CSC2535: 2013
Advanced Machine Learning

Taking Inverse Graphics Seriously

Geoffrey Hinton
Department of Computer Science
University of Toronto

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.

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
    - Overlapping the sub-sampling pools helps a bit.
  - They cannot extrapolate their understanding of geometric relationships to radically new viewpoints.

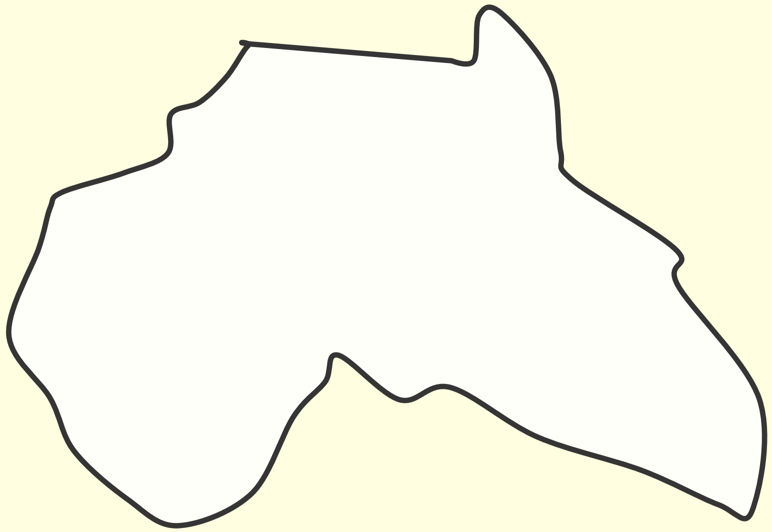
Equivariance vs Invariance

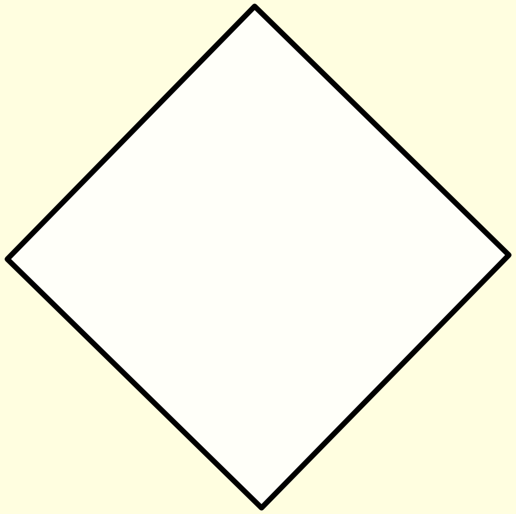
- Sub-sampling tries to make the neural activities invariant to small changes in viewpoint.
  – This is a silly goal, motivated by the fact that the final label needs to be viewpoint-invariant.
- Its better to aim for equivariance: Changes in viewpoint lead to corresponding changes in neural activities.
  – 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.
    - Its just a matrix multiply!

Some psychological evidence that our visual systems impose coordinate frames in order to represent shapes (after Irvin Rock)





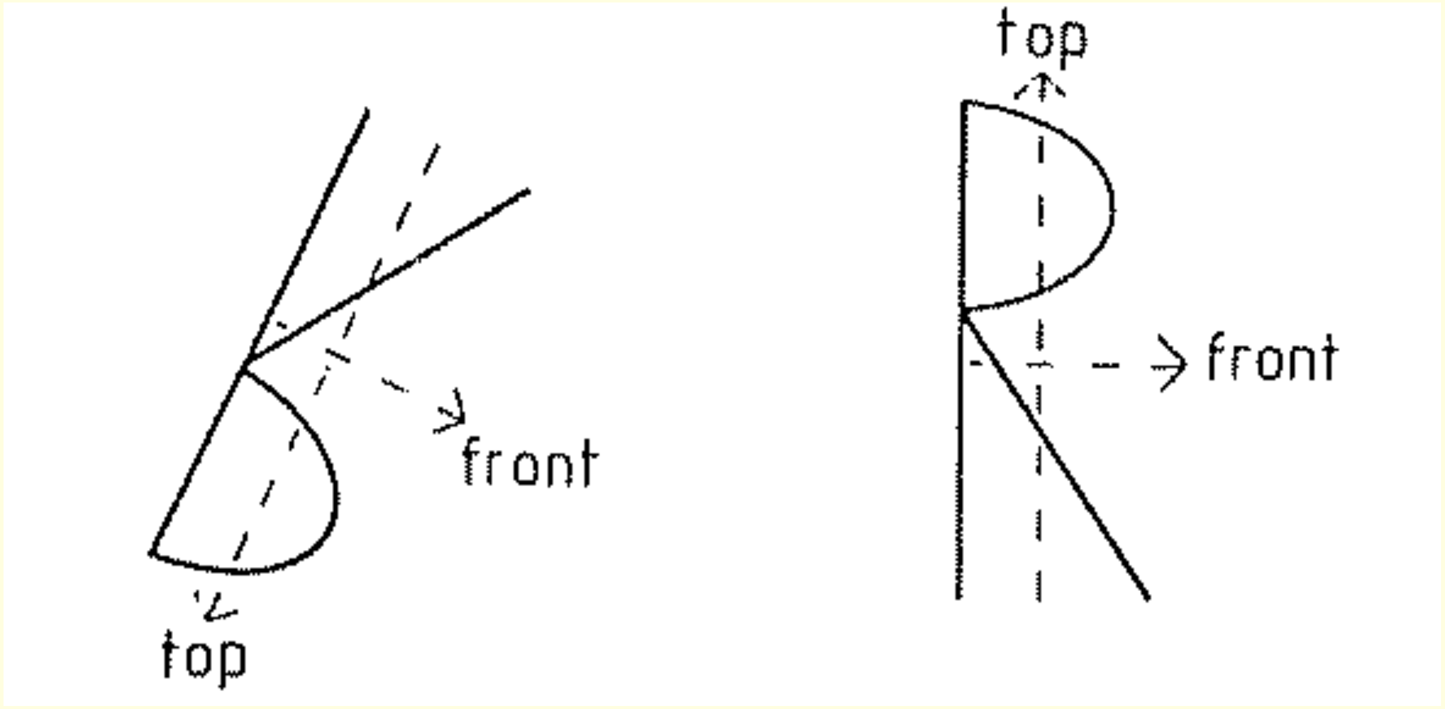
The cube task

- You can all imagine a wire-frame cube.

- Now imagine the cube standing on one corner so that body-diagonal from that corner, through the center of the cube to the opposite corner is vertical.

- Where are the other corners of the cube?
  – You cannot do this task because I forced you to use a coordinate system that is not your usual way of understanding a cube.

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.

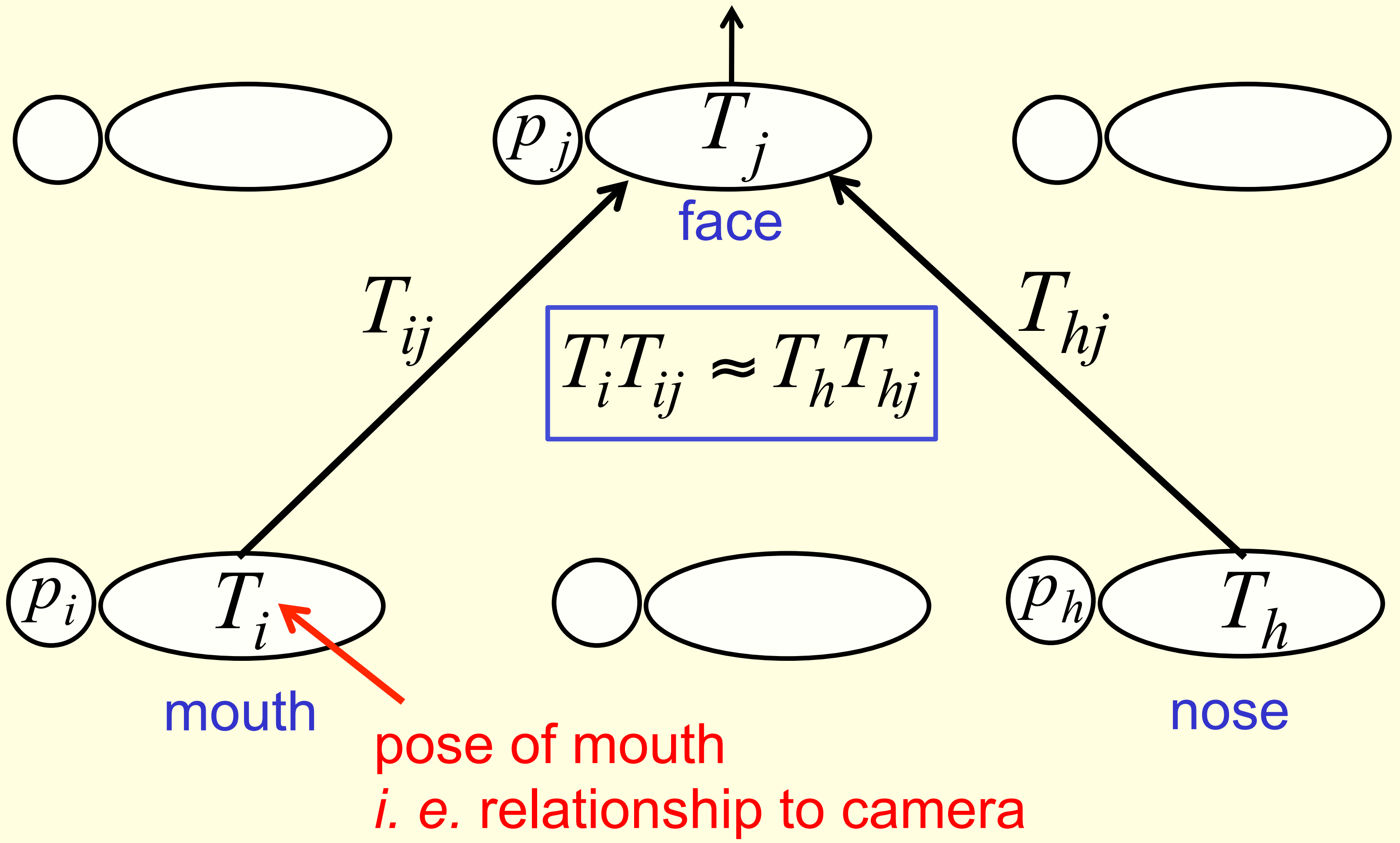
But why do we need to do mental rotation to decide handedness?

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.

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

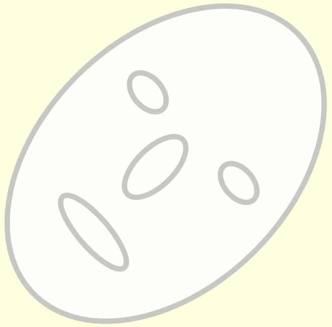
- They allow spatial transformations to be modeled by linear operations.
  - This makes it easy to learn a hierarchy of visual entities.
  - It makes it easy to generalize across viewpoints.
- The invariant geometric properties of a shape are in the weights, not in the activities.
  - The activities are equivariant.

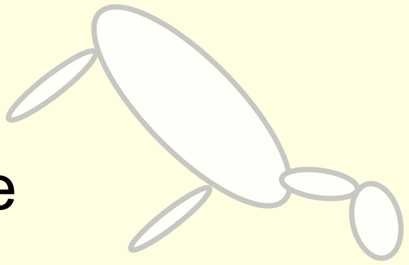
Extrapolating shape recognition to very different sizes, orientations and positions

- Current wisdom:
  - Learn different models for different viewpoints.
  - This requires a lot of training data.
- A much more ambitious approach:
  - The manifold of images of the same rigid shape is highly non-linear in the space of pixel intensities.
  - Transform to a space in which the manifold is globally linear. This allows massive extrapolation.

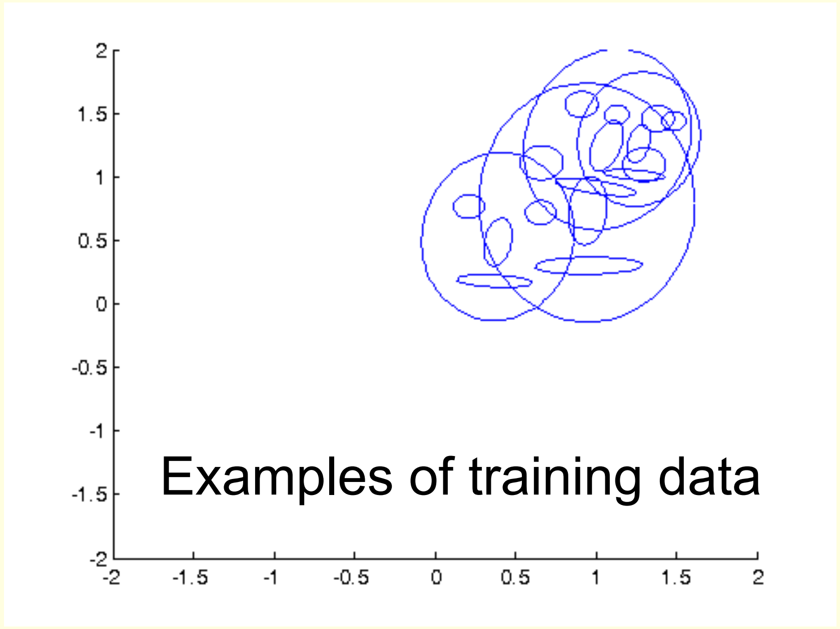
A toy example of what we could do if we could reliably extract the poses of parts of objects

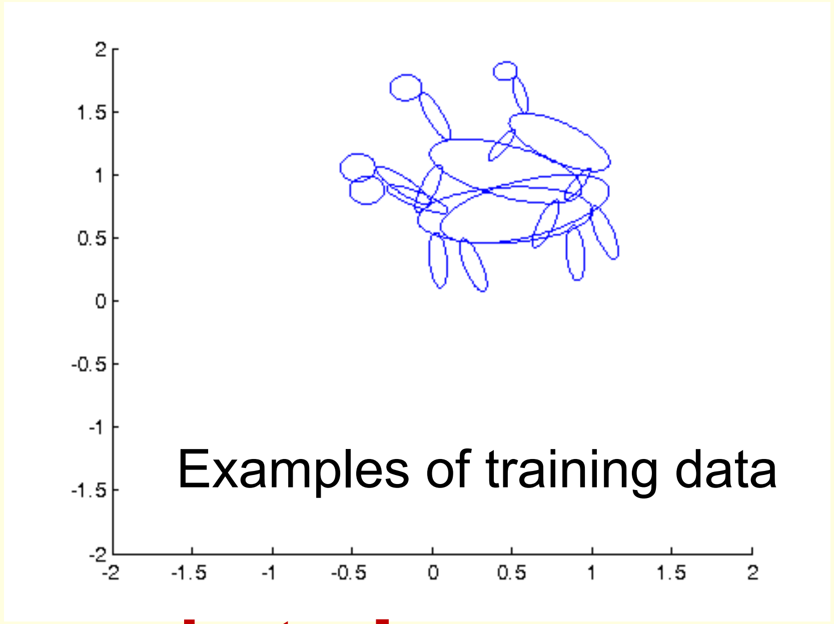
- Consider images composed of five ellipses.
  - The shape is determined entirely by the spatial relations between the ellipses because all parts have the same shape.
- Can we extract sets of spatial relationships from data that contains several different shapes?
- Can we generalize far beyond the range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



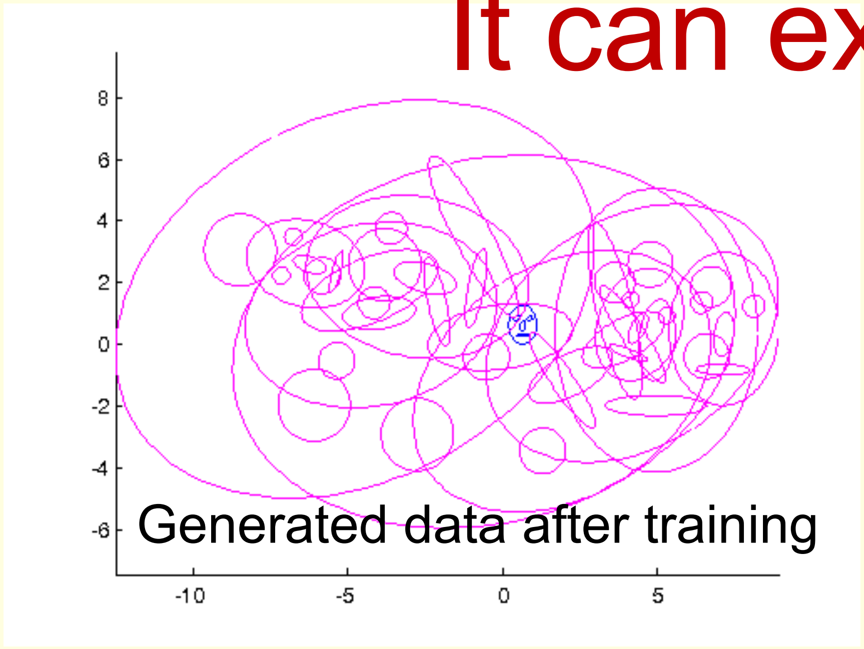


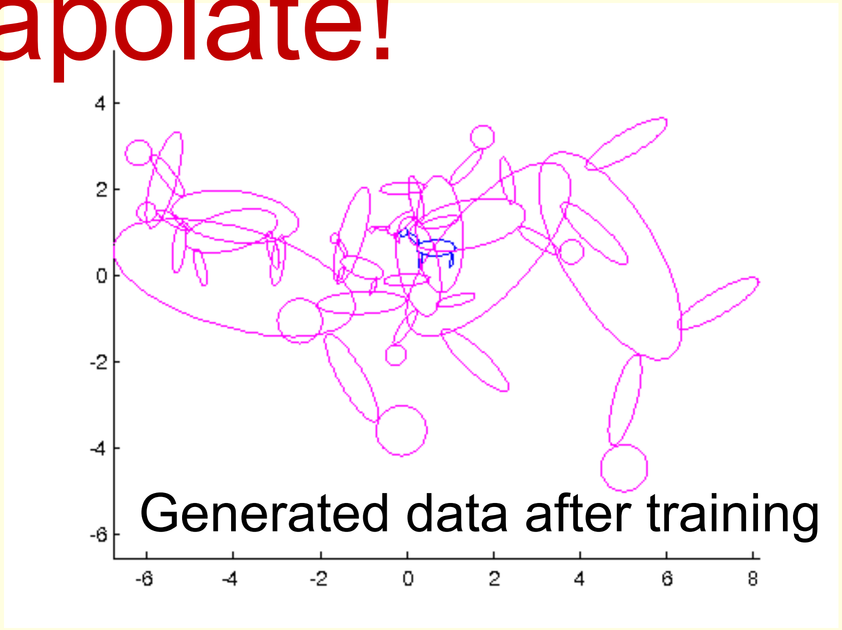
Examples of two shapes (Yichuan Tang)





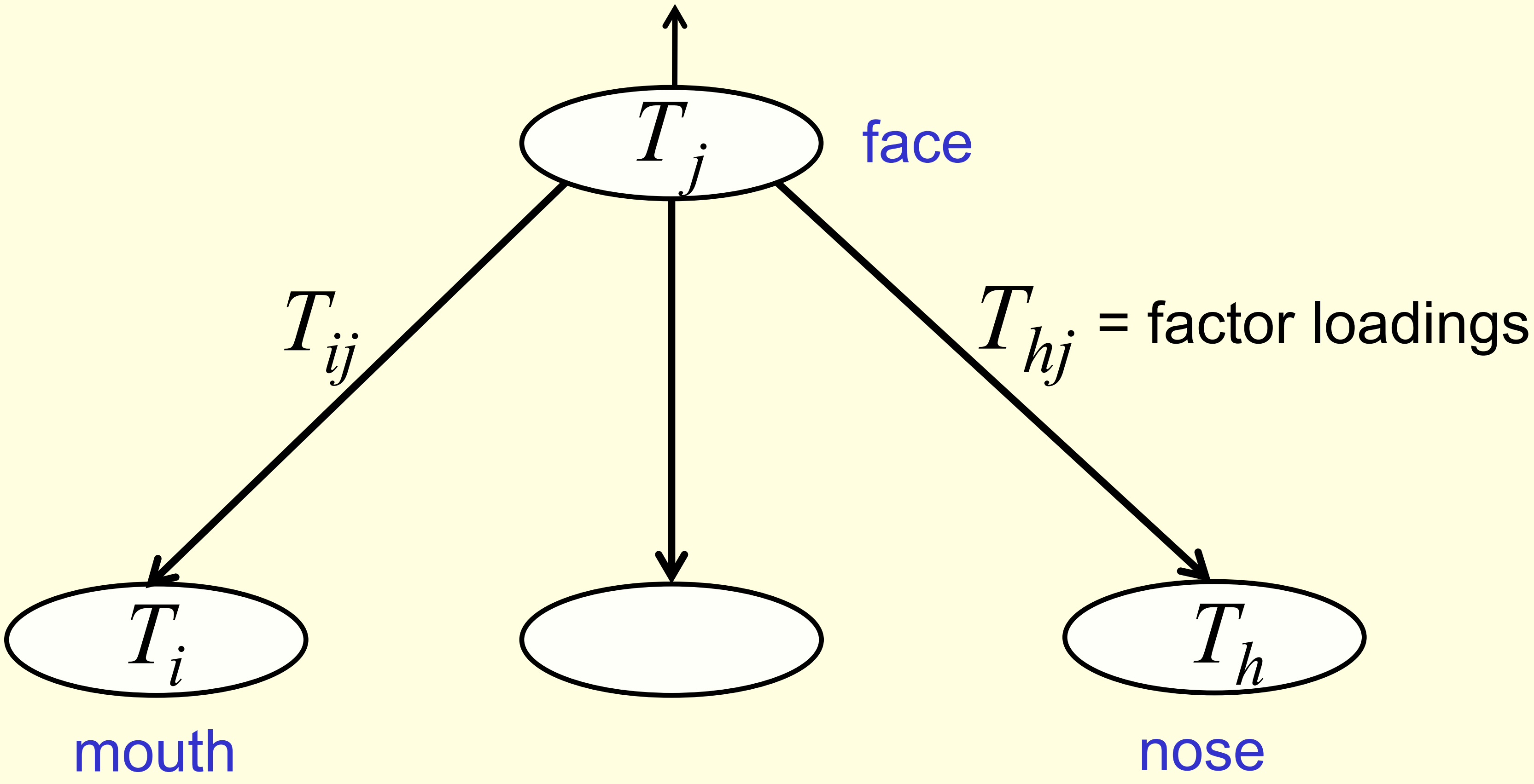
It can extrapolate!





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.



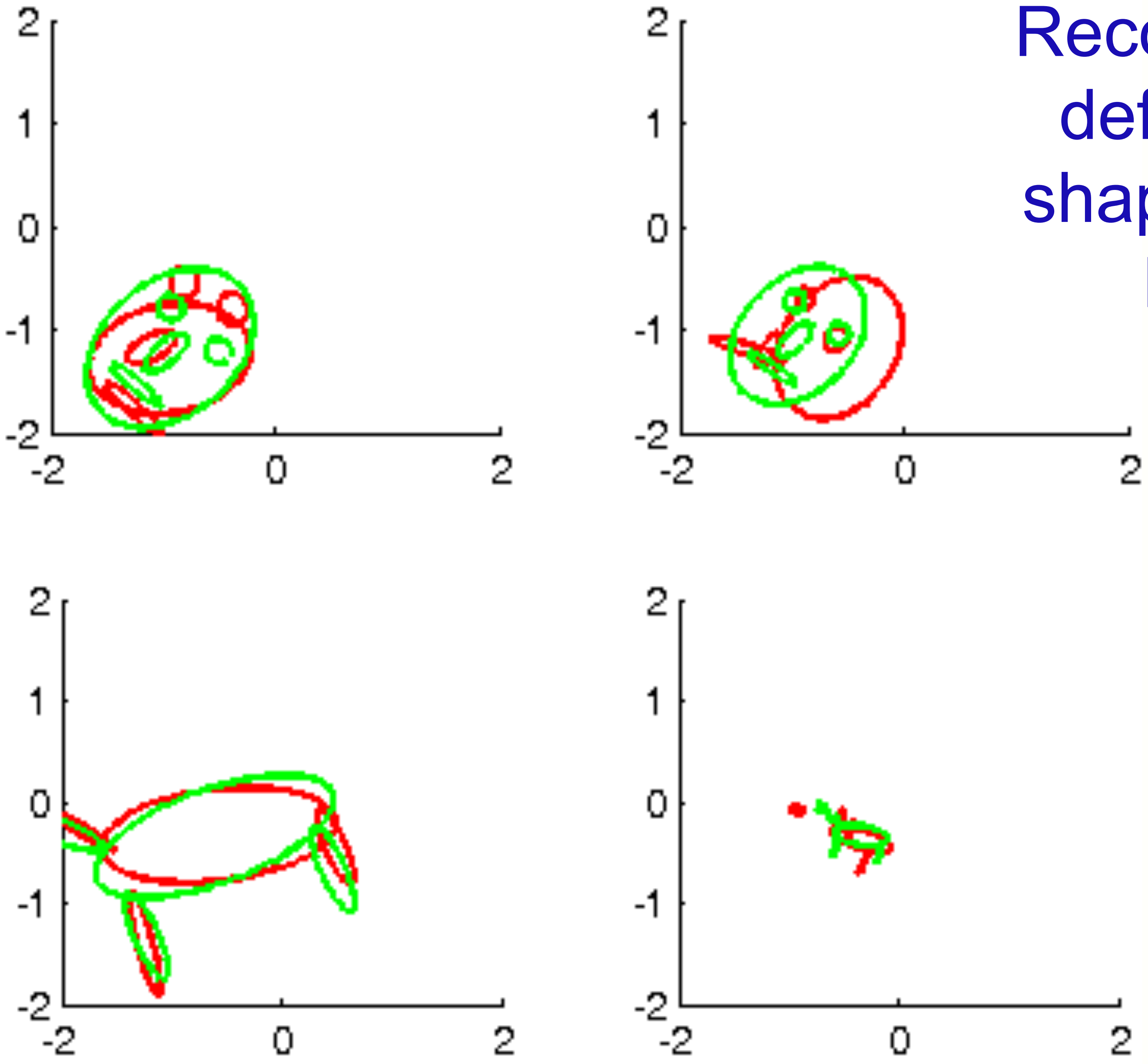
Learning the right assignments of the ellipses to the parts of a shape.

- 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.
- This quickly leads to models that have strong opinions about the correct assignment.
- We could eliminate most of the search by using a feed-forward neural net to predict the assignments.

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.

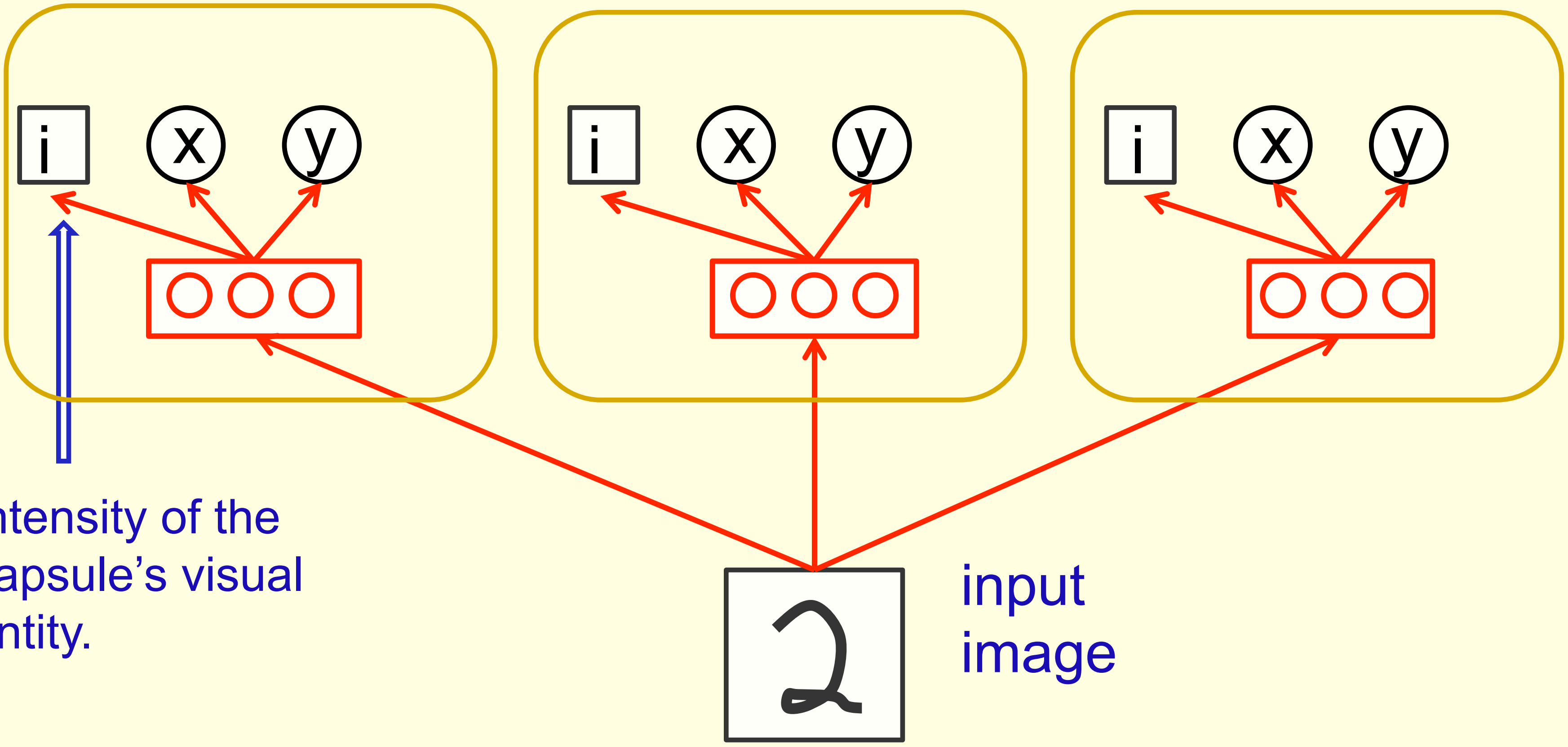
Recognizing deformed shapes with MFA



A big problem

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

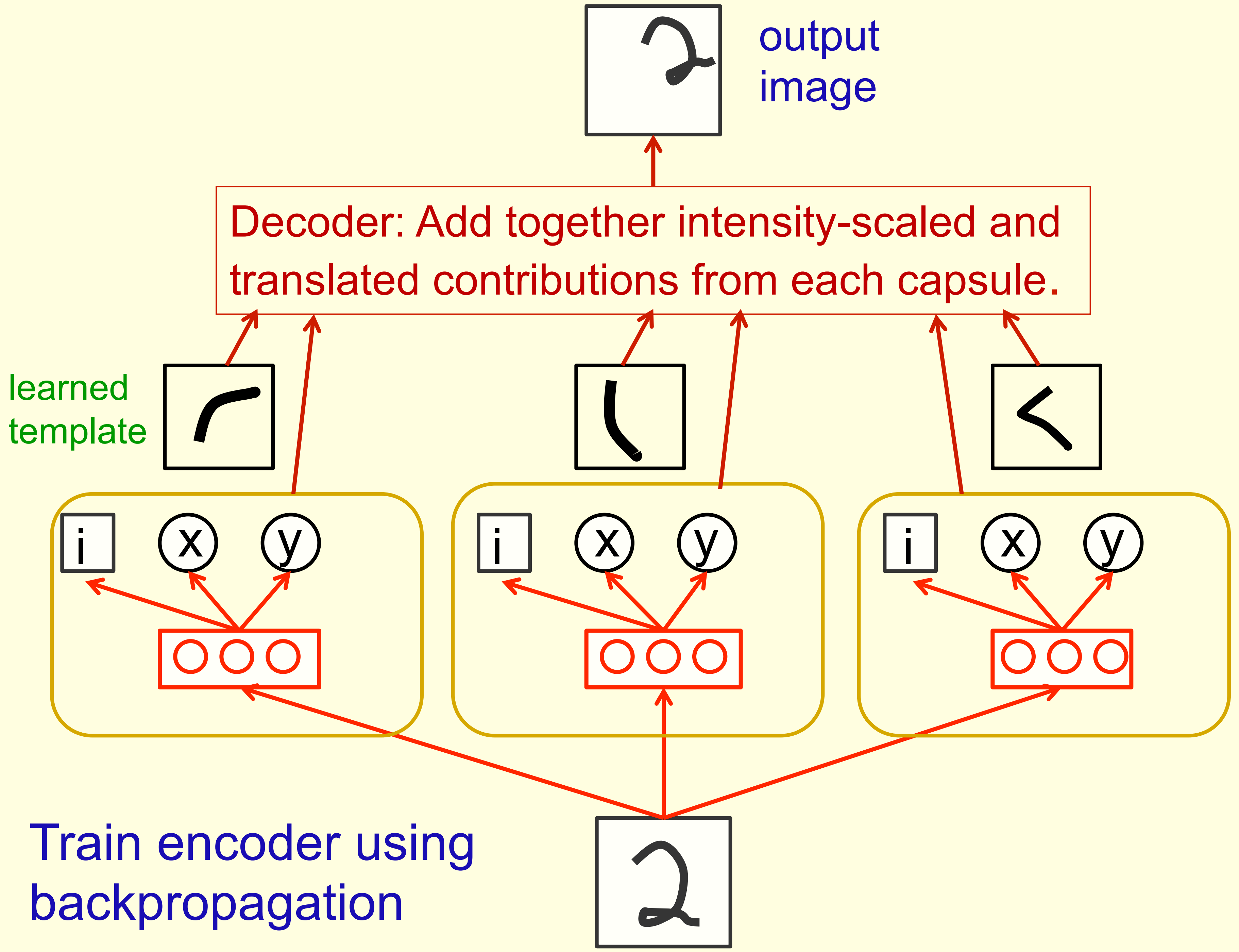
A picture of three capsules



Intensity of the capsule’s visual entity.

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

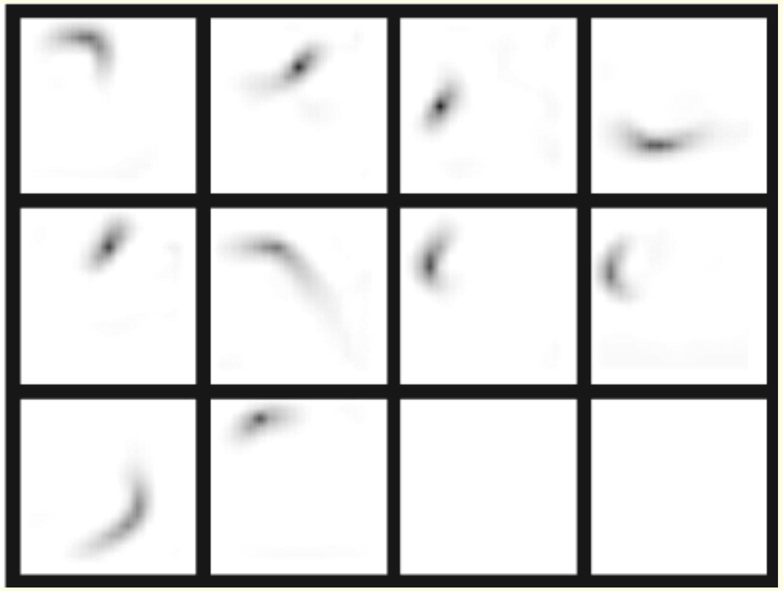
- The idea is to define a simple decoder that produces an image by adding together contributions from each capsule.
- Each capsule learns a fixed “template” that gets intensity-scaled and translated differently for reconstructing each image.
- The encoder must learn to extract the appropriate intensity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



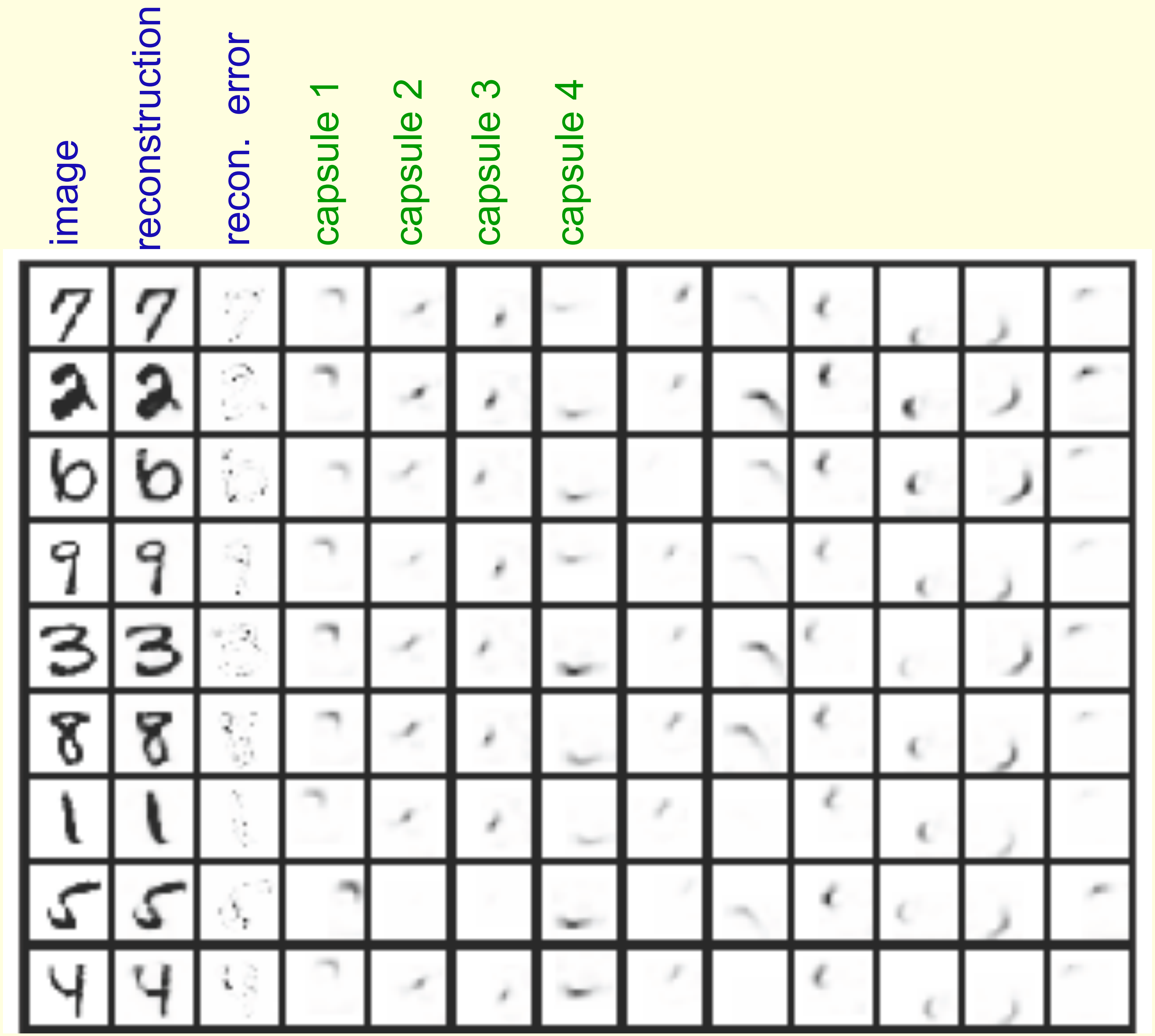
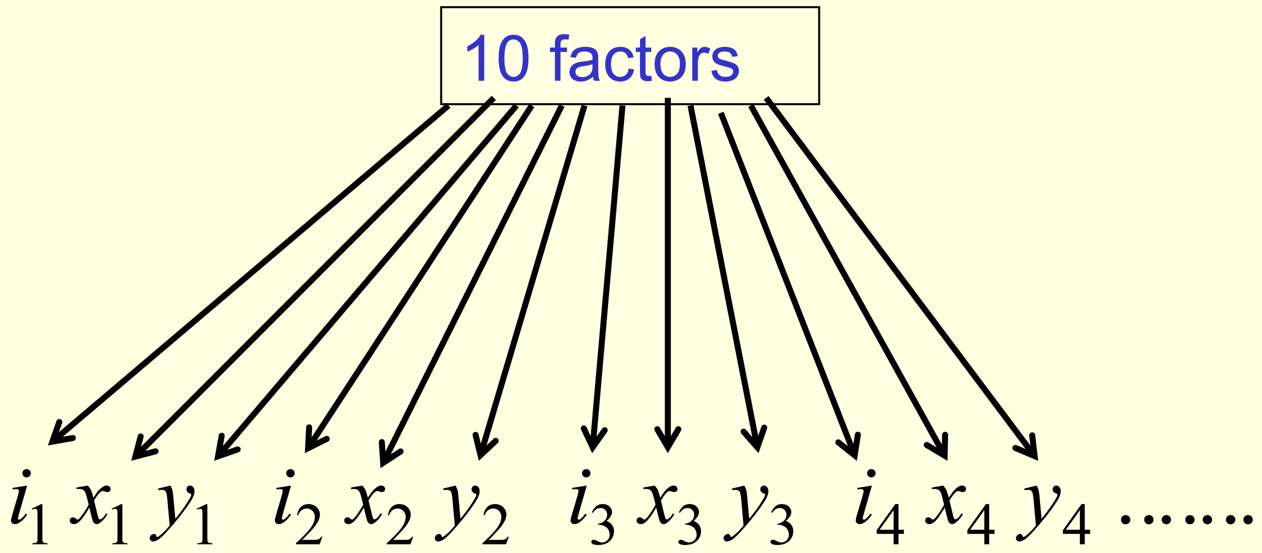


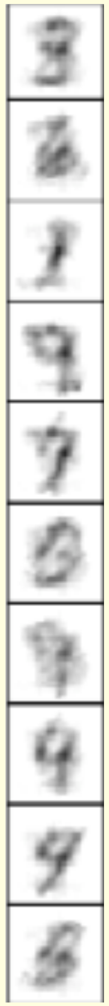
image
reconstruction
recon. error
capsule 1
capsule 2
capsule 3
capsule 4

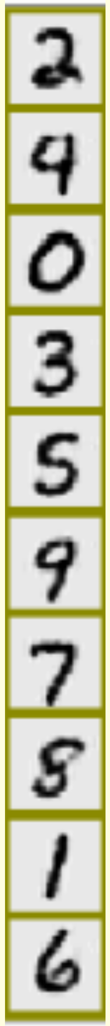
Modeling the capsule outputs with a factor analyser


10 factors
$i_1\ x_1\ y_1\quad i_2\ x_2\ y_2\quad i_3\ x_3\ y_3\quad i_4\ x_4\ y_4$ …….

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

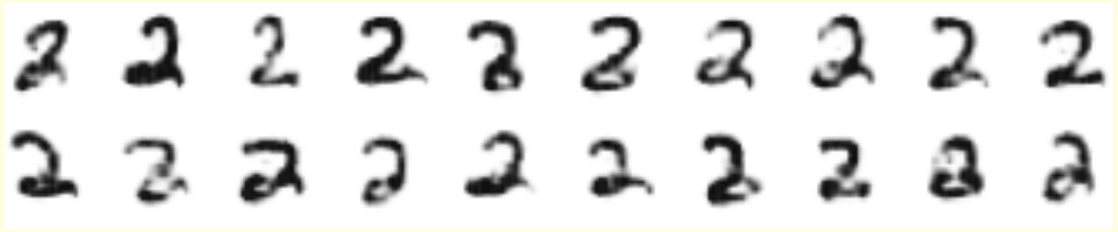




directly on pixels

on outputs of capsules

mean $- 2\sigma$ of each factor



mean $+ 2\sigma$ 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.

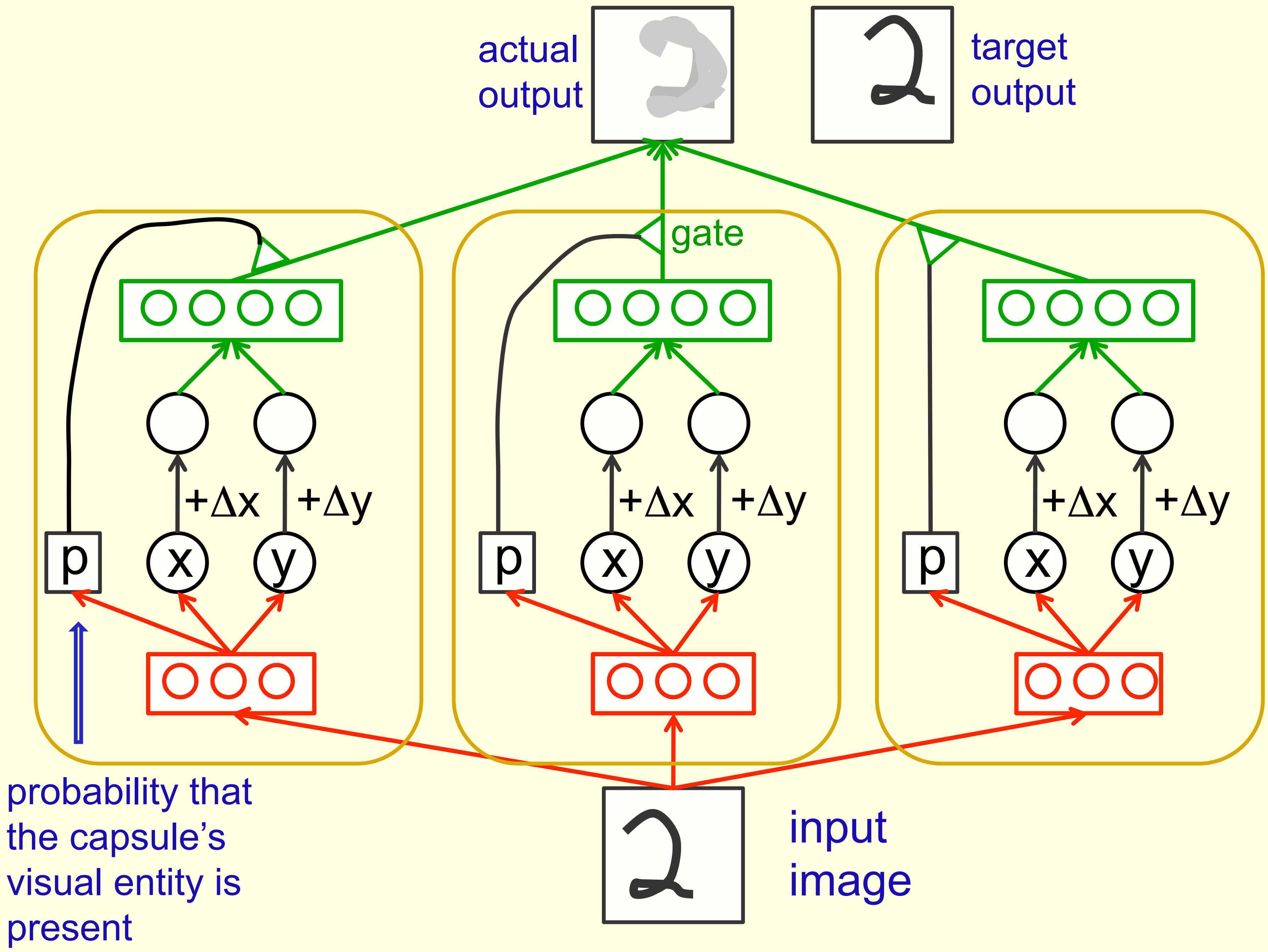
- Each capsule uses its own set of "recognition" hidden units to extract the x and y coordinates of the visual entity it represents (and also the probability of existence)

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

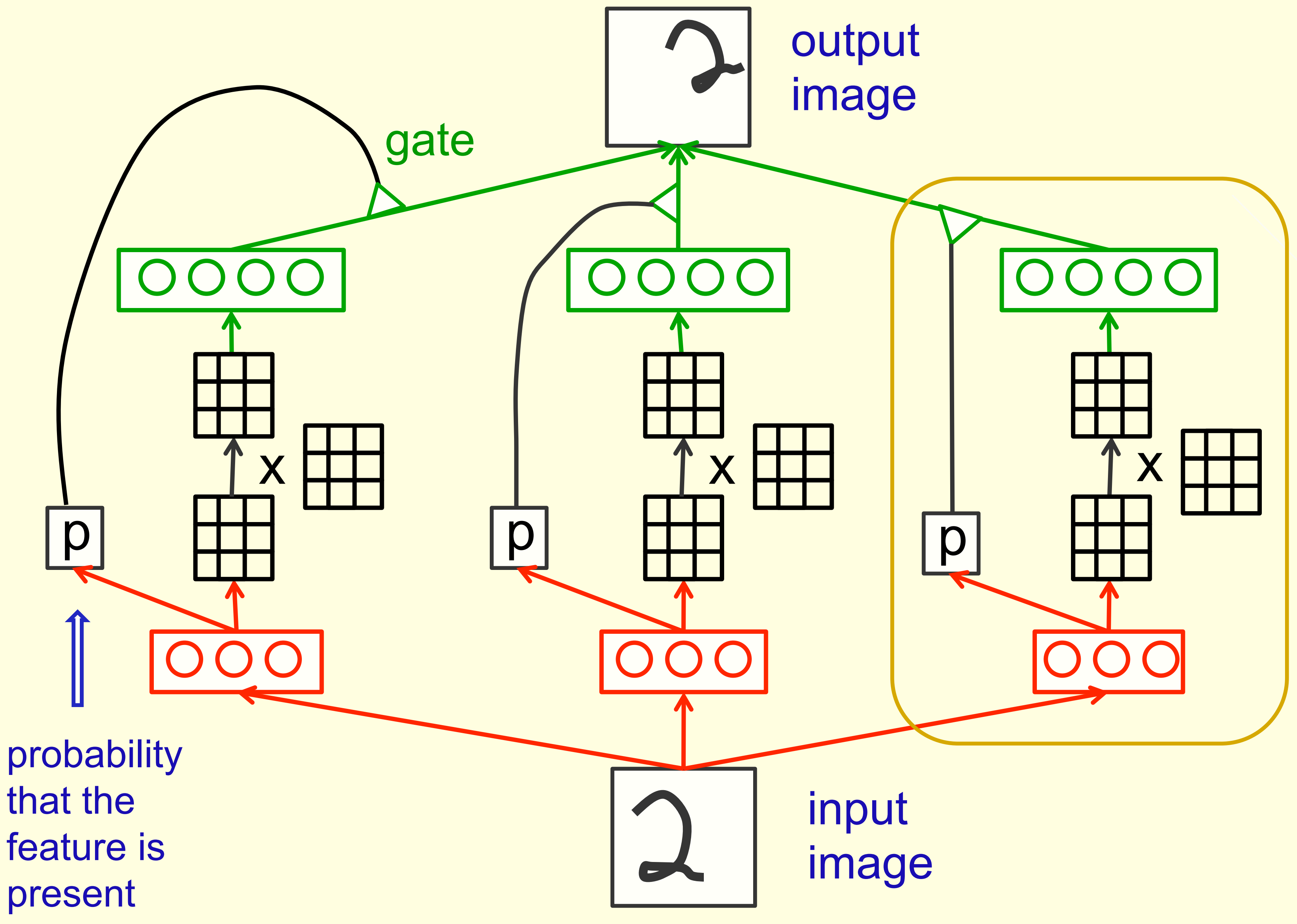
- Each capsule uses its own set of "generative" hidden units to compute its contribution to the prediction.


actual output
target output
gate
p
x
y
+Δx
+Δy
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.


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

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.
- Transforming autoencoders 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.

Relationship to Slow Feature Analysis

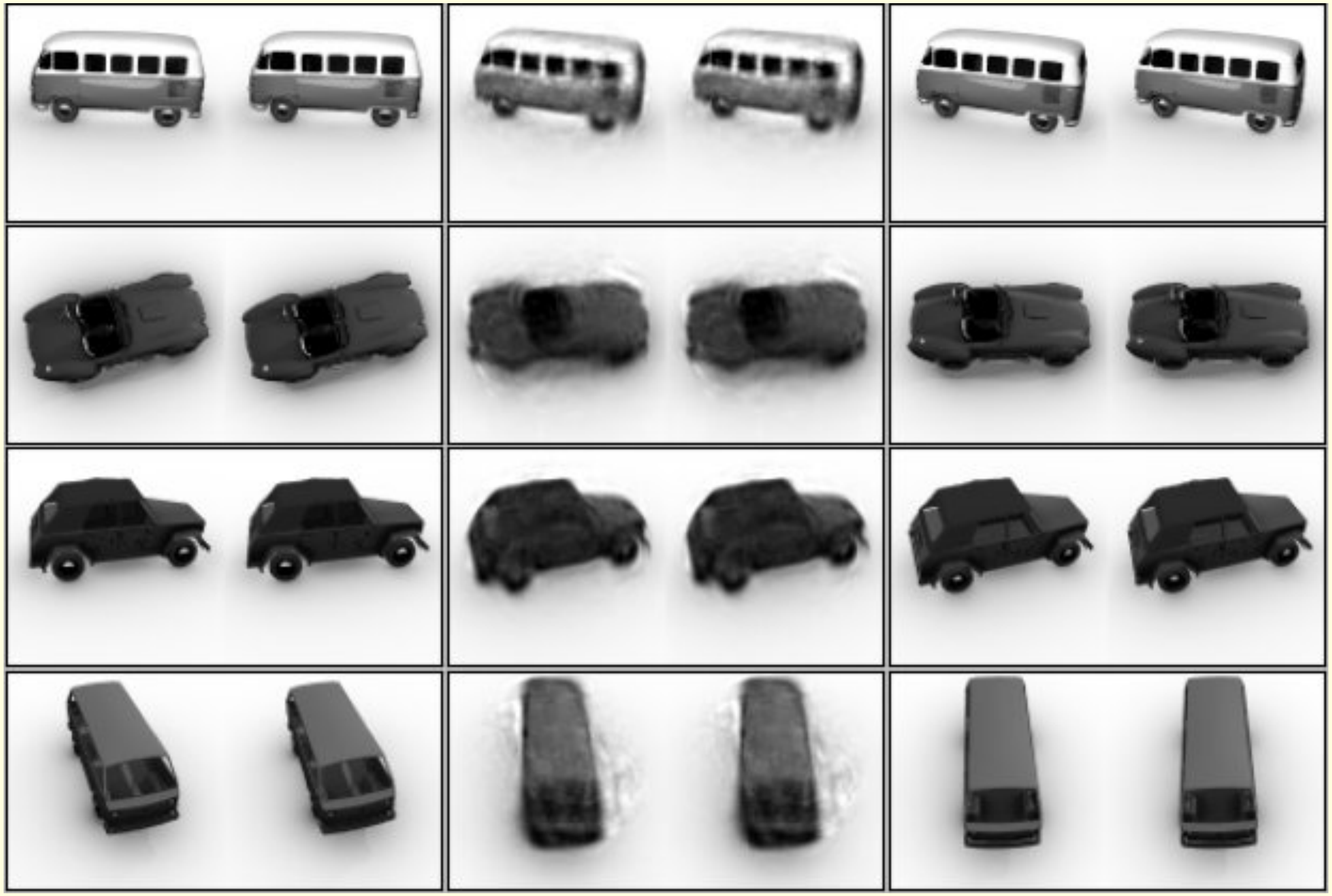
- Slow feature analysis uses a very primitive model of the dynamics.
  - It assumes that the underlying state does not change much.
  - This is a much weaker way of extracting features than using precise knowledge of the dynamics.
- Also, slow feature analysis only produces dumb features that do not output explicit instantiation parameters.

Dealing with the three–dimensional world

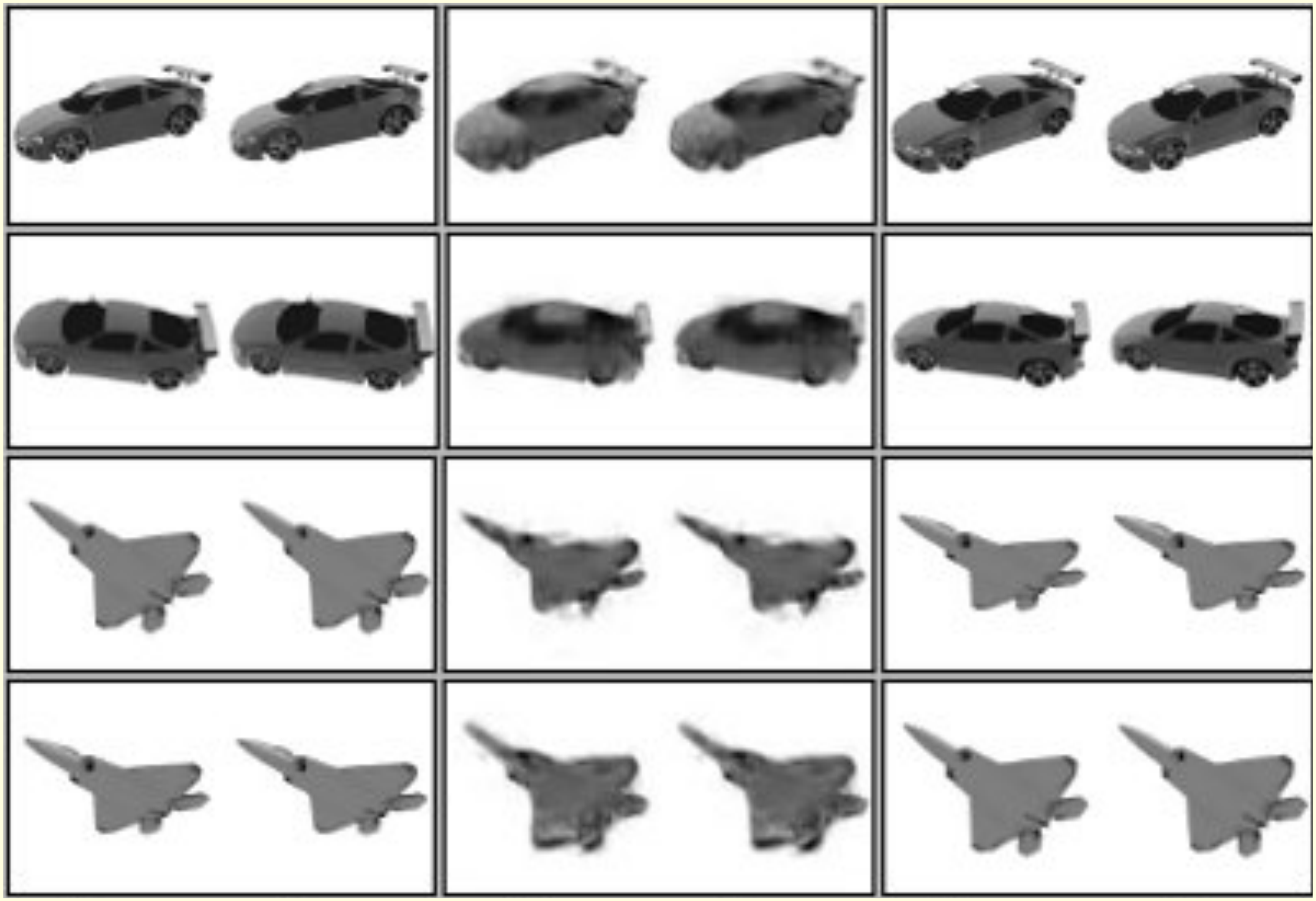
- Use stereo images to allow the encoder to extract 3-D poses.
  - Using capsules, 3-D would not be much harder than 2-D if we started with 3-D pixels.
  - The loss of the depth coordinate is a separate problem from the complexity of 3-D geometry.
- At least capsules stand a chance of dealing with the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



the end

- The work on transforming autoencoders is in a paper called “transforming autoencoders” on my webpage:
- http://www.cs.toronto.edu/~hinton/absps/transauto6.pdf

- The work on the “dropout” technique that is used to make a large number of capsules find different overlapping templates is described in a paper on arxiv:

  http://arxiv.org/abs/1207.0580