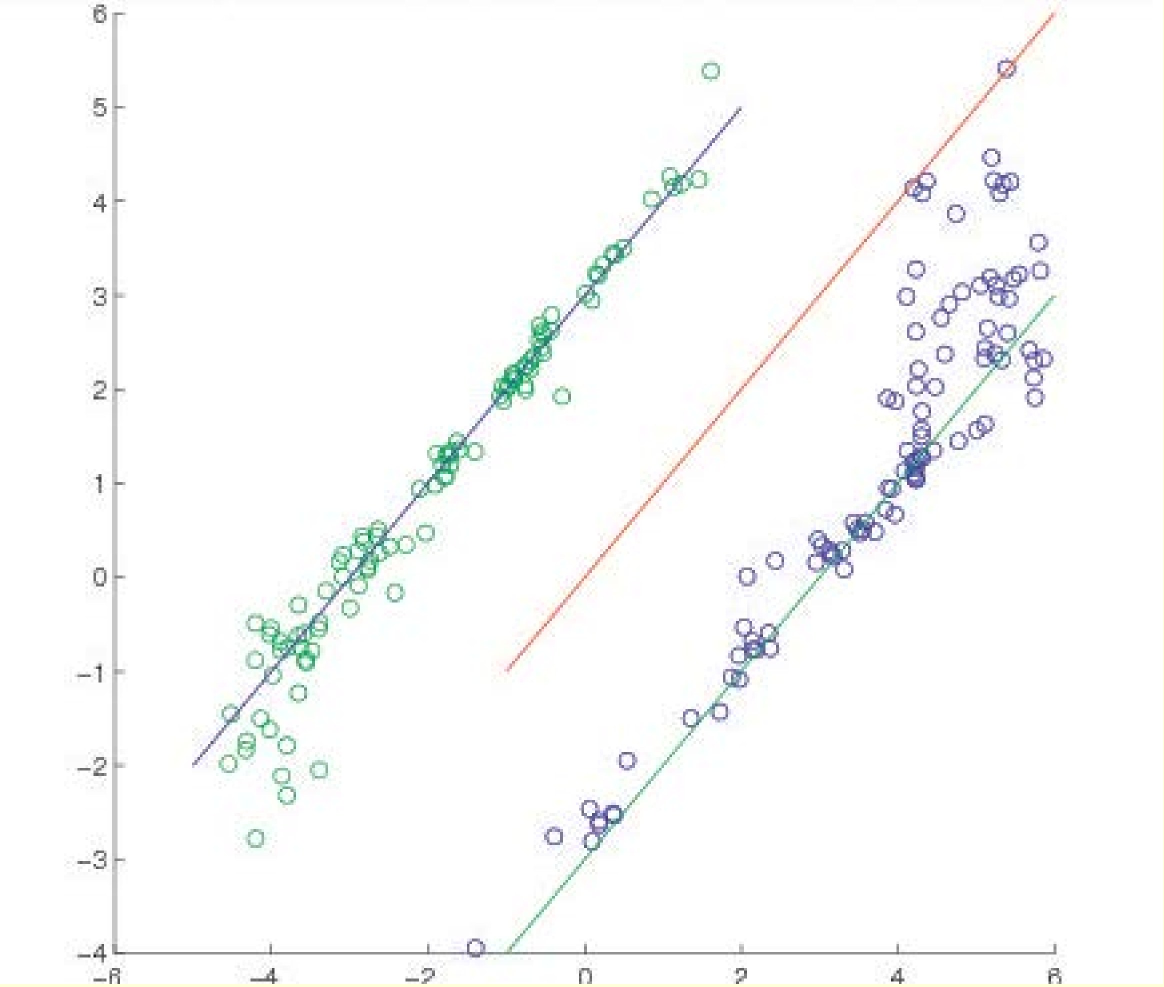
x output before shift



x output after a one pixel shift

What happens to the coordinates that a capsule outputs when we translate the input image?

What happens to the coordinates that a capsule outputs when we translate the input image?

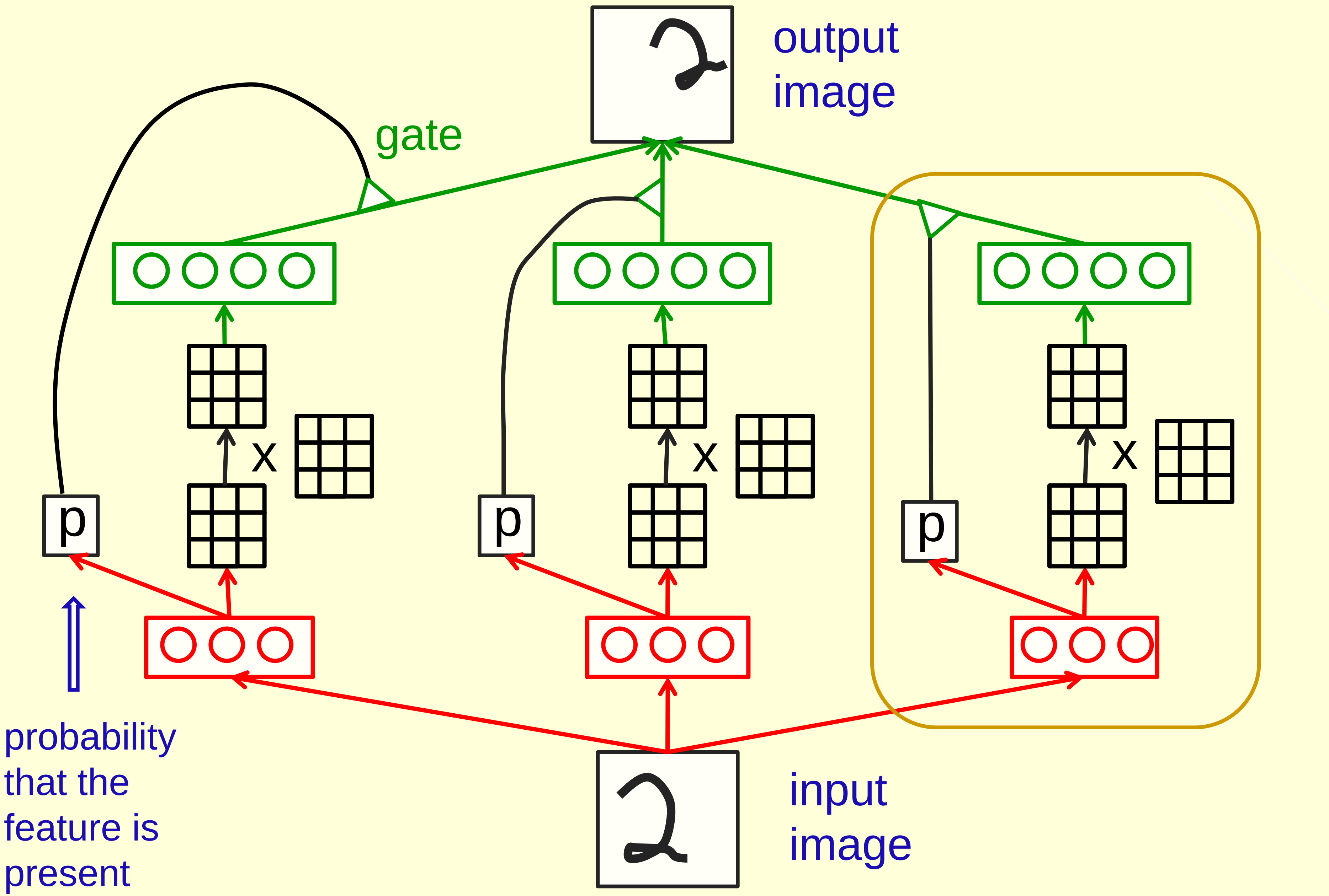


x output before shift ↑

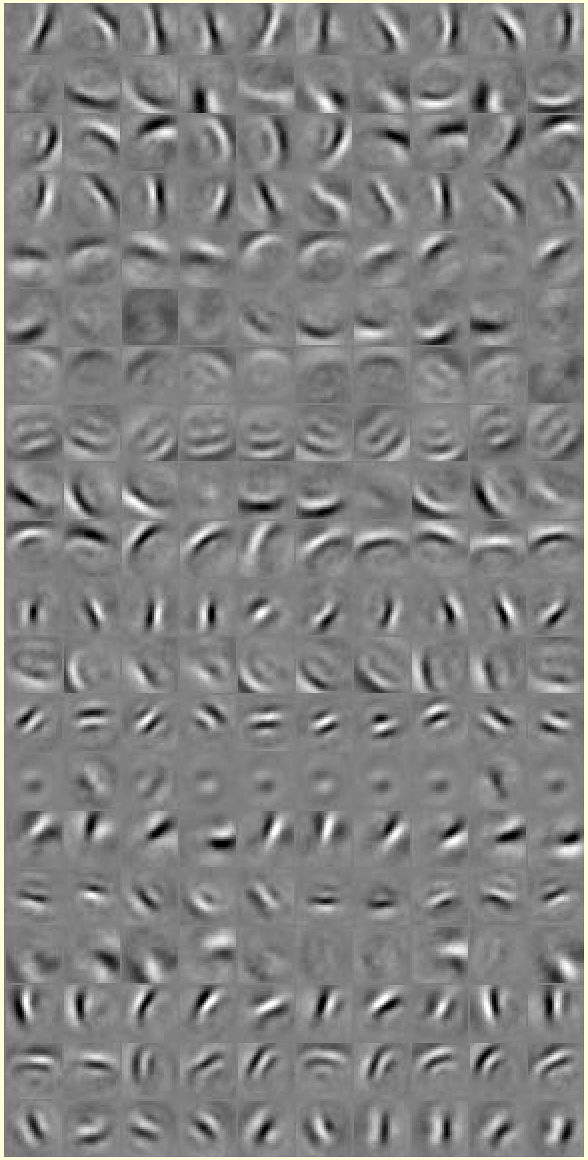
x output after shifts of +3 and -3 pixels ⟶

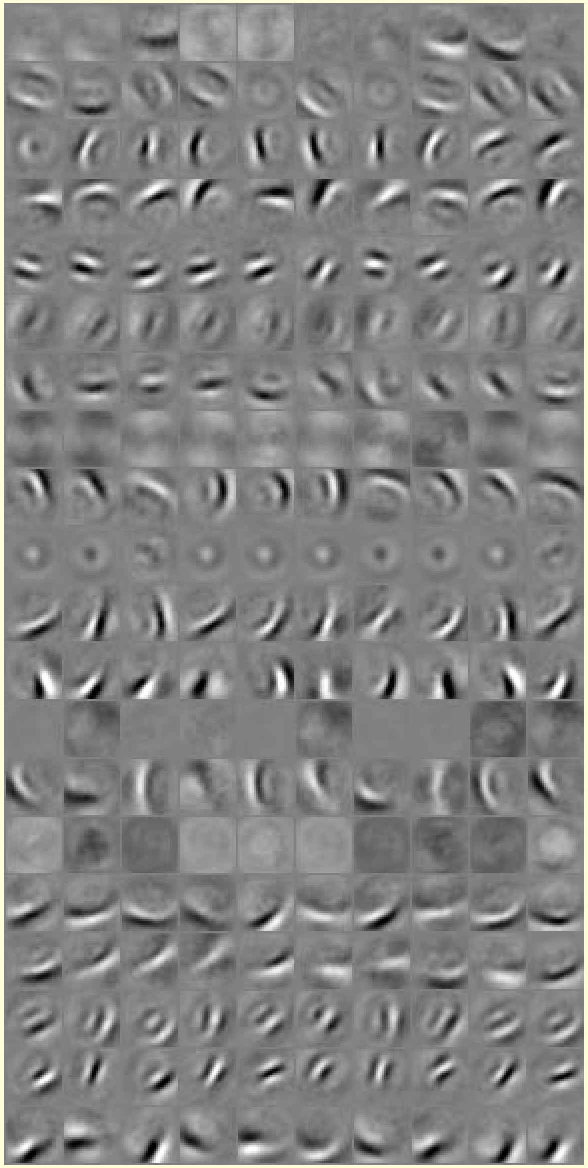
Dealing with scale and orientation (Sida Wang)

- It is easy to extend the network to deal with many more degrees of freedom.
  - Unlike a convolutional net, we do not have to grid the space with replicated filters (which is infeasible for more than a few dimensions).
- The non-linear recognition units of a capsule can be used to compute the elements of a full coordinate transformation.
  - This achieves full equivariance: As the viewpoint changes the representation changes appropriately.
- Rushing to achieve invariance is a big mistake. It makes it impossible to compute precise spatial relationships between high-level features such as noses and mouths.


output
image
gate
x
x
x
p
p
p
probability
that the
feature is
present
input
image

Reconstruction filters of 40 capsules learned on MNIST with full affine transformations (Sida Wang)





Relationship to a Kalman filter

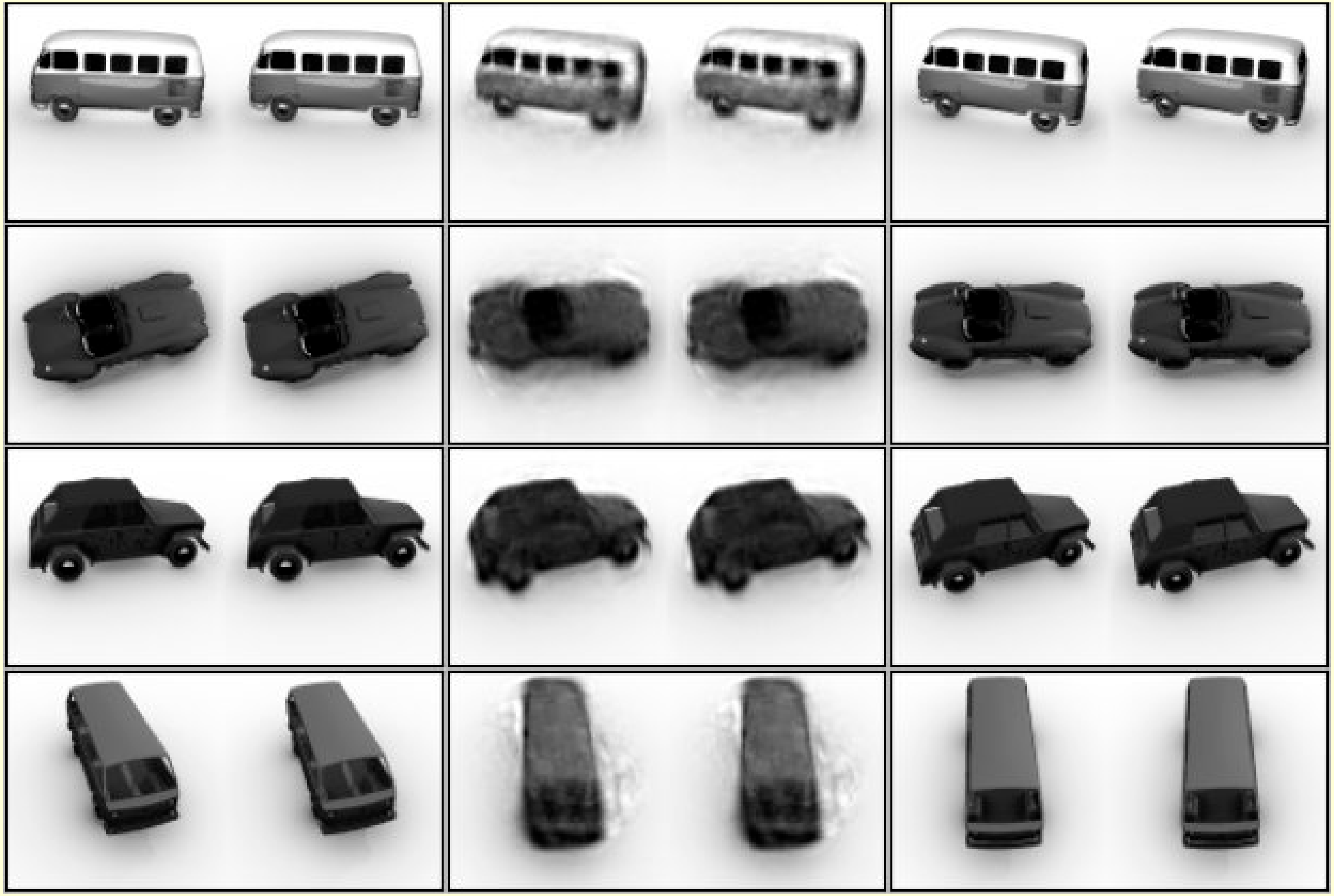
- A linear dynamical system can predict the next observation vector.
  - But only when there is a linear relationship between the underlying dynamics and the observations.
- The extended Kalman filter assumes linearity about the current operating point. It’s a fudge.
- Capsules use non-linear recognition units to map the observation space to the space in which the dynamics is linear. Then they use non-linear generation units to map the prediction back to observation space
  - This is a much better approach than the extended Kalman filter, especially when the dynamics is known.

Dealing with the three–dimensional world

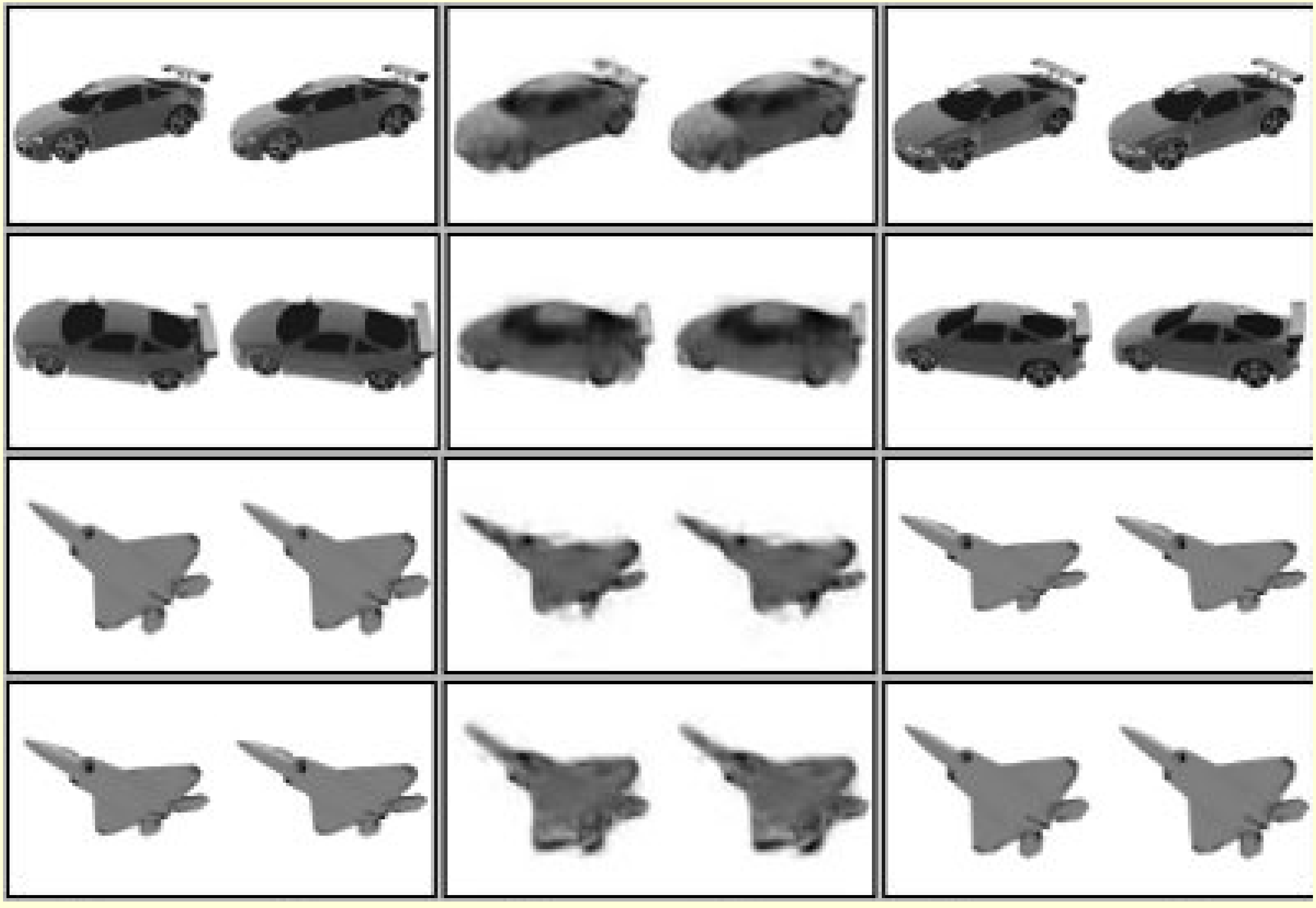
- Use stereo images and make the matrices 4x4
  - Using capsules, 3-D would not be much harder than 2-D if we started with 3-D pixels.
  - The loss of the depth coordinate is a separate problem from the complexity of 3-D geometry.

- At least capsules stand a chance of dealing with the 3-D geometry properly.

An initial attempt to deal with 3-D viewpoint properly (Alex Krizhevsky)



It even works on test data



Hierarchies of capsules

- The first level of capsules converts pixel intensities to the poses of visual entities.
  - It does image de-rendering.
  - It can also extract instantiation parameters for lighting direction, intensity, contrast *etc.*
- Higher levels can use the poses of parts to predict the poses of wholes.
  - Higher-level capsules can also be trained using pairs of transformed images.
  - If they use bigger transformations they can learn to stitch lower-level capsules together.

Relationship to the cortical "what" pathway

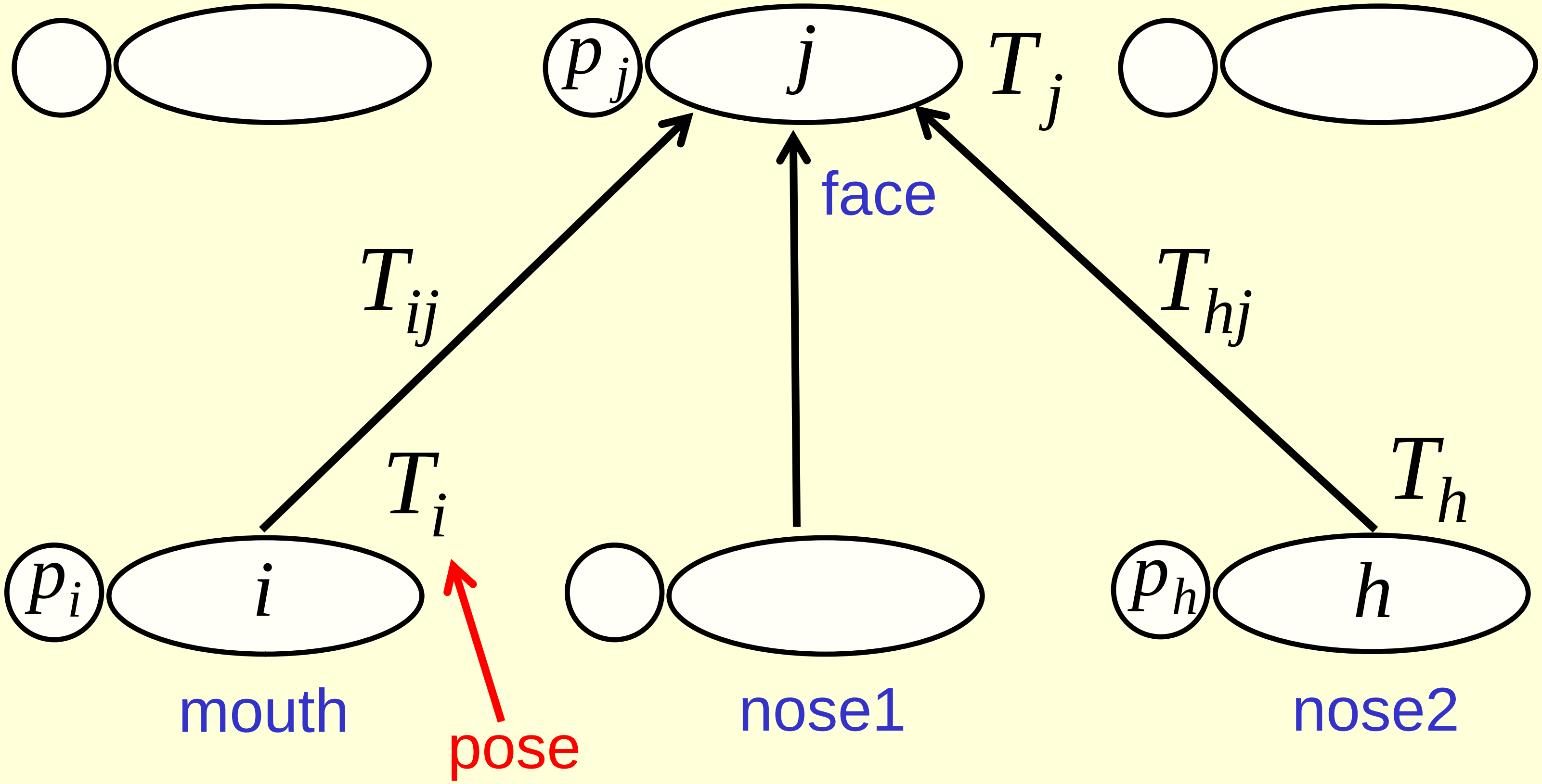
- As we ascend the pathway, the domains get bigger and the visual entities get more complex and rarer.
  – This does not mean that higher-level capsules have lost precise pose information -- a "what" is determined by the relative "wheres" of its parts.

- A capsule could be implemented by a cortical column.
  – It has a lot of internal computation with relatively little communication to the next cortical area.
  – V1 does de-rendering so it looks different.

the end

How a higher level capsule can have a larger domain than its parts

- A face capsule can be connected to several different nose capsules that have more limited domains.



Relationship to the Hough transform

- Standard Hough Transform: Make a high-dimensional array that divides the space of predicted poses for an object into small bins.
- Capsules: Use the capsules to create bottom-up pose hypotheses for familiar visual entities composed of several simpler visual entities.
  - If pose hypotheses agree accurately, the higher-level visual entity exists because high-dimensional agreements don't happen by chance.
- This is much more efficient than binning the space, especially in 3-D.
  - But we must be able to learn suitable capsules.