Distillation and the brain

Geoffrey Hinton, Google Brain Team & The Vector Institute

neuromatch academy

How does the brain store the knowledge about shape that it uses for recognition?

- **Theory 1:** Just learn separate appearance models for all possible viewpoints.
- **Theory 2:** Same as Theory 1 but with weight-sharing for different translations (Convolutional Neural Nets).
- **Theory 3:** Store knowledge in the weights about the viewpoint independent relationships between wholes and parts. Make the activities be viewpoint equivariant.
  - This is what computer graphics did until 2020.

If the knowledge of whole-part relationships is in the weights, how do we access it?

- We convert the viewpoint independent knowledge into viewpoint-dependent neural activities by choosing a viewpoint and forming a “mental image”.

Is the distance between the tips of the ears of a German Shepard bigger or smaller than the distance between its eyes?

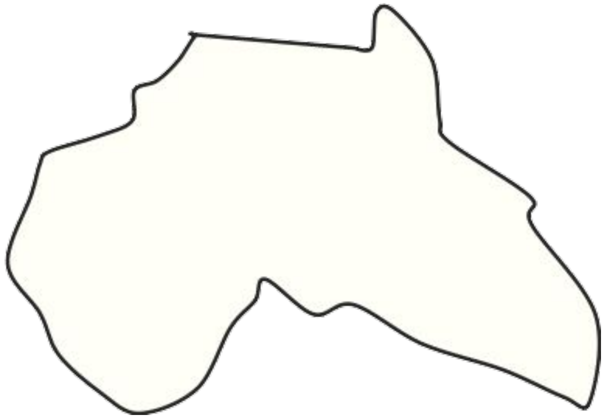
- Once we have chosen a viewpoint and formed the mental image, we can read-off spatial relationships from the neural activities.

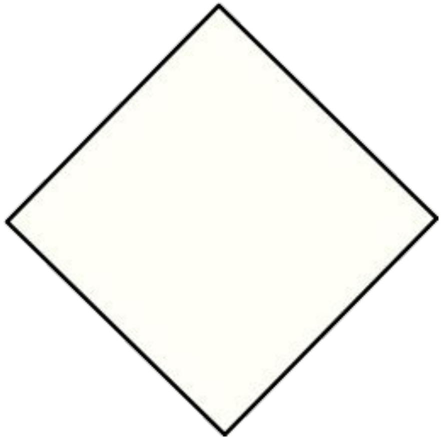
What is a spatial relationship?

- It is the coordinate transformation that maps points relative to one rectangular coordinate frame into points relative to another rectangular coordinate frame.

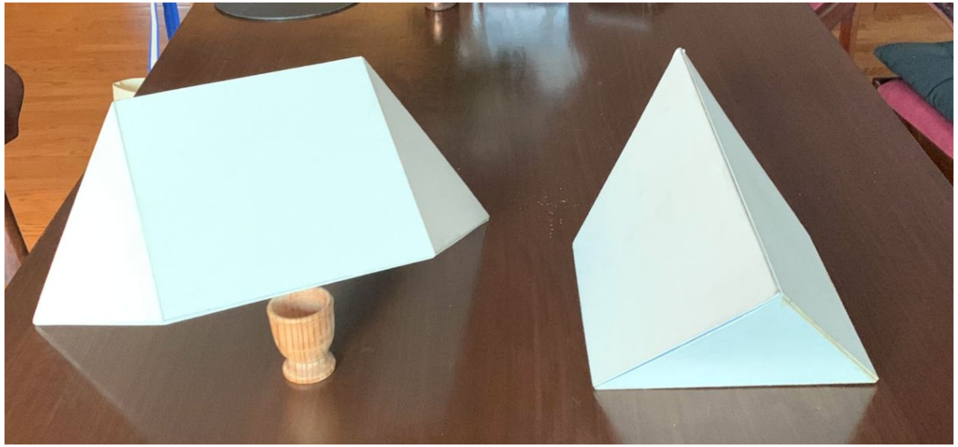
- People impose rectangular coordinate frames on wholes and parts in order to represent shape.

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





More evidence for rectangular coordinate frames



MIT professors take more than a minute to figure out how to put these two identical parts together to make a tetrahedron! Why?

Disclaimer

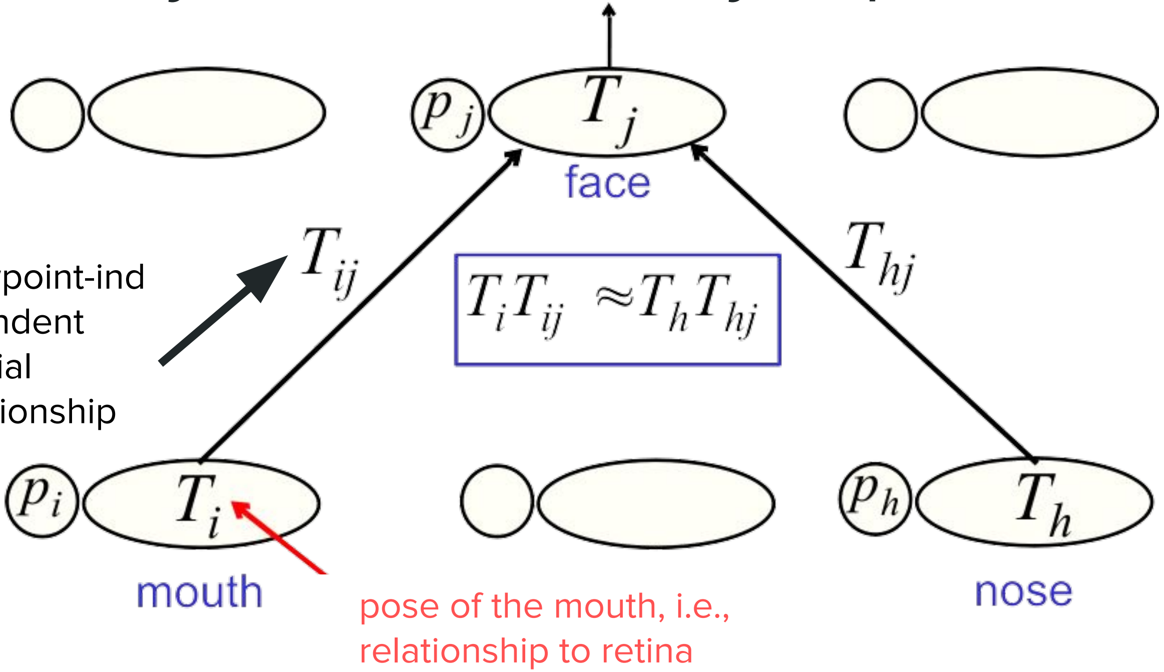
- The outer loop of vision is a sequence of intelligently chosen fixations that sample the optic array to provide the information required to perform a task.
- For each fixation we reuse the same neural net to produce a multi-level representation of the retinal image produced by that fixation.
- This talk is only about what happens on the first fixation.

Theory 3a: Identity-specific capsules

Sabour, Frosst, Hinton (2017)
Hinton, Sabour, Frosst (2018)
Kosiorek, Sabour, Teh, Hinton (2019)

- Localize knowledge about a specific shape into a specific group of neurons that knows, in its weights, how all the parts are related to the whole.
- Recognize shapes by noticing when multiple different parts predict the same pose for the whole.
    - The pose of a part is the coordinate transform between the retina and the intrinsic frame of the part.

Two layers in a hierarchy of parts



p_j
T_j
face
T_{ij}
T_{hj}
$T_i T_{ij} \approx T_h T_{hj}$
viewpoint-independent spatial relationship
p_i
T_i
mouth
p_h
T_h
nose
pose of the mouth, i.e., relationship to retina

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

Theory 3b: Ubiquitous Universal Capsules

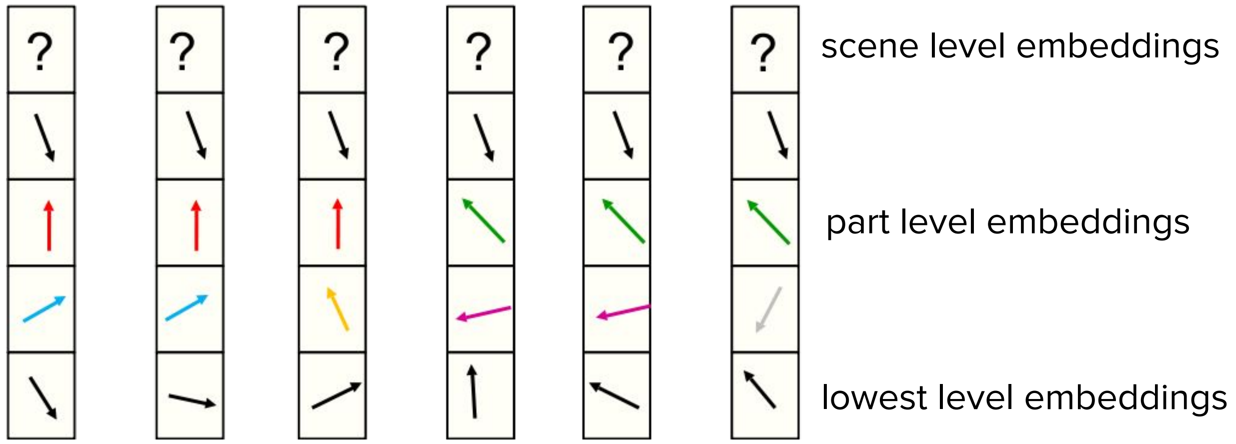
Hinton (2021): arXiv:2102.12627

- Divide the image into many small locations.
- For each location, dedicate hardware to representing whatever it is that occupies that location.
    - *i.e., use a retinotopic map*
- Have multiple different levels of representation for each location, i.e. multiple universal capsules each of which represents one level in the part-whole hierarchy.

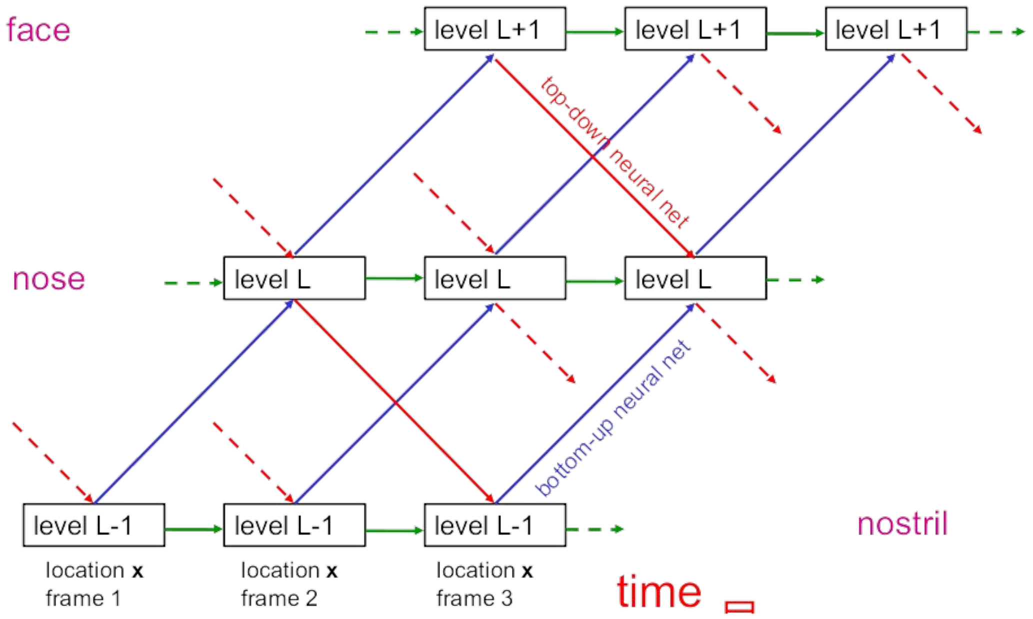
The embedding vectors for nearby columns at a single time-step as GLOM settles

At each level there are islands of agreement. These islands represent the parse tree for the scene.

It is a multi-level, real-valued Ising model with coordinate transforms between levels.



Three adjacent levels of GLOM in a single column



Interactions between and within levels

- The level L embedding at location x is an average of four contributions:

1. The bottom-up contribution from the level L-1 embedding in the same column at the previous time-step.
2. The top-down contribution from the level L+1 embedding in the same column at the previous time-step.
3. The level L embedding at the previous time-step.
4. The attention-weighted average of the level L embeddings in other nearby columns at the previous time step.

The attention-weighted average

- The level L embedding at location **x** tries to agree with similar level L embeddings at other locations.
    - The attention weighted average of the level L embeddings at other locations, y, uses weights proportional to exp[L(x) L(y)]
    - This causes the level L embeddings to form islands of similar embeddings.
        - Islands are echo chambers.

One way to deal with ambiguous parts: disambiguation at the part level

- A possible nose could interact directly with a possible mouth.
- They disambiguate each other if they have the right spatial relationship.
- So we need a “transformational random field” in which the pose of the nose predicts the pose of the mouth via a nose->mouth coordinate transform (and vice versa).
- N interacting parts need O(N^2) coordinate transforms between parts.

A different way to deal with ambiguous parts: The Hough transform

- Instead of allowing the parts to interact directly, allow each part to make an ambiguous multimodal prediction for the identity and pose of the whole object in the same column.
    - Unlike earlier capsules, no dynamic routing is required.
- The whole is present if many multimodal predictions from different columns can agree on a mode via their lateral attention-weighted interactions.
- If each column predicts an unnormalized log probability distribution over the space of possible object instances, we can simply add the predicted distributions in different columns.
    - But we must use attention so we only add similar distributions.

How to implement multimodal predictions in the joint space of identity and pose

- Each neuron in the embedding vector for the object is a basis function that represents a vague distribution in the unnormalized log probability space.
- The activity of the neuron scales this log distribution.
- The full object embedding vector represents the sum of these scaled log distributions.
- The individual distributions can be very vague because they only need to represent one thing at a time: the object occupying that location.

A big problem for ubiquitous universal capsules as a brain theory

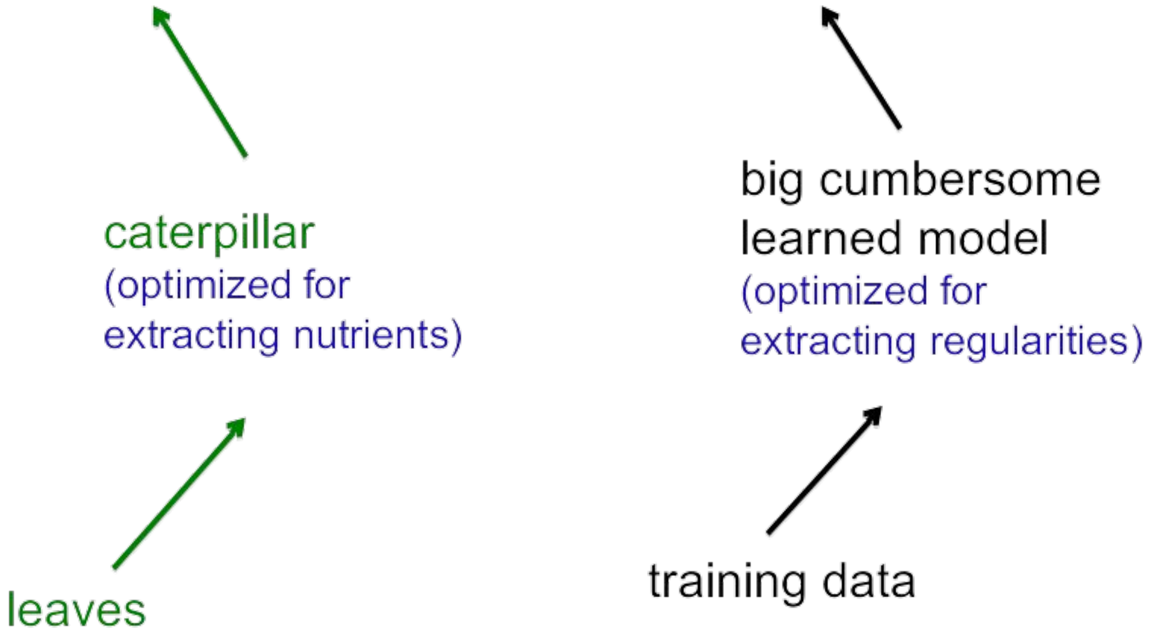
- The bottom-up and top-down neural networks at every location need to be the same.
- It seems extremely wasteful to learn all this knowledge separately for each location.
    - In a computer we can just share the weights of the local neural networks.
    - So how can we get the effect of weight-sharing in a brain?

A brief introduction to distillation

- Distillation is a way of extracting the knowledge from one model and putting it into a model with a different structure.

- It is typically used to convert the knowledge learned by a big model or an ensemble of big models into a smaller model.
  - Such as a speech recognizer that runs on your phone instead of in a data-center.

The analogy that distillation is based on


caterpillar
(optimized for extracting nutrients)
leaves
big cumbersome learned model
(optimized for extracting regularities)
training data

The main idea

- The big cumbersome model implements a function from input to output.
- Forget the architecture and weights of the big model and focus on the function.
    - After learning the big model, we have our hands on the function.
- How do we transfer the knowledge in the function into a model with a different architecture?

Distillation: A way to transfer the knowledge

- If the output is a big N-way softmax, the targets are usually a single 1 and a whole lot of 0's.
    - On average each target puts at most log N bits of constraint on the function.
- If we have our hands on a big model, we can divide its logits by a "temperature" to get a much softer distribution.

$$\boldsymbol{p_i} = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

This reveals much more information about the function on each training case.

An example of hard and soft targets

cow	dog	cat	car
0	1	0	0

Original hard targets

cow	dog	cat	car
10^{-6}	.9	.1	10^{-9}

Output of a big model

cow	dog	cat	car
0.05	.3	.2	.00

Softened output of the big model

Softened outputs reveal the dark knowledge in the big model. Most of its knowledge is in the relative probabilities of wrong answers.

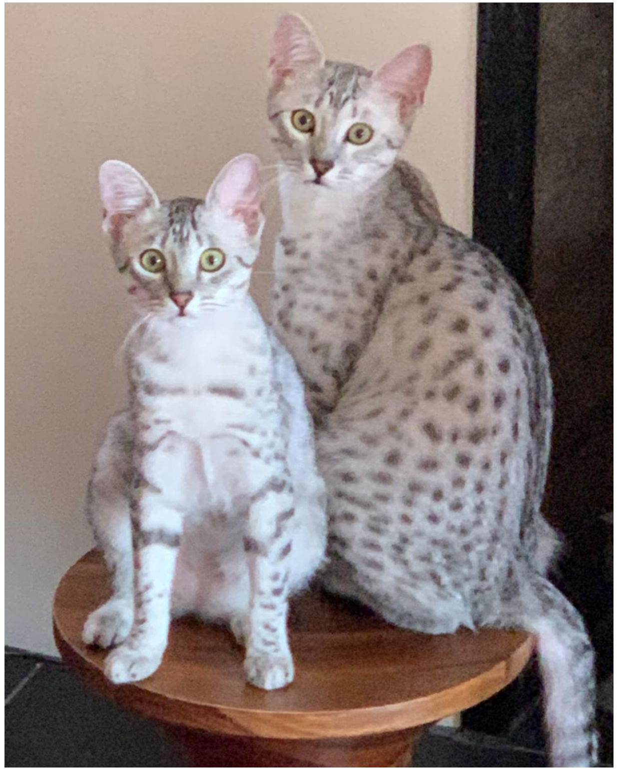
What species is this?

Snow leopard?

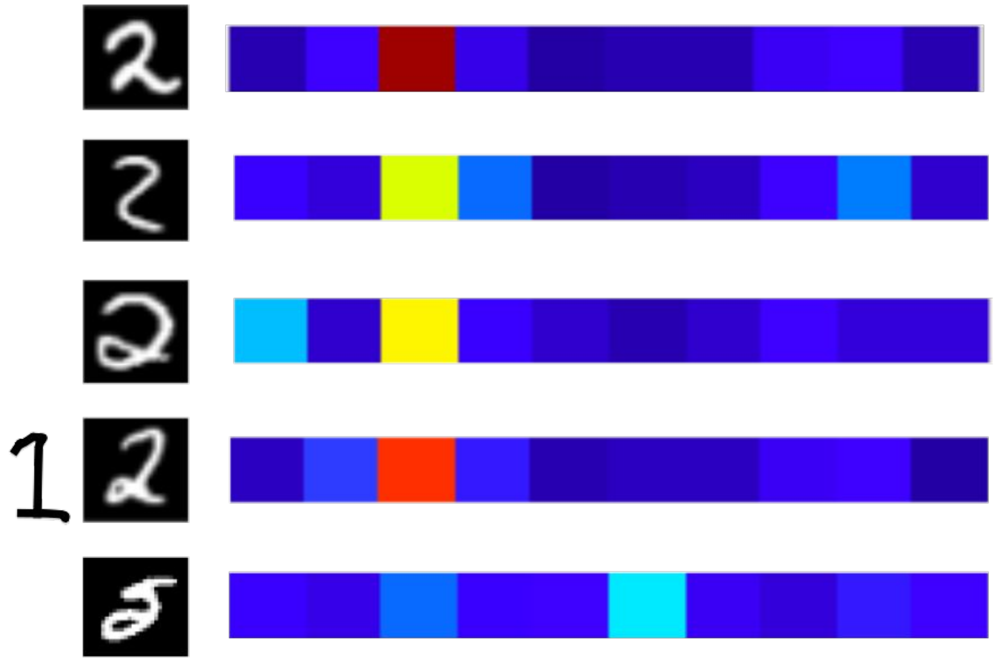
Tiger?

Cow?

Carrot?



Some examples of soft targets



Experiment on MNIST

- Vanilla backprop in a 784 ➔ 800 ➔ 800 ➔ 10 net with rectified linear hidden units gives 146 test errors.
    - RELU: y = max(0, x)
- If we train a 784 ➔ 1200 ➔ 1200 ➔ 10 net using **dropout** and **weight constraints** and **jittering the input**, we eventually get 67 errors.
- How much of this improvement can be transferred to the 784 ➔ 800 ➔ 800 ➔ 10 net?

Transfer to the small net

- Using both the soft targets obtained from the big net and the hard targets, we get 74 errors in the 784 → 800 → 800 → 10 net.
  - The transfer training uses the same training set but with **no dropout** and **no jitter**.
  - It’s just vanilla backprop (with added soft targets).
- The soft targets show the small net how to generalize well (if you trust the big model).
  - In distillation, we are finally training with the correct objective function which is to generalize well.

The mythical digit three:
A very surprising result on MNIST

- Using soft targets from the big model, train the 784 ➔ 800 ➔ 800 ➔ 10 net on a transfer set that does not contain any examples of a 3.
- After the transfer training, raise the bias of the 3 (the distilled net thinks that 3's are very rare).
    - The distilled net then gets **98.6%** of the test threes correct even though it never saw any threes during the transfer training.

Co-distillation: training a community of neural nets

- If we train ten 784 → 500 → 300 → 10 nets independently on MNIST, they average 158 test errors. The ensemble gets 143 errors.
- What if we let each net try to match soft targets derived by averaging the opinions of the whole community as it is training? (in addition to matching the hard targets)
    - The nets now average 126 errors!
    - The ensemble gets 120 errors.

How can columns share knowledge in a brain?

- In GLOM, the bottom-up and top-down neural nets between two adjacent levels are the same for all columns.
    - But a brain cannot share weights.

- All we actually need to share is the **knowledge** in different columns.
    - We can do this by co-distillation

Transferring knowledge between columns

- In each column, the bottom-up and top-down neural nets treat the consensus embedding at level L as a target.
- The consensus embedding is the average of
  - the bottom-up prediction for level L from level L+1 at that location,
  - the top-down prediction for level L from level L-1 at that location,
  - the attention-weighted predictions from nearby level L embeddings.
- By trying to model the consensus embedding, each local neural net is using the neural nets in nearby columns as a teacher.
  - The whole system is doing co-distillation.

Why distillation is actually better than weight-sharing

- The retina does not have uniform resolution and the photoreceptors are not in a regular grid.
- So the optic array is preprocessed differently in each location.
- We want the neural nets at each location to compute the same function of the optic array.
  - Distillation achieves this

THE END

Paper on distillation: arXiv:1503.02531

Paper on GLOM: arXiv:2102.12627