HMAX Models
Architecture

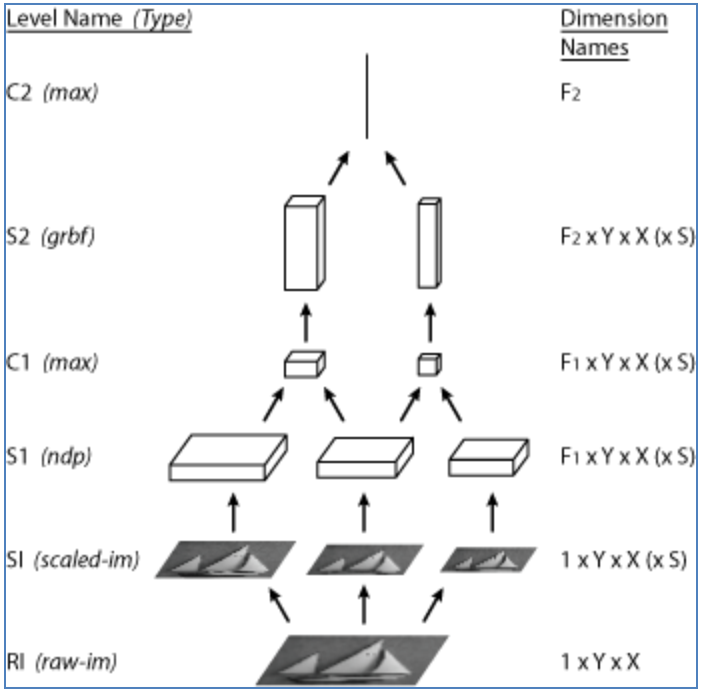
Jim Mutch

March 31, 2010

Topics

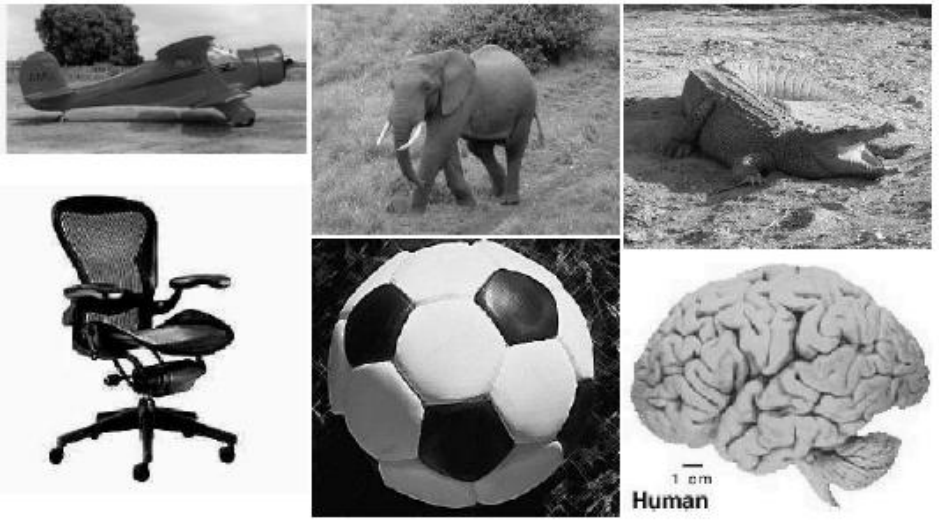
- Basic concepts:
  - Layers, operations, features, scales, etc.
  - Will use one particular model for illustration; concepts apply generally.
- Introduce software.
- Model variants.
  - Attempts to find best parameters.
- Some current challenges.

Example Model

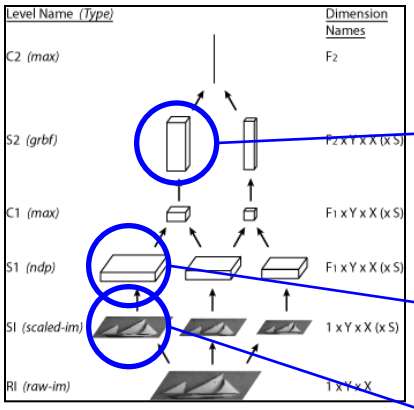


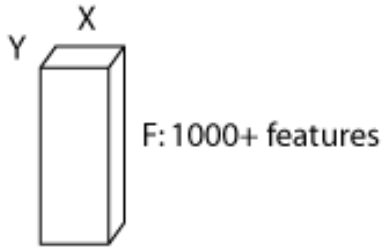
- Our best-performing model for multiclass categorization (with a few simplications).
- Similar to:
  - J. Mutch and D.G. Lowe. *Object class recognition and localization using sparse features with limited receptive fields.* IJCV 2008.
  - T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. *CVPR* 2005.

- Results on the Caltech 101 database: around 62%.
- State of the art is in the high 70s using multiple kernel approaches.



Layers

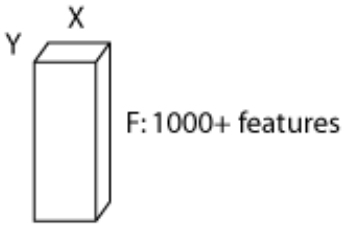


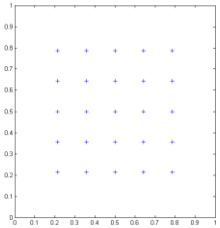


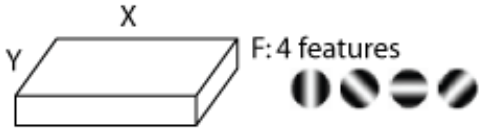
- A layer is a 3-D array of units which collectively represent the activity of some set of features (F) at each location in a 2-D grid of points in retinal space (X, Y).

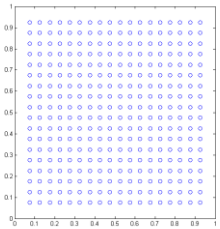
- The number and kind of features change as you go higher in the model.
  - Input: only one feature (pixel intensity).
  - S1 and C1: responses to gabor filters of various orientations.
  - S2 and C2: responses to more complex features.

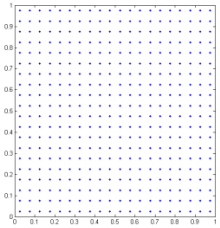
Common Retinal Coordinate System for (X, Y)







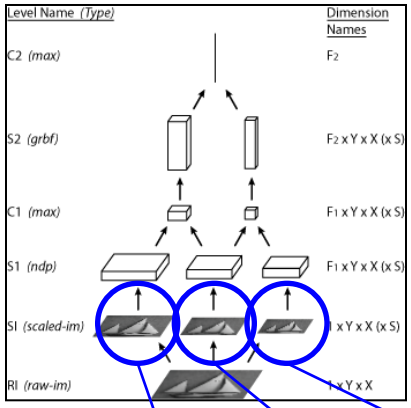




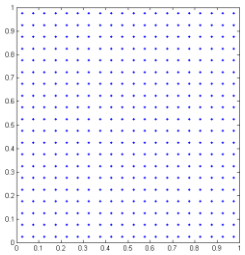
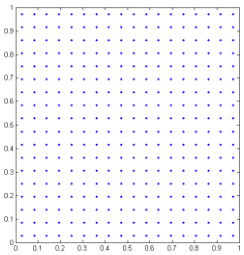
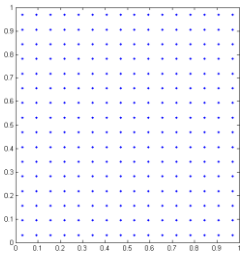
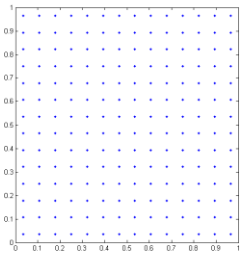
- The number of (X, Y) positions in a layer gets smaller as you go higher in the model.
  - (X,Y) indices aren't meaningful across layers.
- However: each layer's cells still cover the entire retinal space.
  - With wider spacing.
  - With some loss near the edges.

- Each cell knows its (X, Y) center in a **real-valued retinal coordinate system** that is **consistent across layers**.
  - Keeping track of this explicitly turns out to simplify some operations.

Scale Invariance



- In a single visual cortical area (e.g. V1) you will find cells tuned to different spatial scales.
- For simplicity in our computational models, we represent different spatial scales using **multiple** layers.

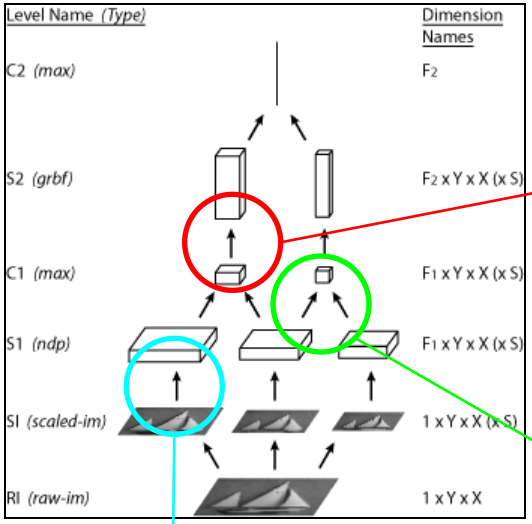





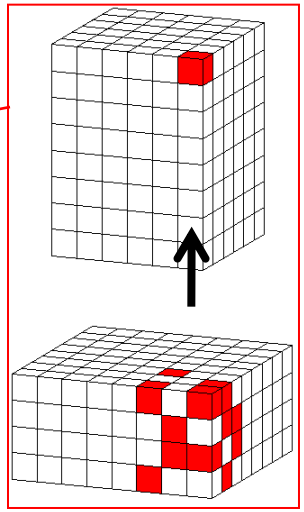
...

- Finer scales have more (X, Y) positions.
  - Each such position represents a smaller region of the visual field.
- Not all scales are shown (there are 12 in total).

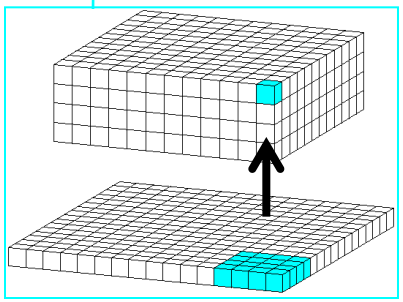
Operations

- Every cell is computed using cells in layer(s) immediately below as inputs.
- We always pool over a local region in (X, Y) …

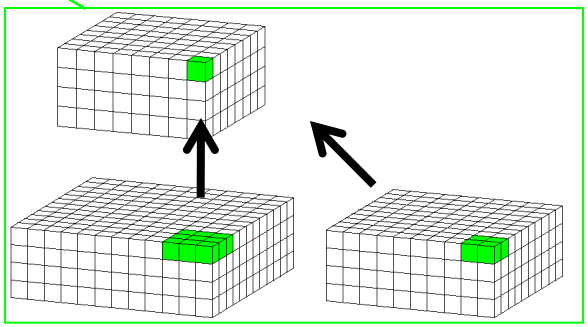




… sometimes over multiple feature types.

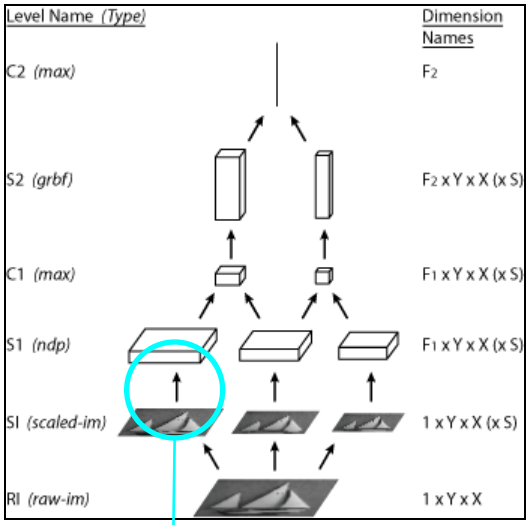


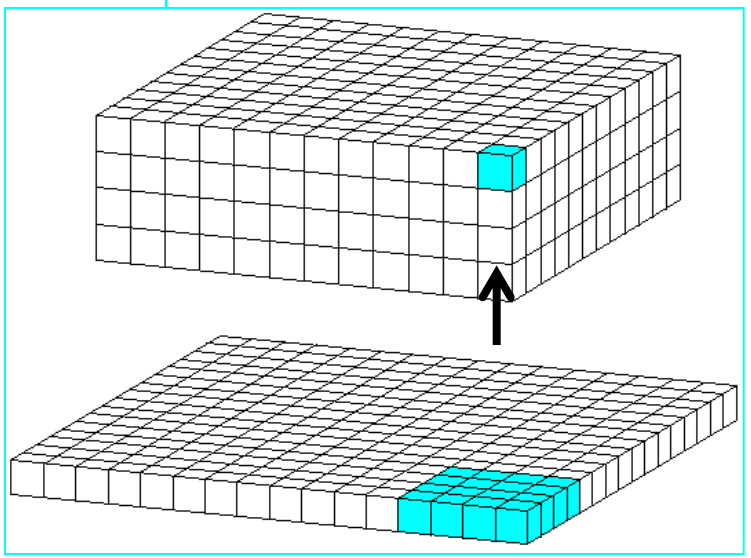
… sometimes over one scale at a time.



… sometimes over multiple scales (tricky!)

S1 (Gabor Filter) Layers



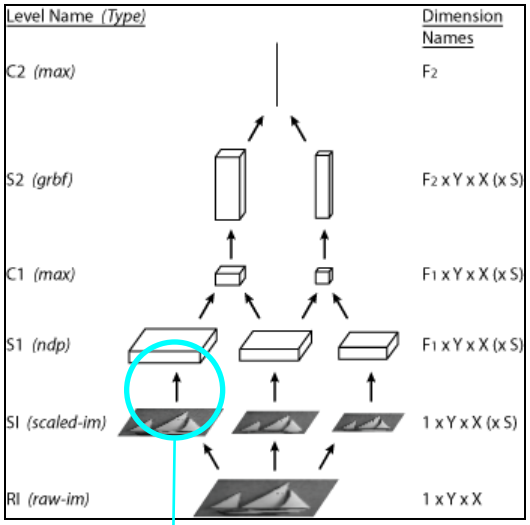


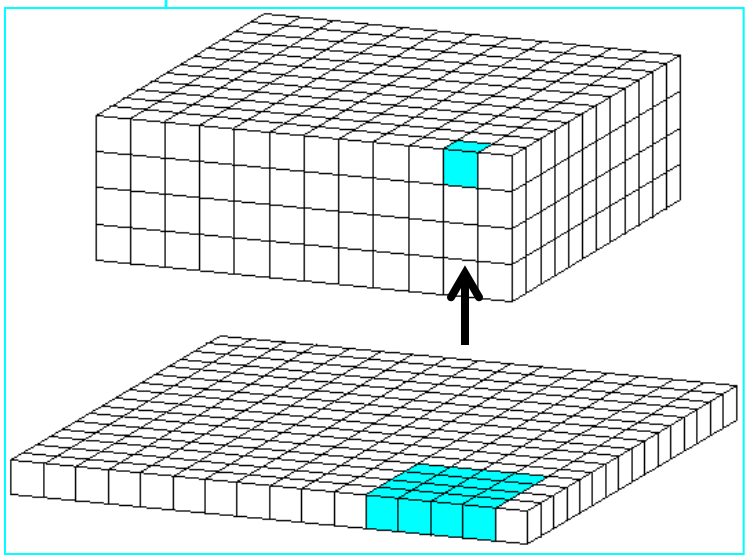
- Image (at finest scale) is [256 x 256 x 1].
- Only 1 feature at each grid point: image intensity.

- Center **4 different gabor filters** over each pixel position.
- Resulting S1 layer (at finest scale) is [246 x 246 x 4].
  - Can't center filters over pixels near edges.
  - Actual gabors are 11 x 11.

$$R(X,G) = \left| \frac{\sum X_i G_i}{\sqrt{\sum X_i^2}} \right|$$

S1 (Gabor Filter) Layers



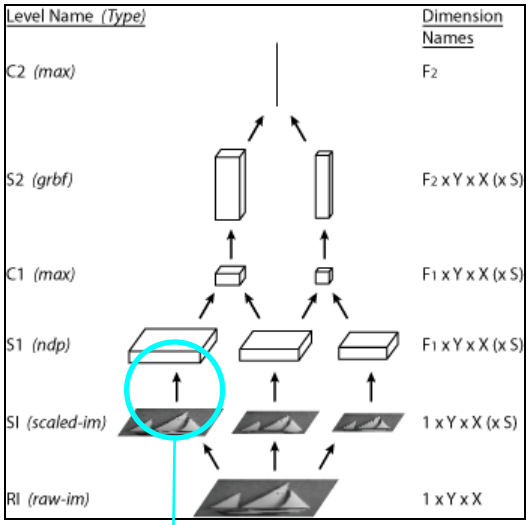


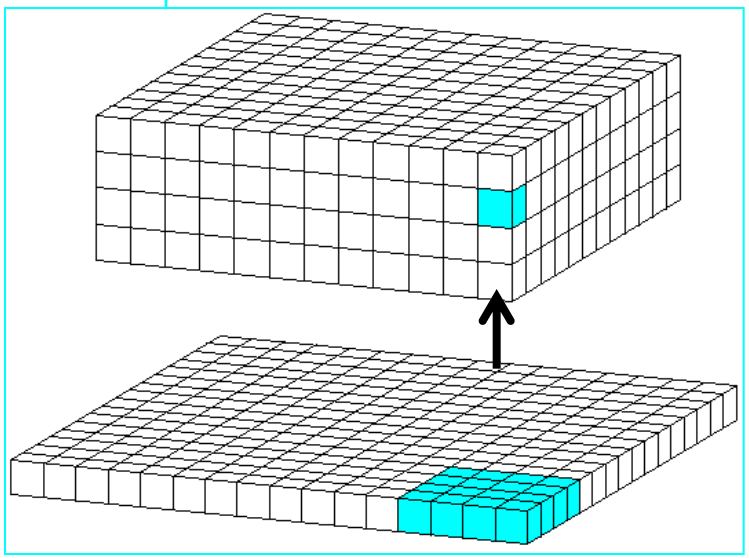
- Image (at finest scale) is [256 x 256 x 1].
- Only 1 feature at each grid point: image intensity.

- Center **4 different gabor filters** over each pixel position.
- Resulting S1 layer (at finest scale) is [246 x 246 x 4].
  - Can't center filters over pixels near edges.
  - Actual gabors are 11 x 11.

$$R(X,G) = \left| \frac{\sum X_i G_i}{\sqrt{\sum X_i^2}} \right|$$

S1 (Gabor Filter) Layers



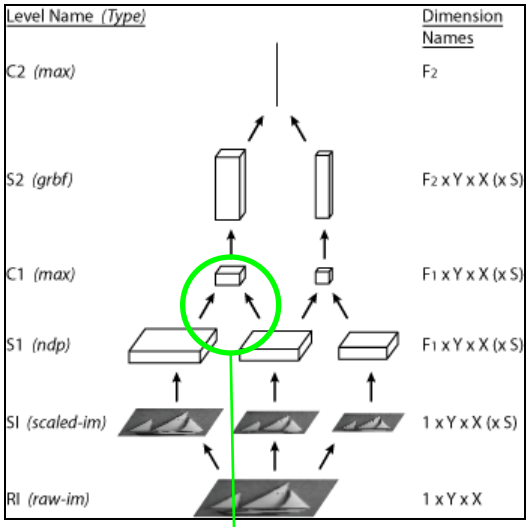


- Image (at finest scale) is [256 x 256 x 1].
- Only 1 feature at each grid point: image intensity.

- Center **4 different gabor filters** over each pixel position.
- Resulting S1 layer (at finest scale) is [246 x 246 x 4].
  - Can't center filters over pixels near edges.
  - Actual gabors are 11 x 11.

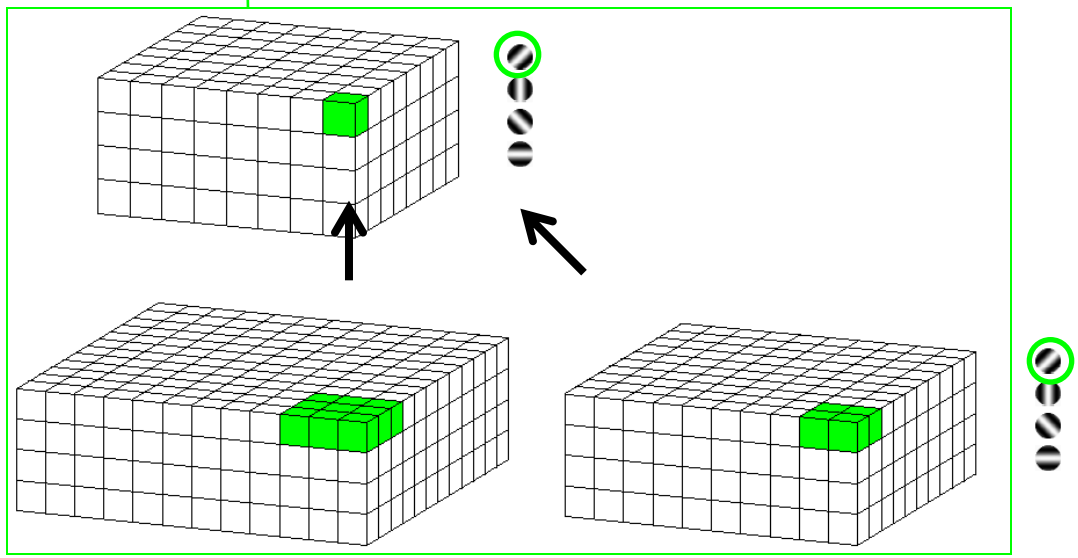
$$R(X,G) = \left| \frac{\sum X_i G_i}{\sqrt{\sum X_i^2}} \right|$$

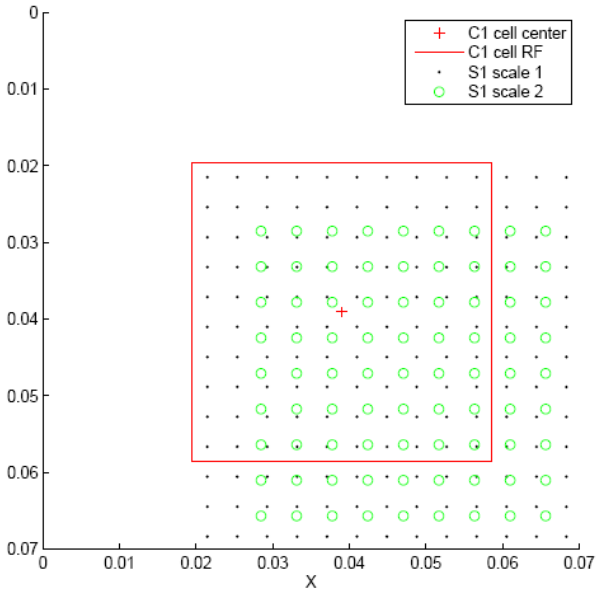
C1 (Local Invariance) Layers



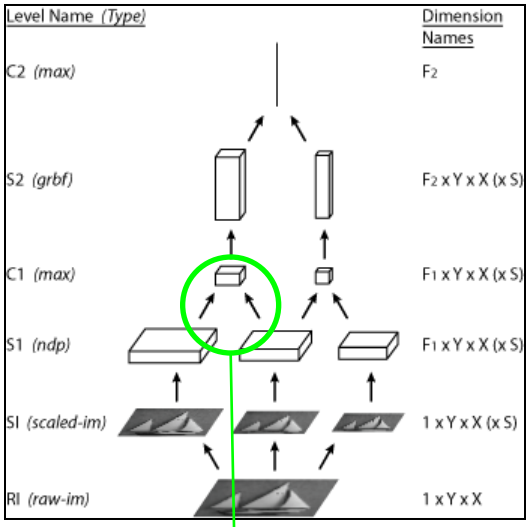
- S1 layer (finest scale) is [246 x 246 x 4].
- For each orientation we compute a **local maximum over (X, Y) and scale**.
- We also **subsample** by a factor of 5 in both X and Y.
- Resulting C1 layer (finest scale) is [47 x 47 x 4].

- Pooling over scales is tricky to define because adjacent scales differ by non-integer multiples. The common, real-valued coordinate system helps.



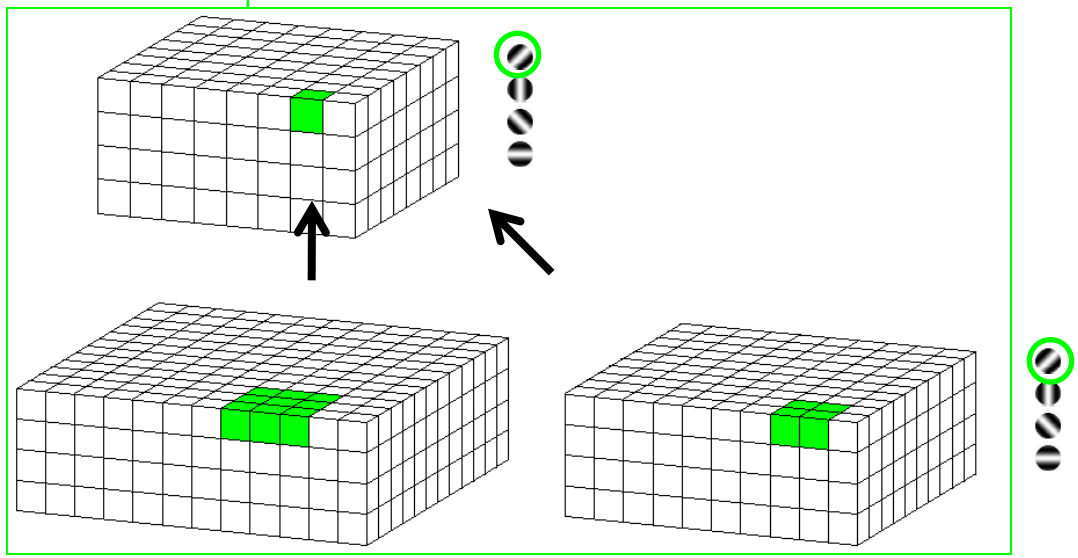


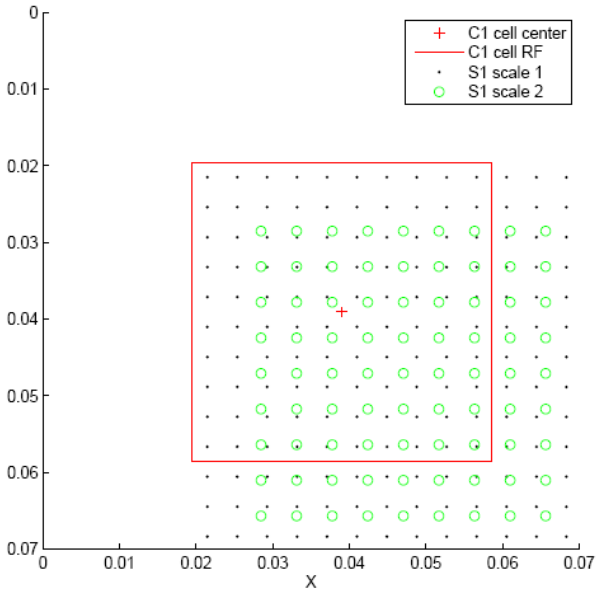
C1 (Local Invariance) Layers



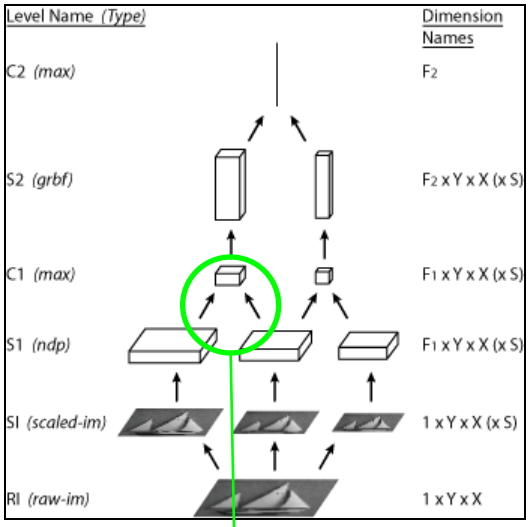
- S1 layer (finest scale) is [246 x 246 x 4].
- For each orientation we compute a **local maximum over (X, Y) and scale**.
- We also **subsample** by a factor of 5 in both X and Y.
- Resulting C1 layer (finest scale) is [47 x 47 x 4].

- Pooling over scales is tricky to define because adjacent scales differ by non-integer multiples. The common, real-valued coordinate system helps.



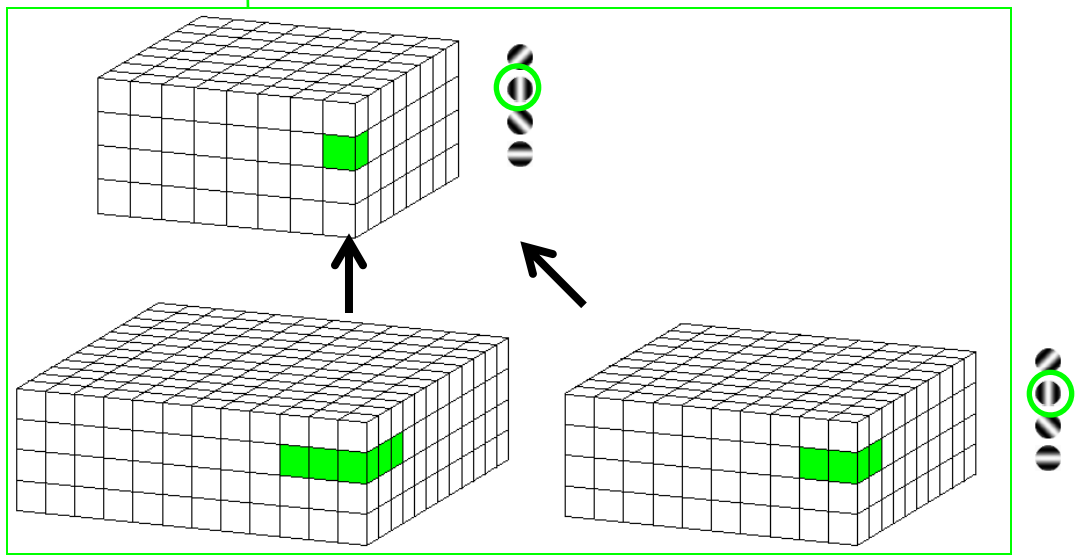


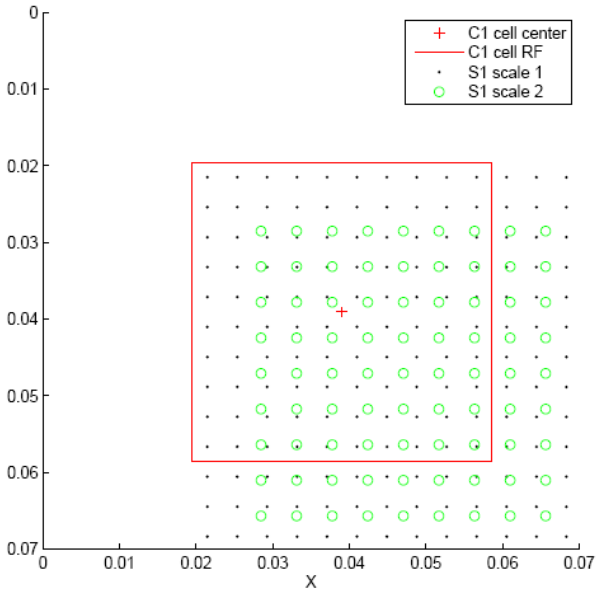
C1 (Local Invariance) Layers



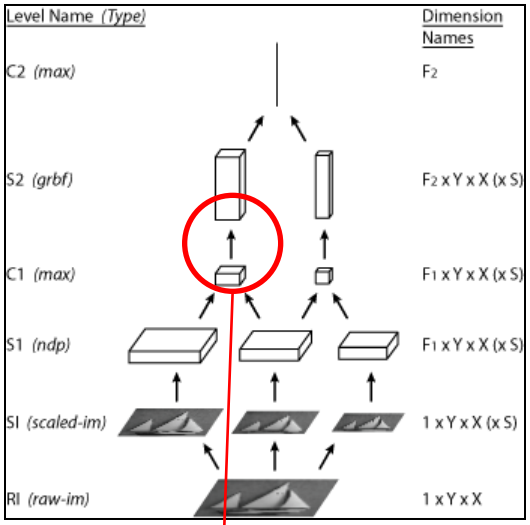
- S1 layer (finest scale) is [246 x 246 x 4].
- For each orientation we compute a **local maximum over (X, Y) and scale**.
- We also **subsample** by a factor of 5 in both X and Y.
- Resulting C1 layer (finest scale) is [47 x 47 x 4].

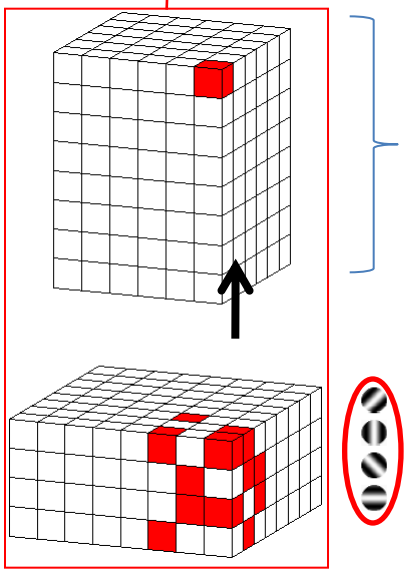
- Pooling over scales is tricky to define because adjacent scales differ by non-integer multiples. The common, real-valued coordinate system helps.





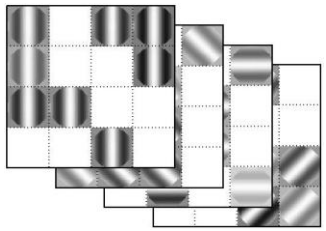
S2 (Intermediate Feature) Layers





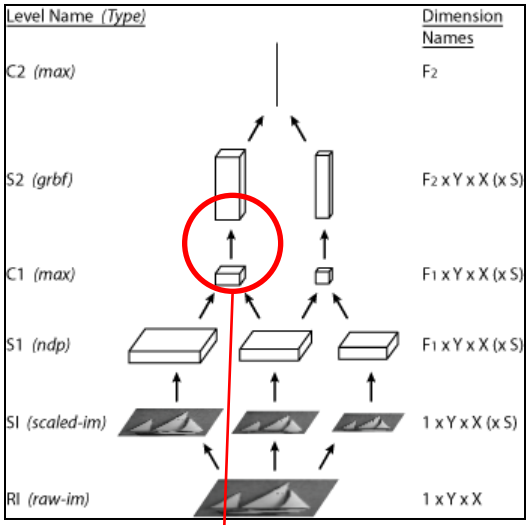
4000 features

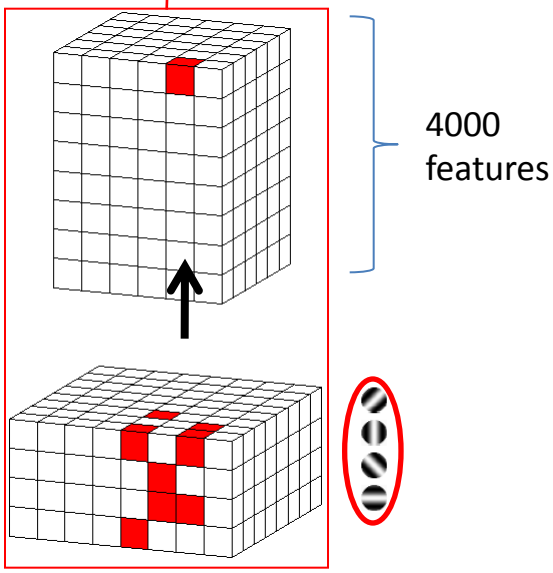
- C1 layer (finest scale) is [47 x 47 x 4].
- We now compute the response to (the same) large dictionary of **learned** features at each C1 grid position (separately for each scale).
- Each feature is looking for its preferred stimulus: a particular **local combination of different gabor filter responses** *(each of which is already locally invariant)*.
- Features can be of different sizes in (X, Y).
- Resulting S2 layer (finest scale) is [44 x 44 x 4000].
- The dictionary is learned by **sampling** from the C1 layer of training images.
  - Can decide to ignore some orientations at each position:



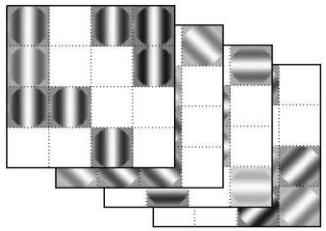
$$R(X,P) = \exp\left(-\frac{\|X-P\|^2}{2\sigma^2}\right)$$

S2 (Intermediate Feature) Layers



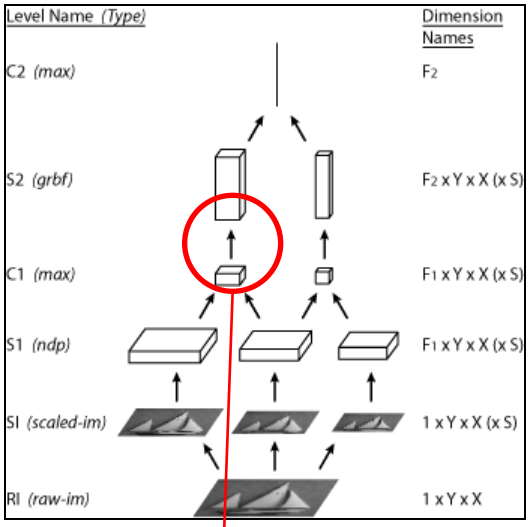


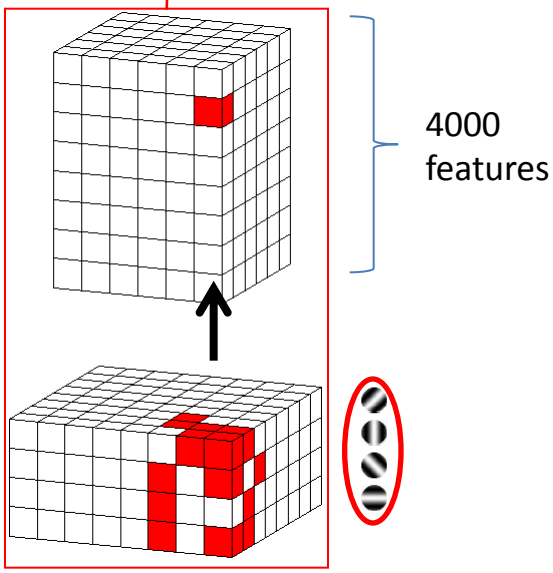
- C1 layer (finest scale) is [47 x 47 x 4].
- We now compute the response to (the same) large dictionary of **learned** features at each C1 grid position (separately for each scale).
- Each feature is looking for its preferred stimulus: a particular **local combination of different gabor filter responses** *(each of which is already locally invariant)*.
- Features can be of different sizes in (X, Y).
- Resulting S2 layer (finest scale) is [44 x 44 x 4000].
- The dictionary is learned by **sampling** from the C1 layer of training images.
  - Can decide to ignore some orientations at each position:



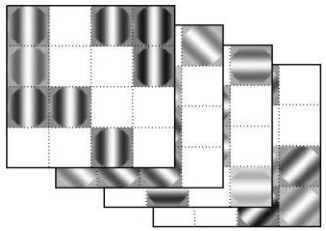
$$R(X,P) = \exp\left(-\frac{\|X-P\|^2}{2\sigma^2}\right)$$

S2 (Intermediate Feature) Layers



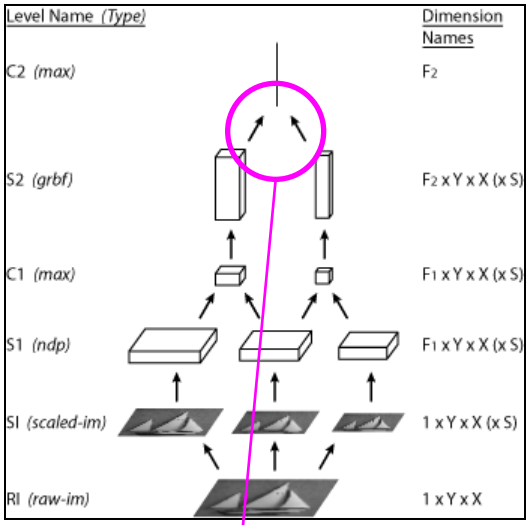


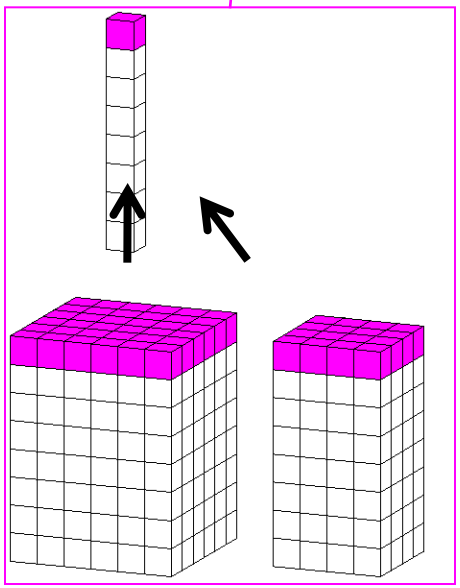
- C1 layer (finest scale) is [47 x 47 x 4].
- We now compute the response to (the same) large dictionary of **learned** features at each C1 grid position (separately for each scale).
- Each feature is looking for its preferred stimulus: a particular **local combination of different gabor filter responses** *(each of which is already locally invariant)*.
- Features can be of different sizes in (X, Y).
- Resulting S2 layer (finest scale) is [44 x 44 x 4000].
- The dictionary is learned by **sampling** from the C1 layer of training images.
  - Can decide to ignore some orientations at each position:



$$R(X,P) = \exp\left(-\frac{\|X-P\|^2}{2\sigma^2}\right)$$

C2 (Global Invariance) Layers



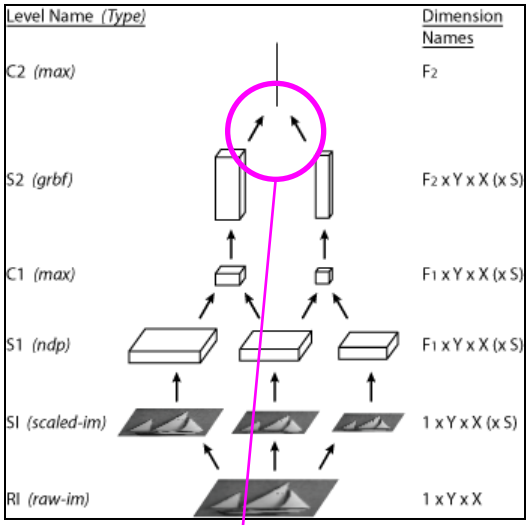


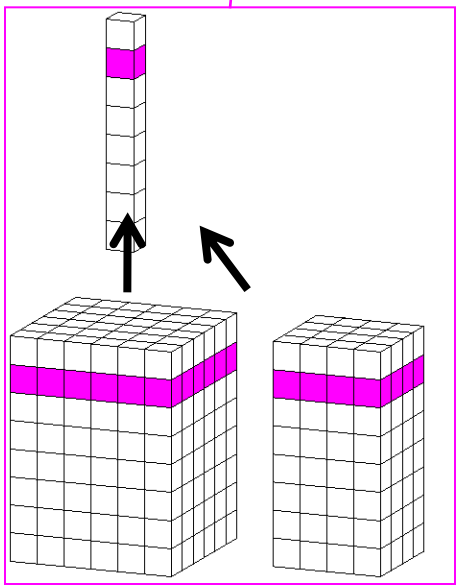
4000 features

4000 features

- Finally, we find the **maximum response to each intermediate feature over all (X, Y) positions and all scales**.
- Result: a 4000-D feature vector which can be used in the classifier of your choice.

C2 (Global Invariance) Layers





4000 features

4000 features

- Finally, we find the **maximum response to each intermediate feature over all (X, Y) positions and all scales**.
- Result: a 4000-D feature vector which can be used in the classifier of your choice.

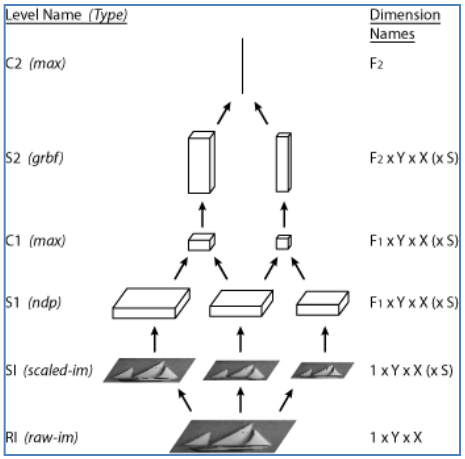
CBCL Software

http://cbcl.mit.edu/software-datasets

- Many different implementations, most now obsolete.
  - One reason: many different solutions to the "pooling over scales" problem.

- Two current implementations:
  1. "hmin" – a simple C++ implementation of exactly what I've described here.
  2. "CNS" – a much more general, GPU-based (i.e., fast) framework for simulating any kind of "cortically organized" network, i.e. a network consisting of n-dimensional layers of similar cells. Can support recurrent / dynamic models.
     - Technical report describing the framework.
     - Example packages implementing HMAX and other model classes.
     - Programming guide.

Some CNS Performance Numbers

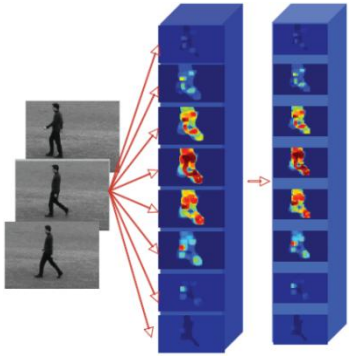


Feedforward object recognition (static CBCL model):

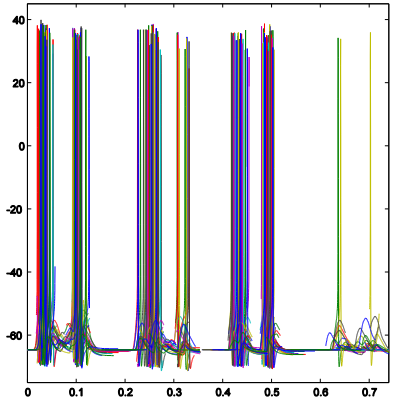
- 256x256 input, 12 orientations, 4,075 "S2" features.
- Best CPU-based implementation: 28.2 sec/image.
- CNS (on NVIDIA GTX 295): 0.291 sec/image (97x speedup).

Action recognition in streaming video:

- 8 9x9x9 spatiotemporal filters, 300 S2 features.
- Best CPU-based implementation: 0.55 fps.
- CNS: 32 fps (58x speedup).



Jhuang et al. 2007



Spiking neuron simulation (dynamic model):

- 9,808 Hodgkin-Huxley neurons and 330,295 synapses.
- 310,000 simulated time steps required 57 seconds.

Parameter Sets

- The CNS “HMAX” package (“fhpkg”) contains parameter sets for several HMAX variants, other than the one I described.
  - In particular, the more complex model used in the animal/no-animal task, which has two pathways and higher-order learned features.

- Current project: automatic searching of parameter space using CMA-ES (covariance matrix adaptation – evolutionary strategy).
  - Mattia Gazzola

Some Current Challenges

- In practice, little/no benefit is seen in models using more than one layer of learned features. (True for other hierarchical cortical models as well.) Clearly not true for the brain.

- Hard to improve on our overly-simple method of learning features (i.e. just sampling and possibly selecting ones the classifier finds useful).

- Loss of dynamic range: units at higher levels tend towards maximal activation, contrary to actual recordings.

- Better test datasets needed.