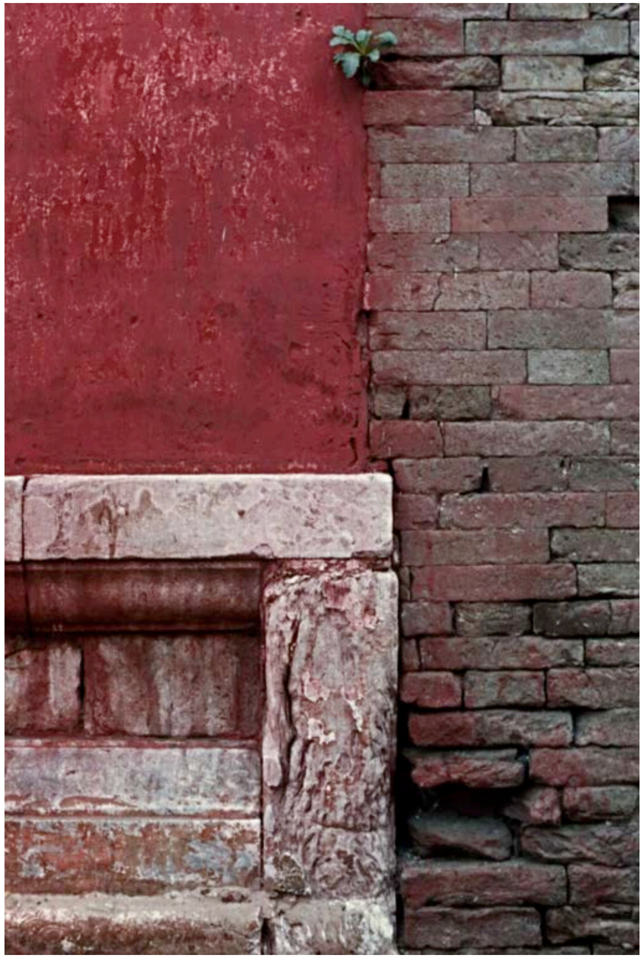
Texture

- What is texture?
- Texture analysis
- Deep Texture



Reminder: Homogeneous or Not?

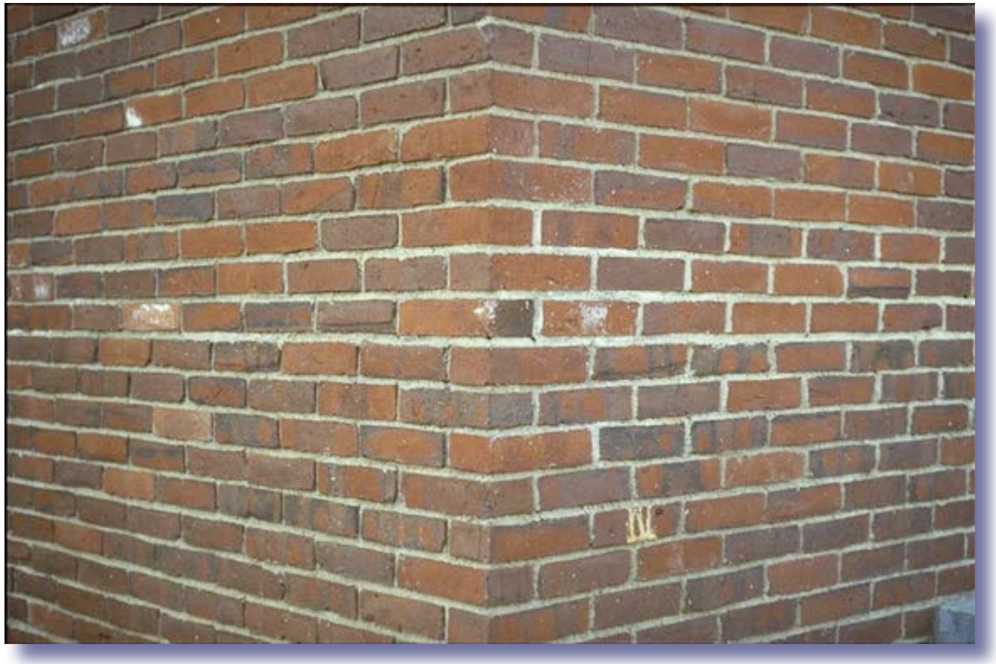
What is homogeneous in some parts of these images are the statistical properties, not the actual pixel values.

Texture-Based Segmentation

Ideally, we would like to:

- Assign to individual pixels whose texture is similar the same values to form a textural image.
- Evaluate homogeneity both in the original image and in the textural one.

Texture-Based Edges



Similarly, we would like to be able to find boundaries between textures.

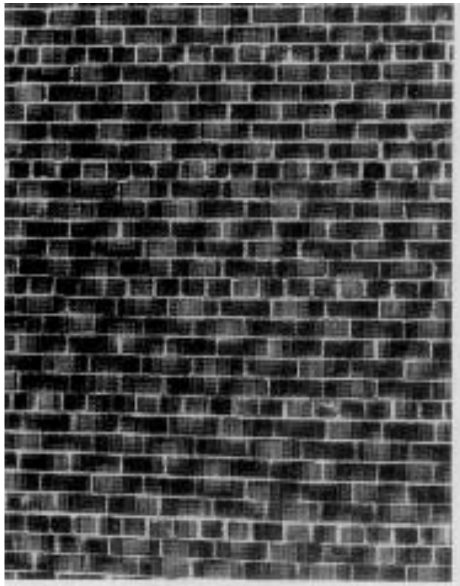
What is Texture?

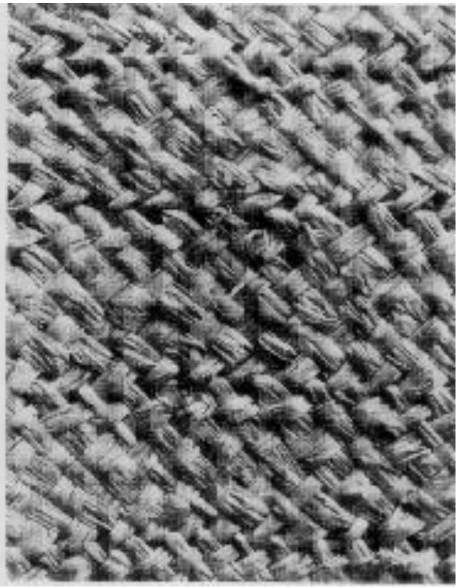
Repetition of a basic pattern:

- Structural
- Statistical

→ Non local property, subject to distortions.

Structural Textures

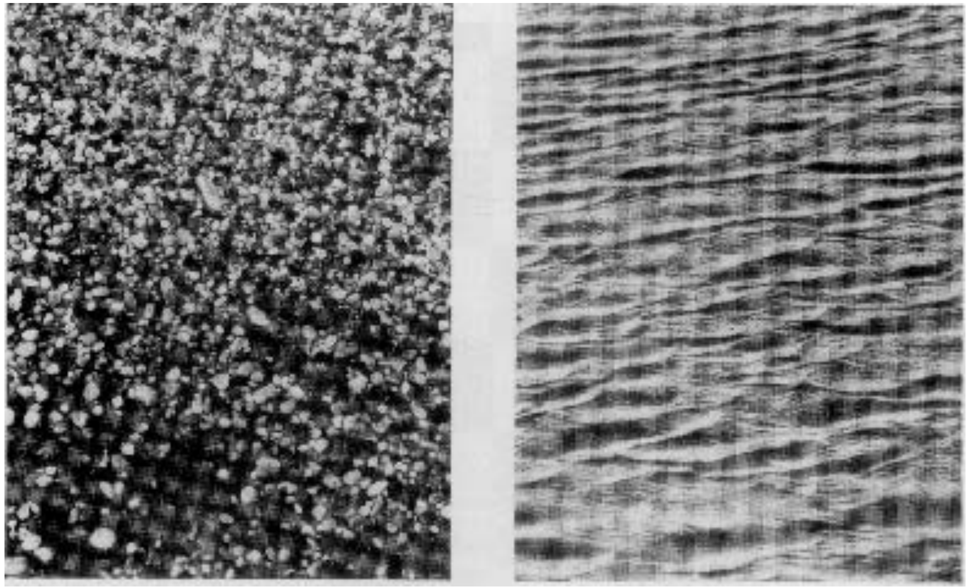




Repetitive Texture Elements (Texels)

A texel represents the smallest graphical element in a two-dimensional texture that creates the impression of a textured surface.

Statistical Textures



Homogeneous Statistical Properties

Textured vs Smooth

A “featureless” surface can be regarded as the most elementary spatial texture:

- Microstructures define reflectance properties.
- They may be uniform or smoothly varying.

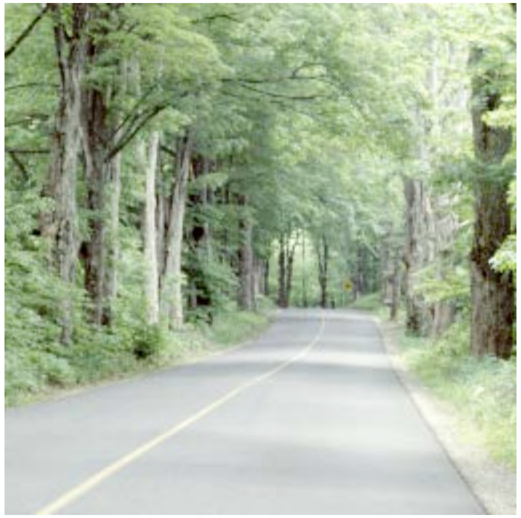
→ Texture is a scale dependent phenomenon

Scale Dependence

At these two different scales, the texture seems very different.

Structural vs Statistical

- Segmenting out texels is difficult or impossible in most real images.



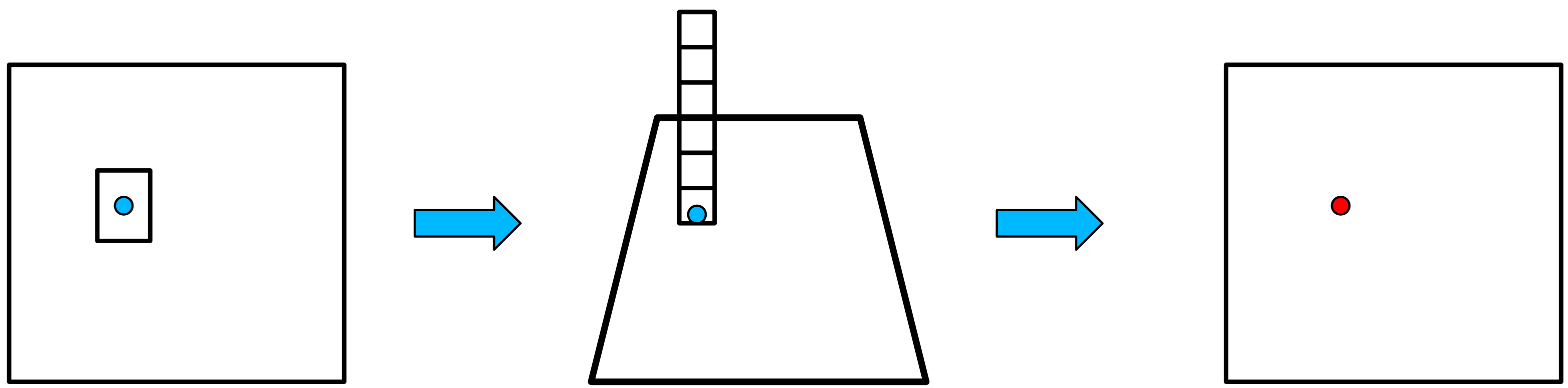
What are the fundamental texture primitives in this image?

- Numeric quantities or statistics that describe a texture can be computed from the gray levels or colors alone.

→ The statistical approach is less intuitive, but more effective in practice.

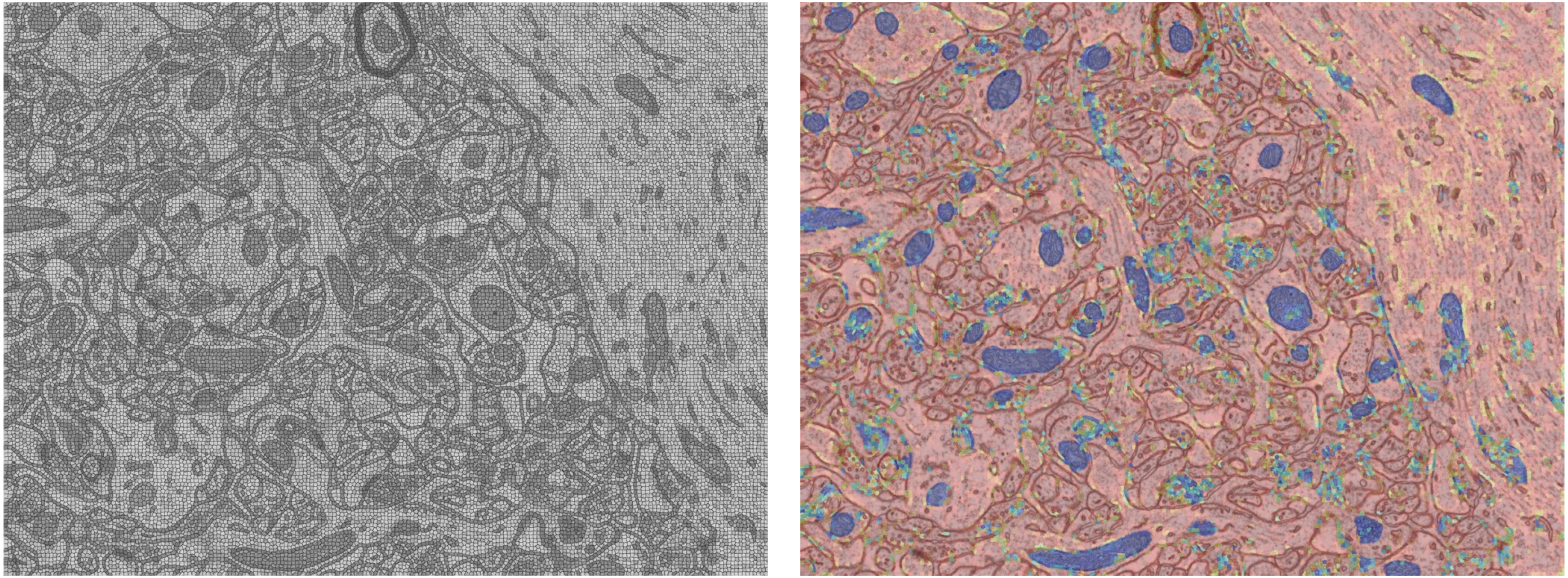
Creating Textural Images

Because texture is non-local, the texture of individual pixels must be estimated using neighborhoods that surround them:



- For each pixel, compute a feature vector using either an image patch or a set of filters.
- Run a classification algorithm to assign a texture value to each pixel.

Reminder: Mitochondria



- Compute image statistics for each superpixel.
- Train a classifier to assign a probability to be within a mitochondria.

—> We used the super pixels to compute local statistics.

Textural Metrics

Spectral metrics:

- Texture is characterized by the properties of its Fourier transform.

Statistical Metrics:

- Texture is as statistical property of the pixels' intensity and color in a region.

Deep Net Metrics:

- They have now mostly superseded the others.
- They encompass the earlier concepts.

Discrete Fourier Transform

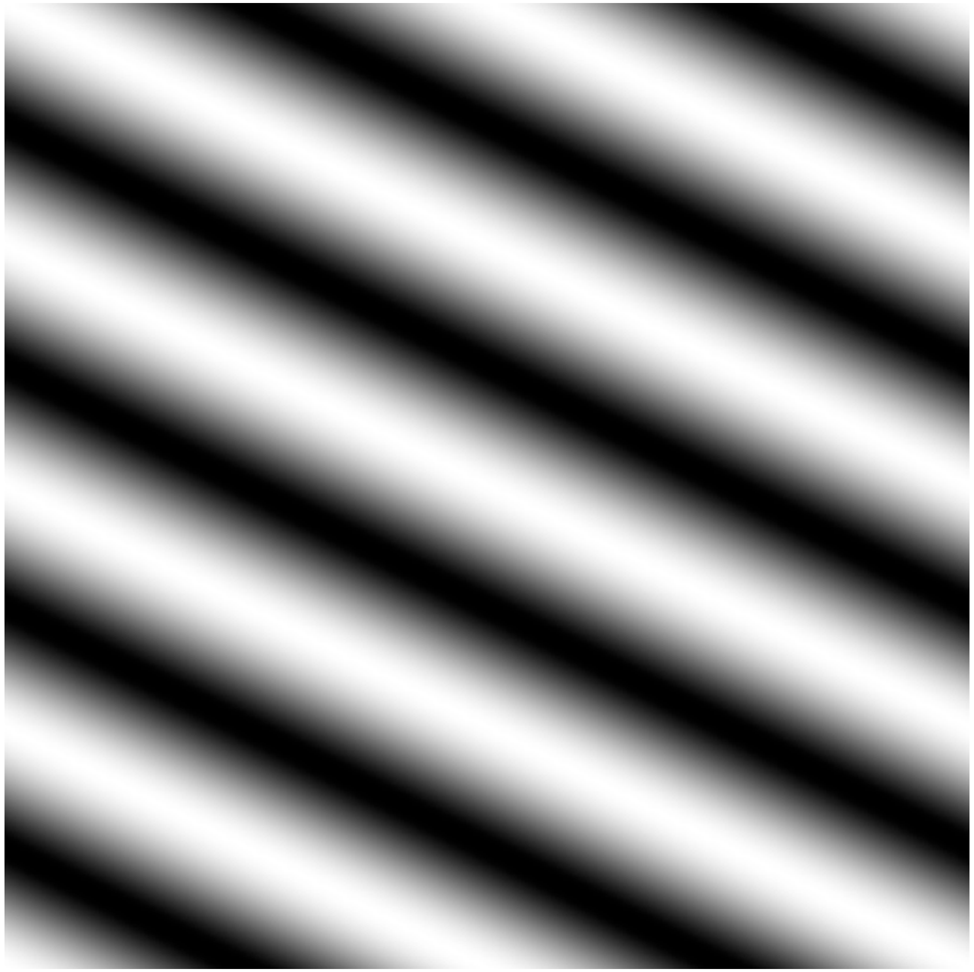
$$F(\mu,\nu) = \frac{1}{\sqrt{M*N}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2i\pi(\mu x/M + \nu y/N)}$$

$$f(x,y) = \frac{1}{\sqrt{M*N}} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} F(\mu,\nu) e^{+2i\pi(\mu x/M + \nu y/N)}$$

The DFT is the discrete equivalent of the 2D Fourier transform:

- The 2D function f is written as a sum of sinusoids.
- The DFT of f convolved with g is the product of their DFTs.

Fourier Basis Element



Real part of

$$e^{+2i\pi(ux+vy)}$$

where

- $\sqrt{u^2 + v^2}$ represents the frequency,
- $\text{atan}(v, u)$ represents the orientation.

Fourier Basis Element

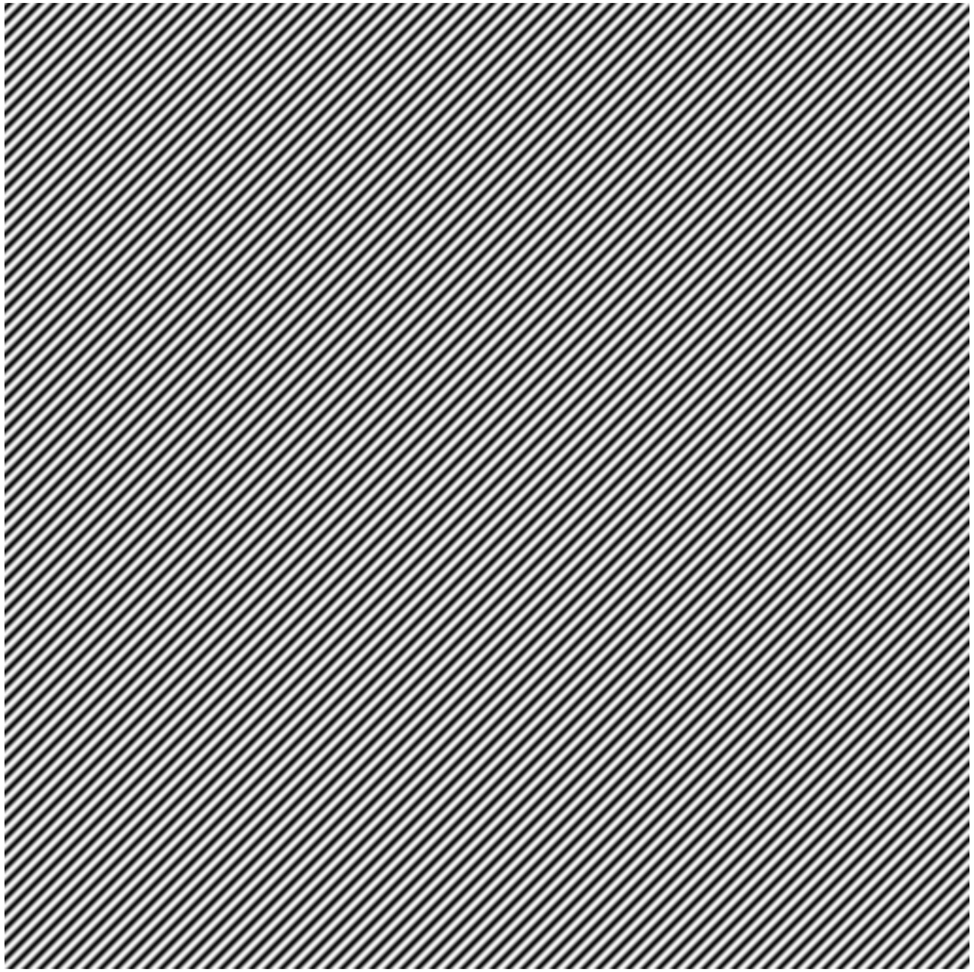
Real part of

$$e^{+2i\pi(ux+vy)}$$

where

- $\sqrt{u^2 + v^2}$ is larger than before.

Fourier Basis Element



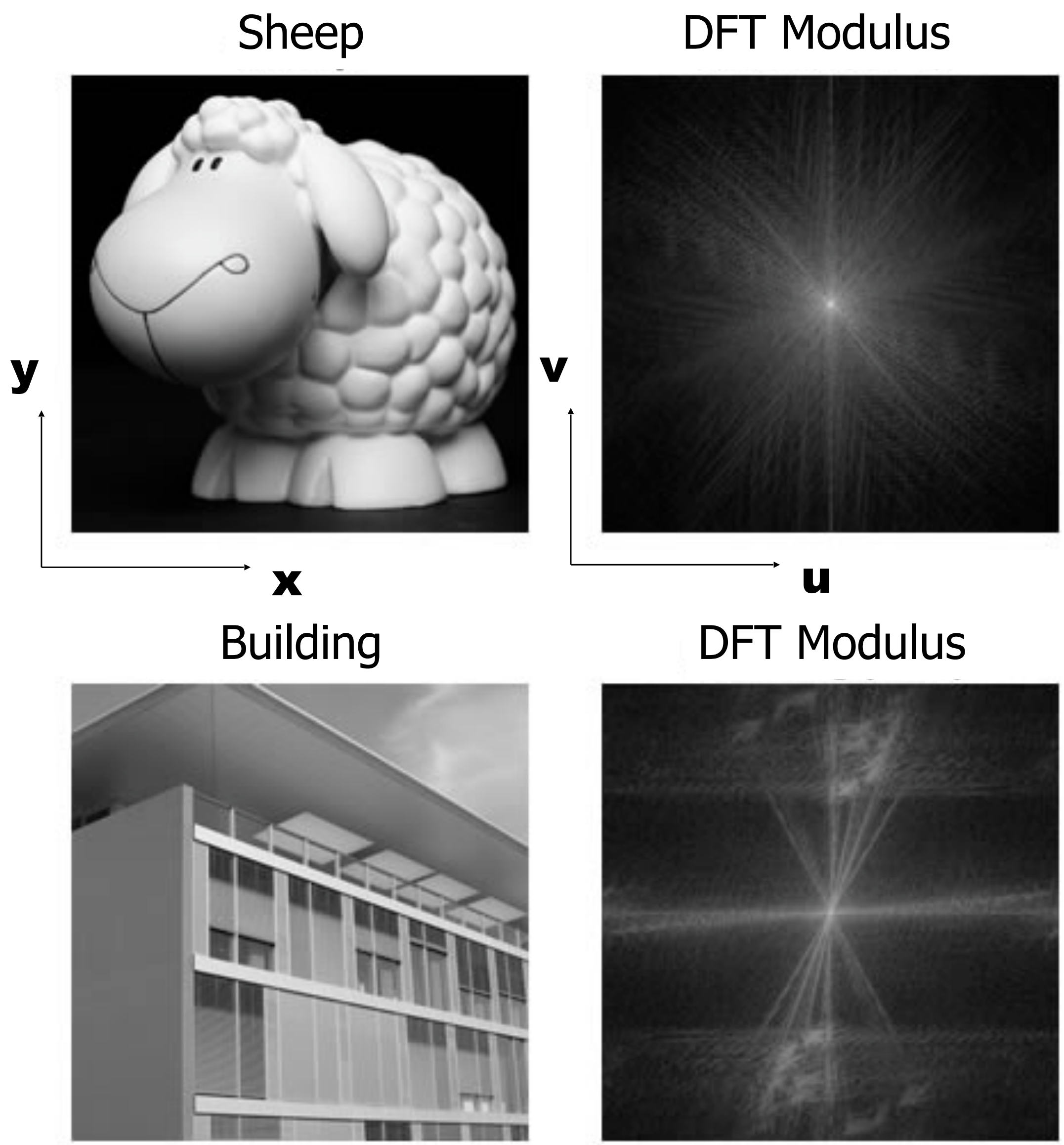
Real part of

$$e^{+2i\pi(ux+vy)}$$

where

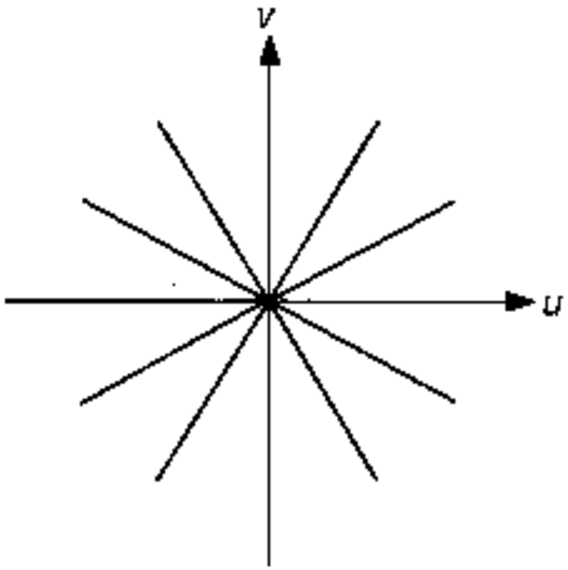
- $\sqrt{u^2 + v^2}$ is larger still.

Spectral Analysis

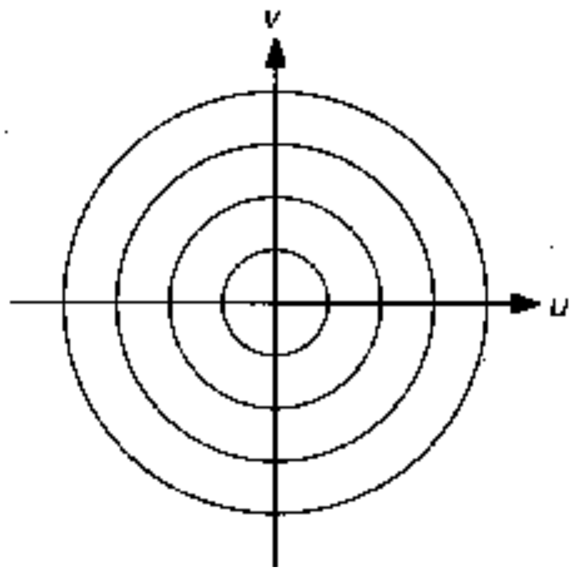


Lines in the DFT modulus images capture the main orientations in the image.

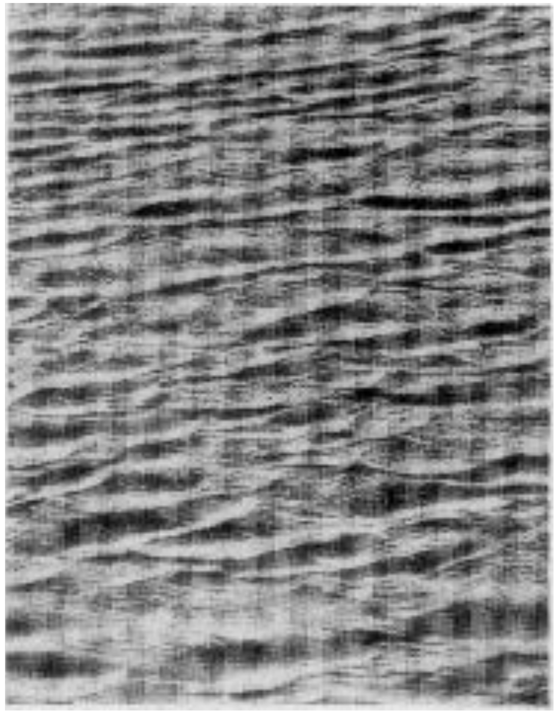
Texture Analysis



Angular bins
in |DFT| image

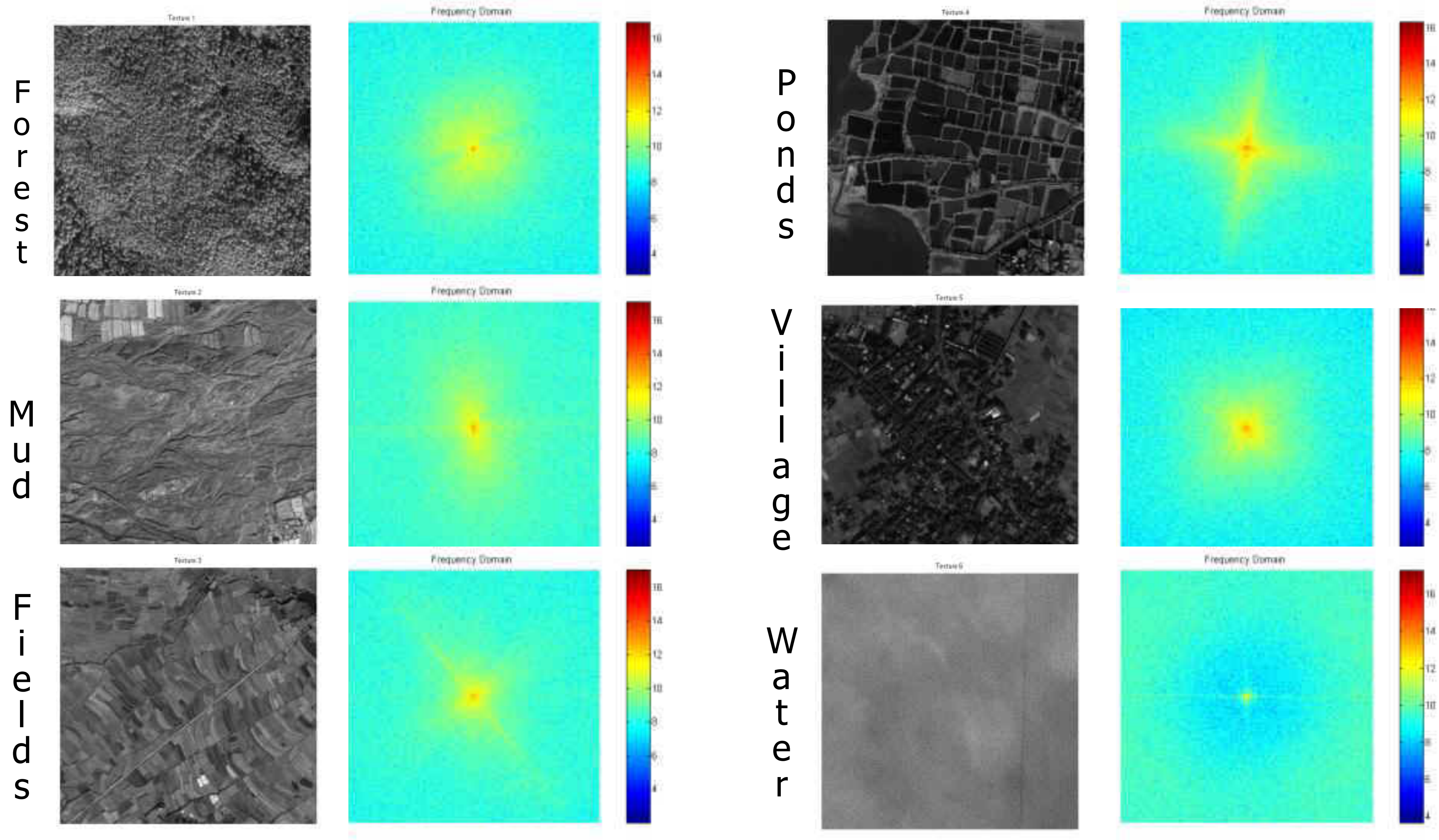


Radial bins
in |DFT| image



Angular and radial bins in the Fourier domain capture the directionality and fluctuation speed of an image texture, respectively.

Fourier Texture Classification



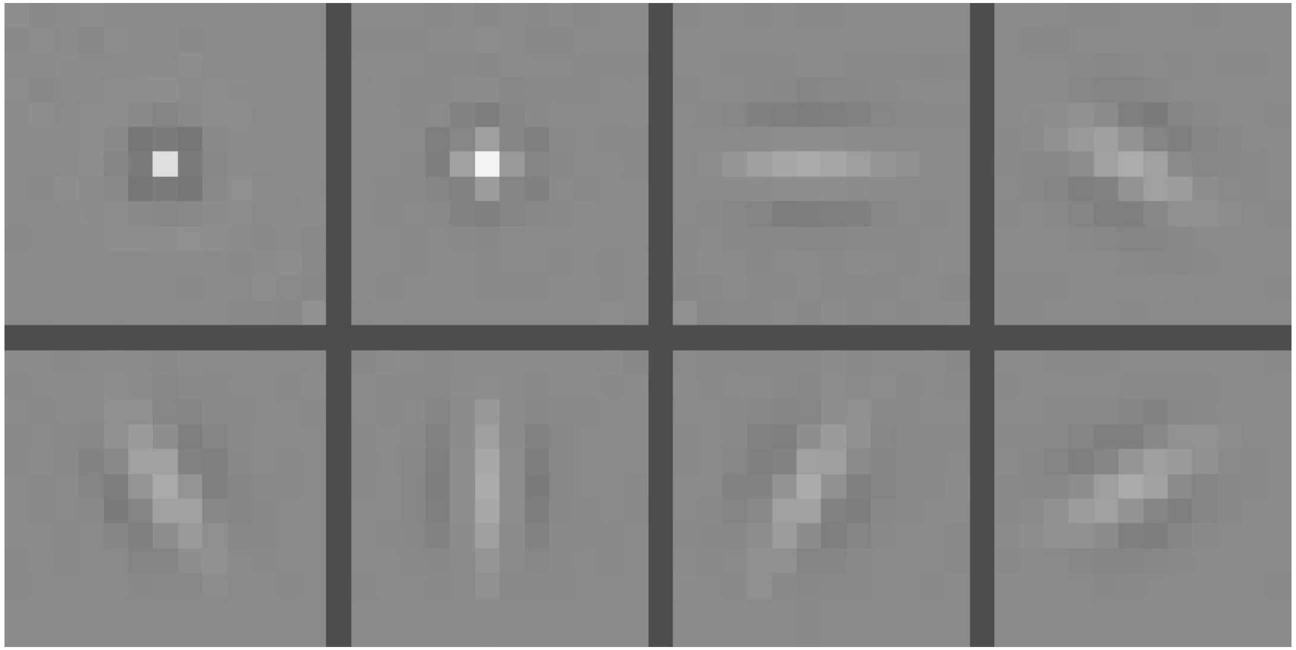
- For some types of textures, the Fourier spectra are easily distinguishable.
- A classifier can be trained to tell them appart.
- However, one must have the same texture in the whole image patch.

Limitations

- DFT on small patches is subject to severe boundary effects.
- Only applicable if texture is uniform over large areas.
- Results can be improved by using wavelets instead, but only up to a point.

—> More local metrics are required.

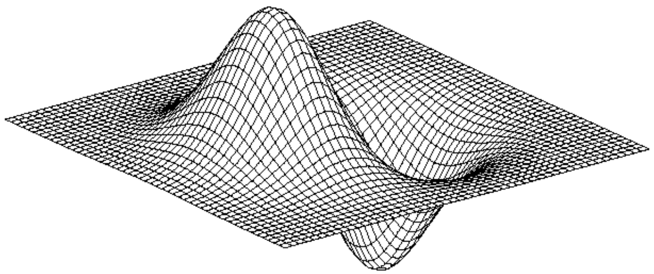
Filter Based Measures



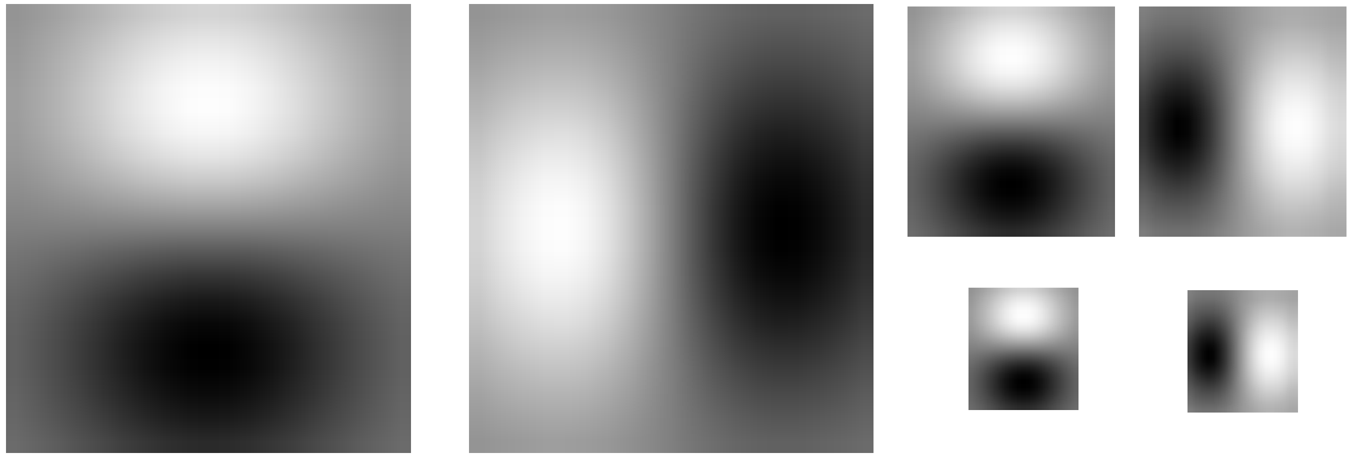
Represent image textures using the responses of a collection of filters.

- An appropriate filter bank will extract useful information such as spots and edges.
- Traditionally one or two spot filters and several oriented bar filters.

Gaussian Filter Derivatives



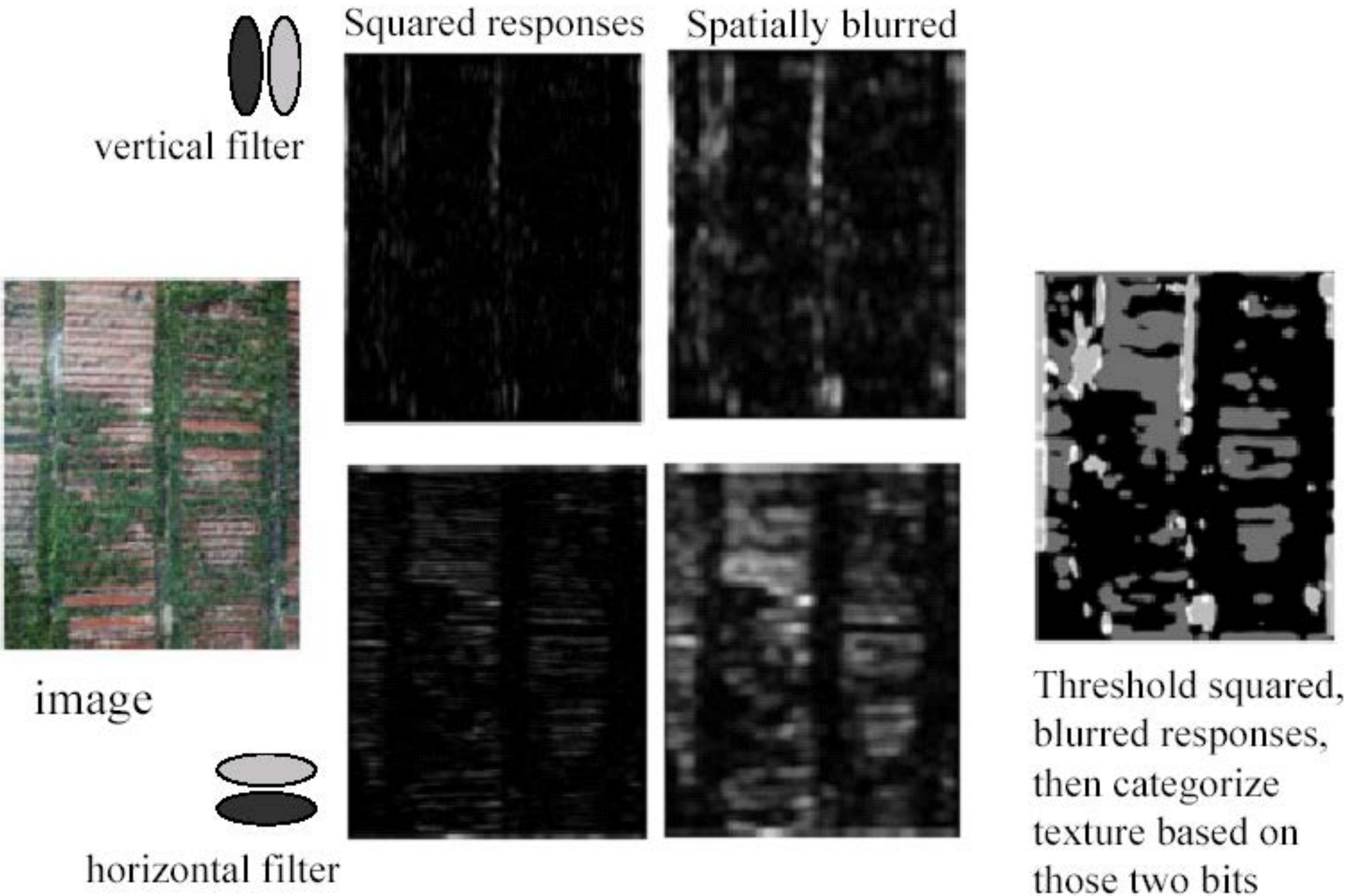
Gaussian Derivative



x and y derivatives at different scales

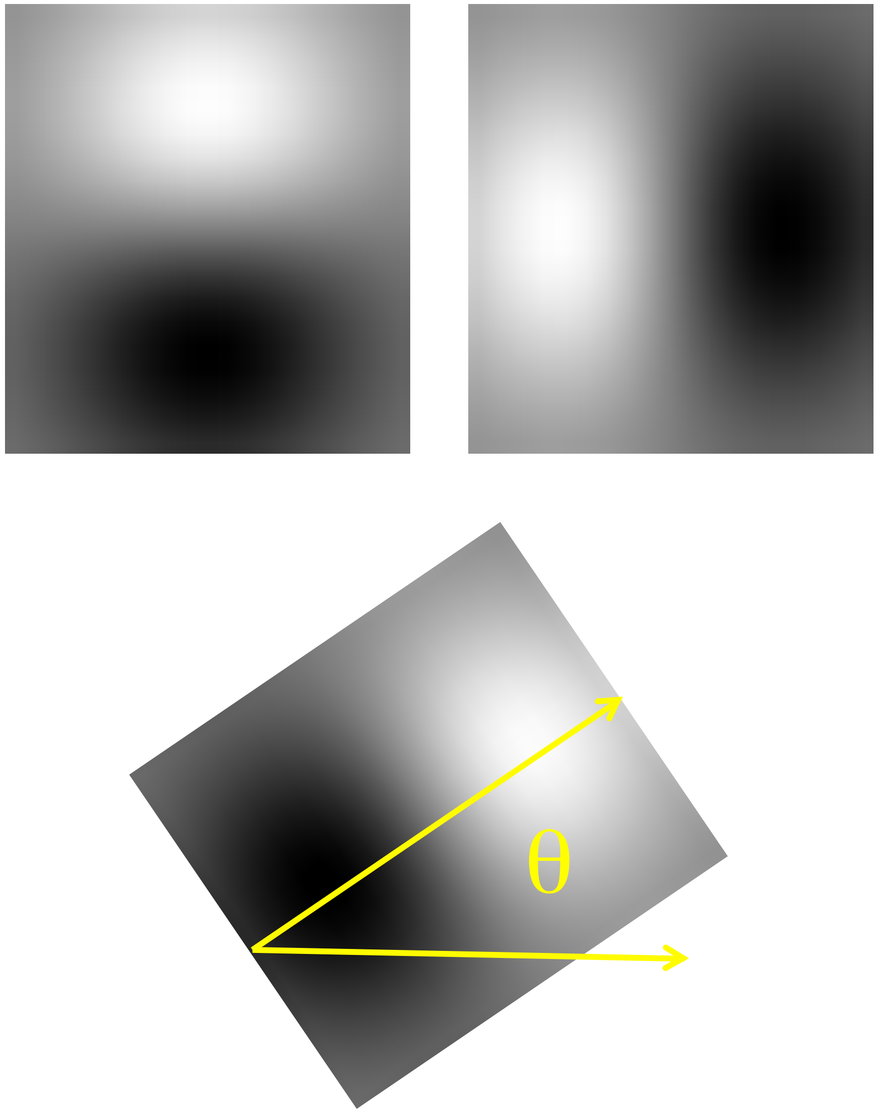
These filters respond to horizontal and vertical edges.

Horizontal and Vertical Structures

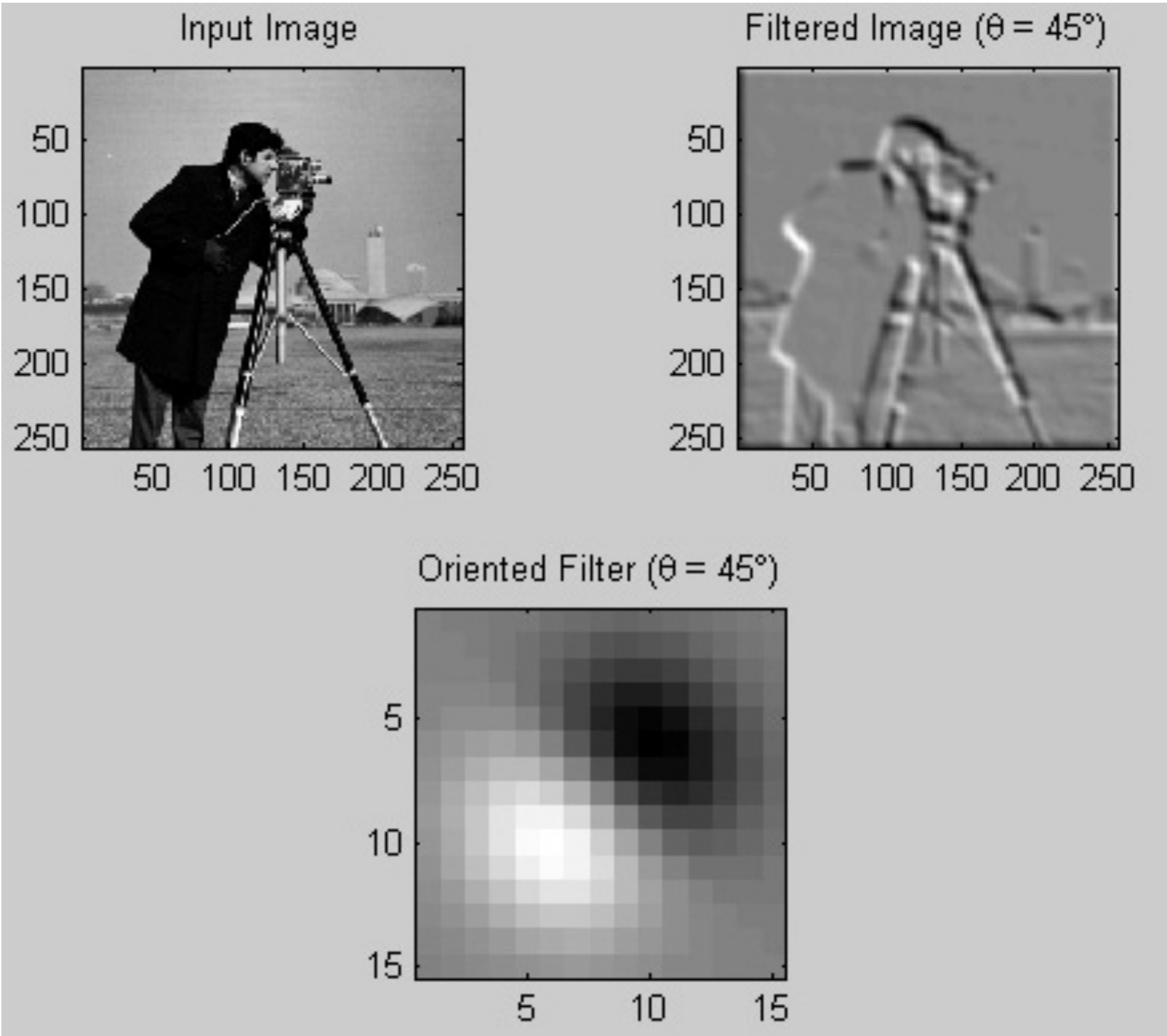


Oriented Filters

$$\frac{\partial I}{\partial \theta} = cos(\theta)\frac{\partial I}{\partial x} + sin(\theta)\frac{\partial I}{\partial y}$$

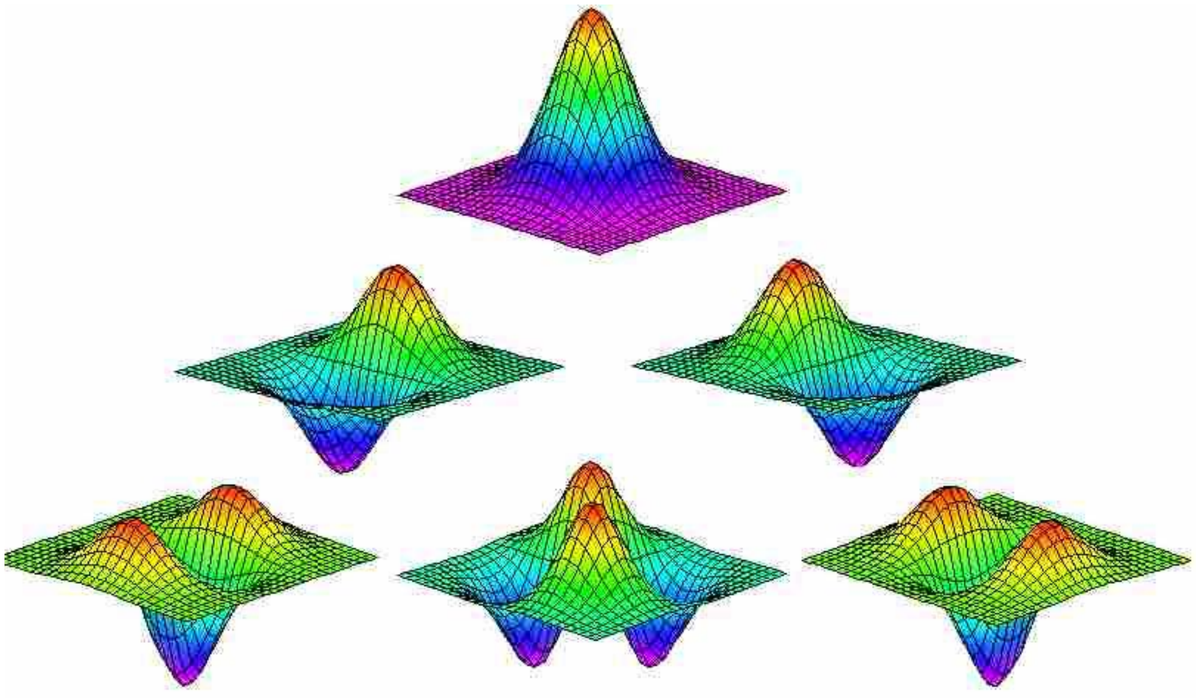


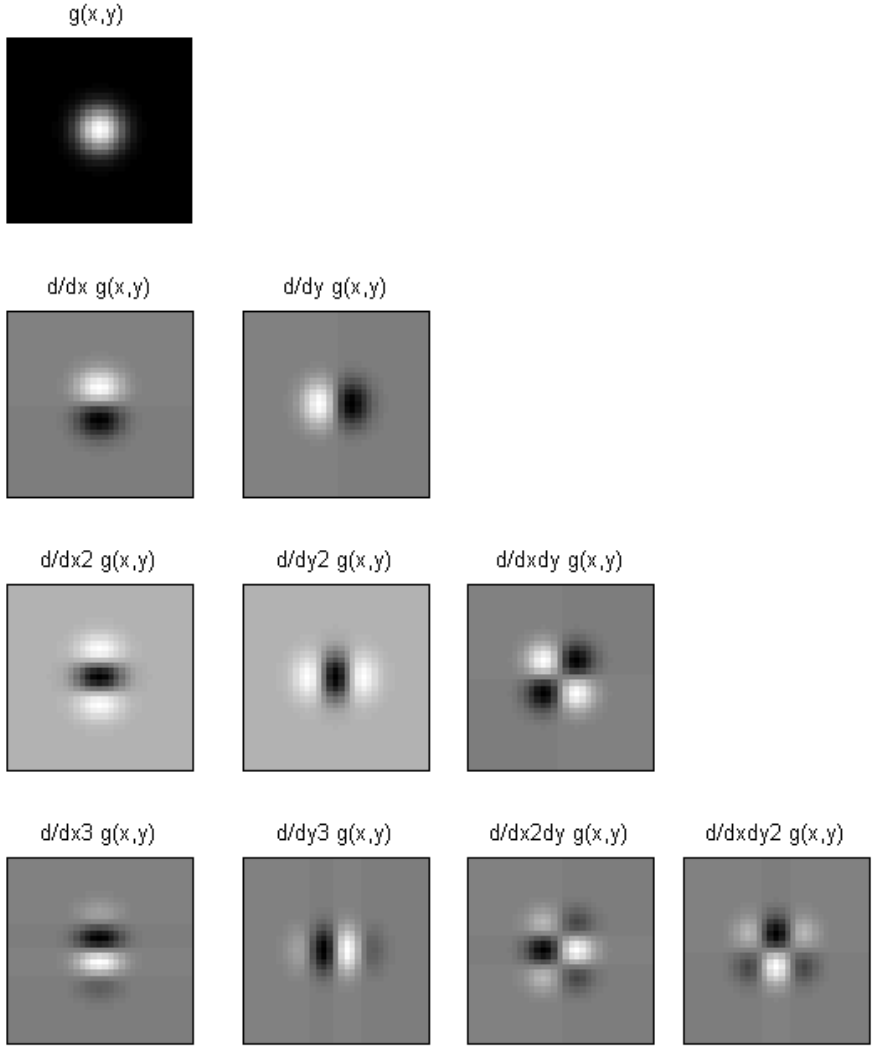
Directional Gradients



Oriented filters respond to edges in a specific direction.

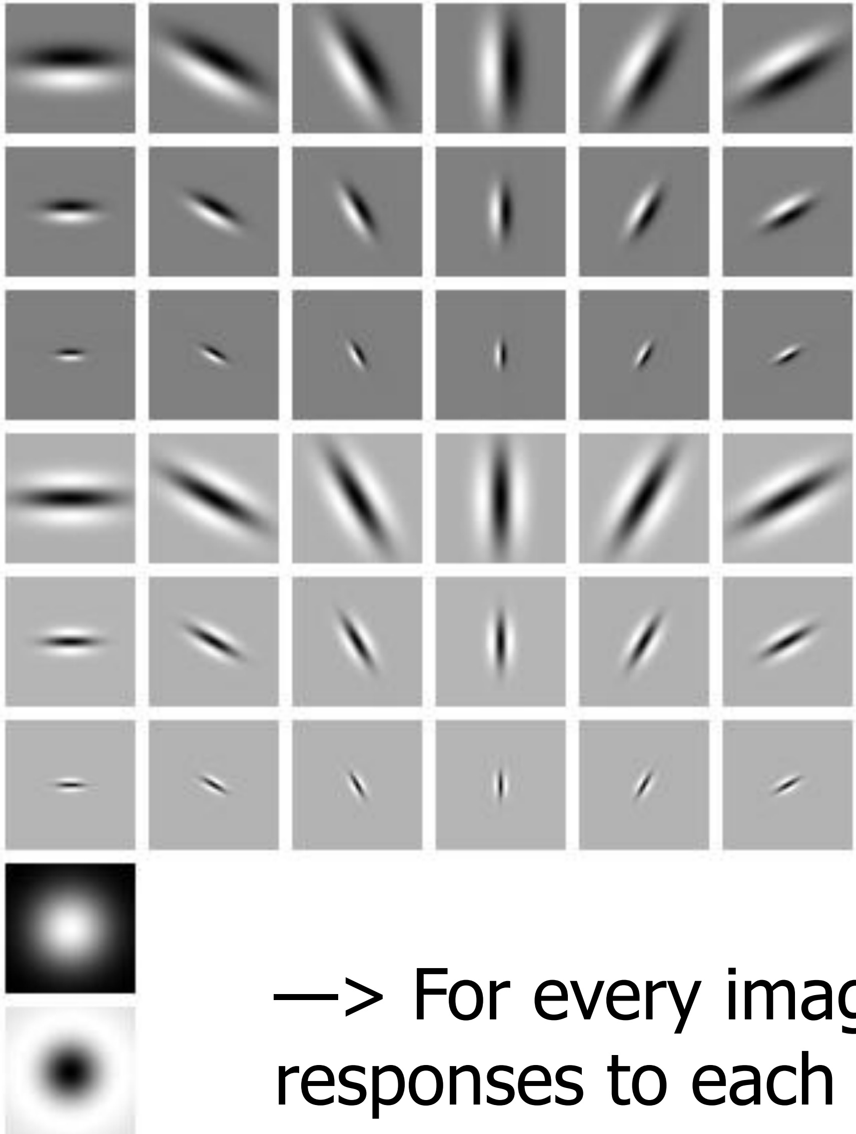
Higher Order Derivatives





Higher-order derivatives of the Gaussian filters can be used to compute higher-order image derivatives.

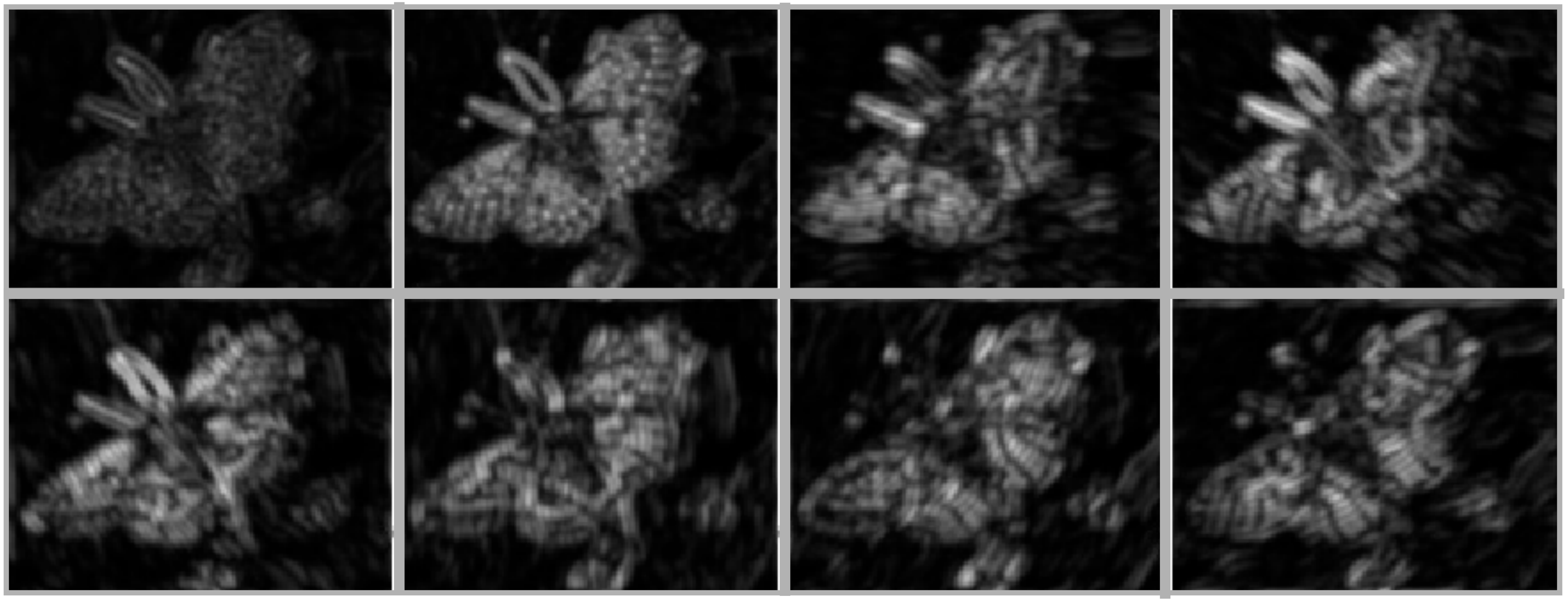
Filter Bank



- Different scales.
- Different orientations.
- Derivatives order 0, 1, 2 ..

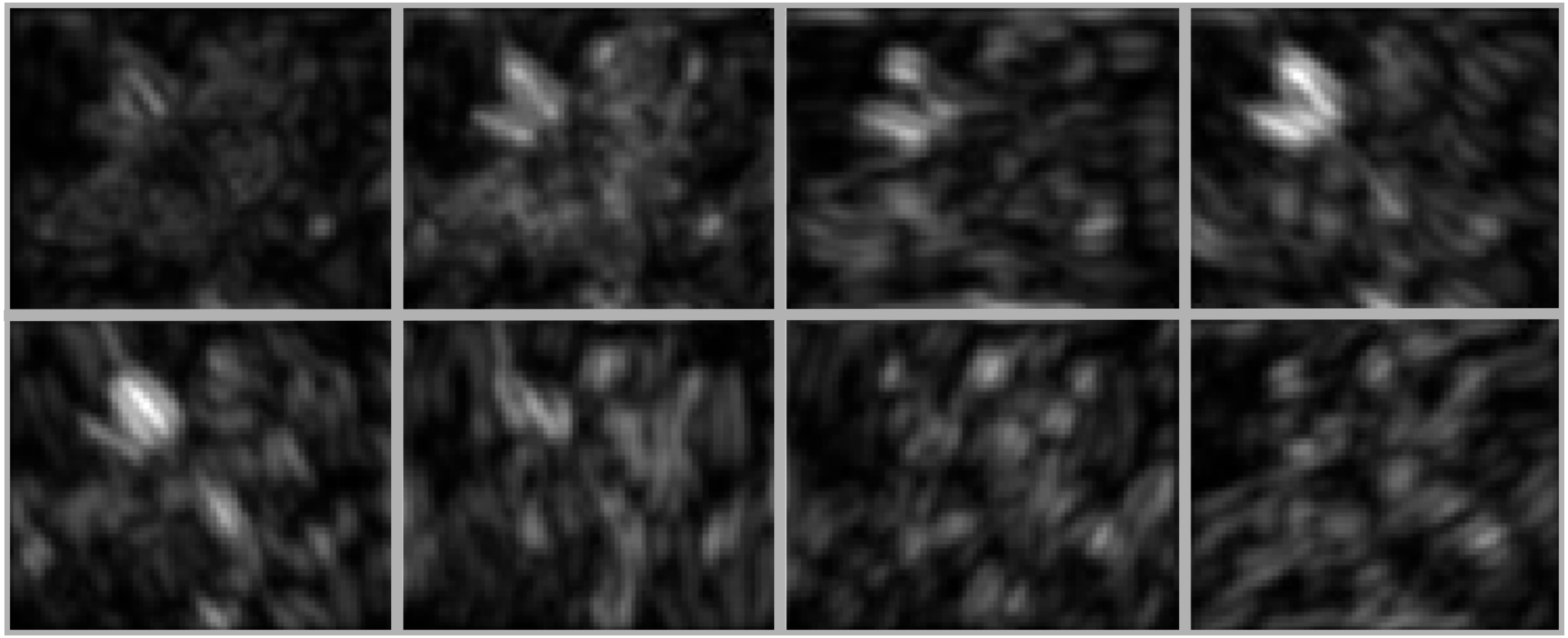
—> For every image pixel, compute a vector of responses to each filter.

Filter Responses: Small Scales



Gaussian filters with a small σ. Capture local details.

Filter Responses: Large Scales



Gaussian filters with a large σ. Capture larger details.

Gabor Filters

Gabor filters are the products of a Gaussian filter with oriented sinusoids. They come in pairs, each consisting of a symmetric filter and an anti-symmetric filter:

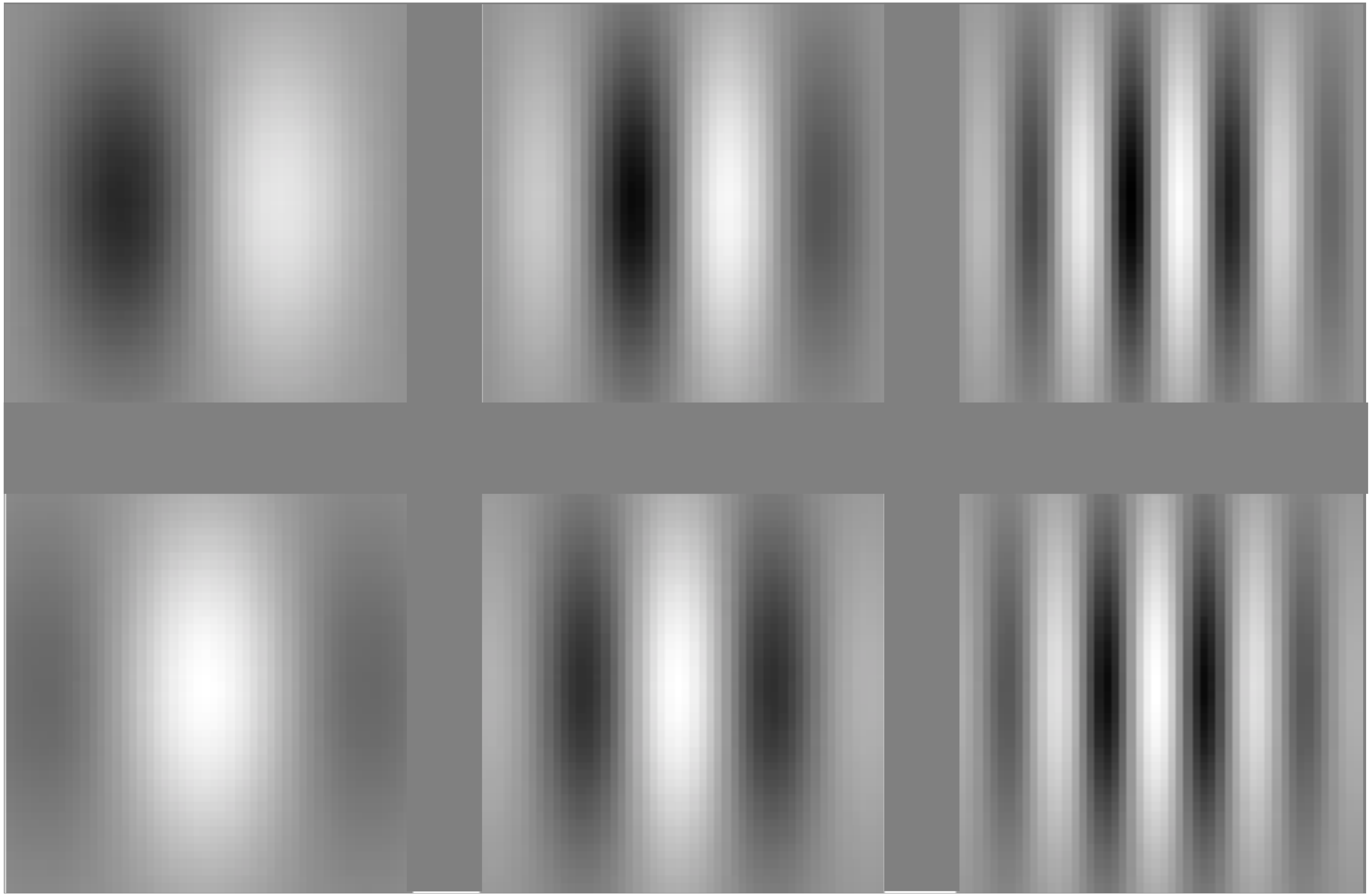
$$G_{\text{sym}}(x,y) = \cos(k_x x + k_y y)\exp(-\frac{x^2+y^2}{2\sigma^2})$$

$$G_{\text{asym}}(x,y) = \sin(k_x x + k_y y)\exp(-\frac{x^2+y^2}{2\sigma^2})$$

where k_x and k_y determine the spatial frequency and the orientation of the filter and σ determines the scale.

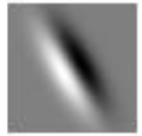
→ A filter bank is formed by varying the frequency, the scale, and the filter orientation

Vertical Derivatives

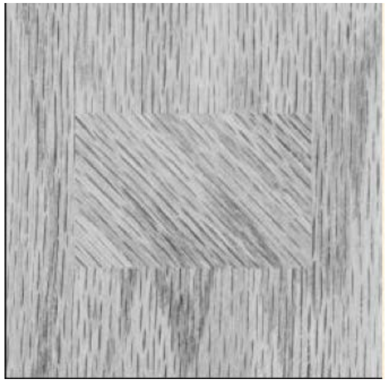


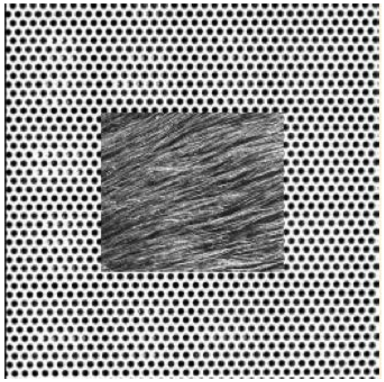
Gabor Responses

Filters:

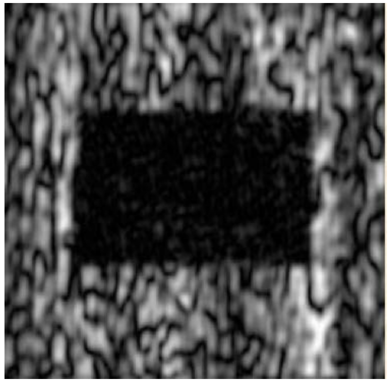


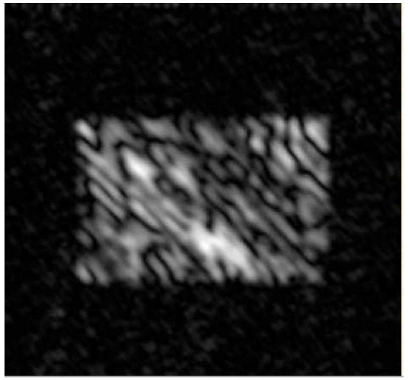
Images:

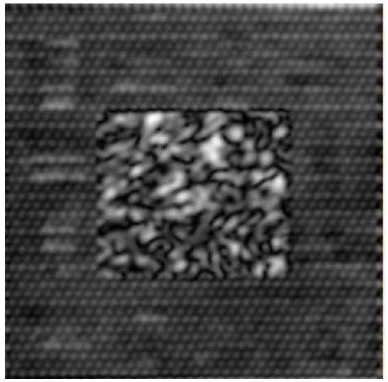


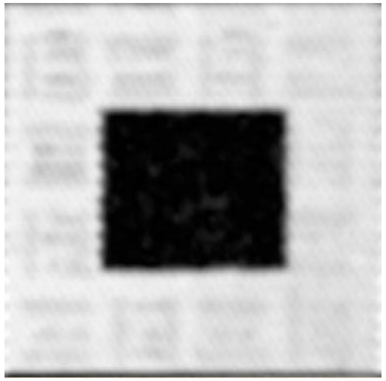


Responses:

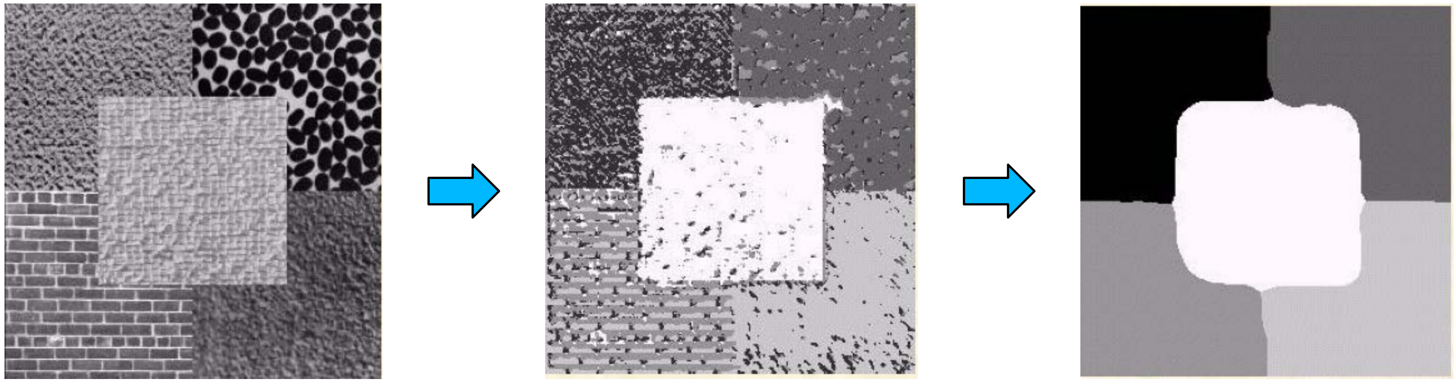








GABOR FILTER CHARACTERISTICS

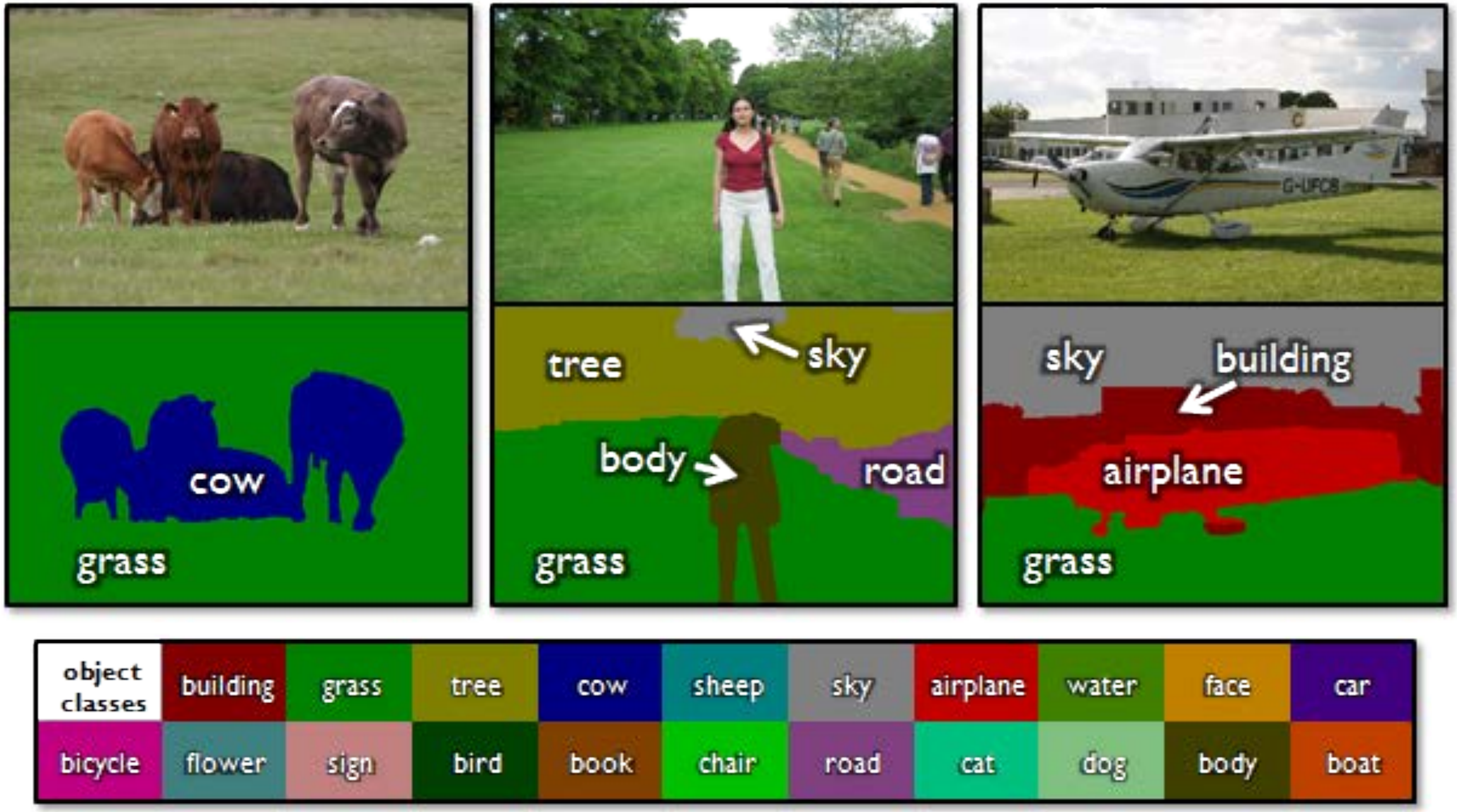


- Respond strongly at points in an image where there are components that locally have a particular spatial frequency and orientation.

- In theory, by applying a very large number of Gabor filters at different scales, orientations and spatial frequencies, one can analyze an image into a detailed local description.

- In practice, it is not known how many filters, at what scale, frequencies, and orientations, to use. This tends to be application dependent.

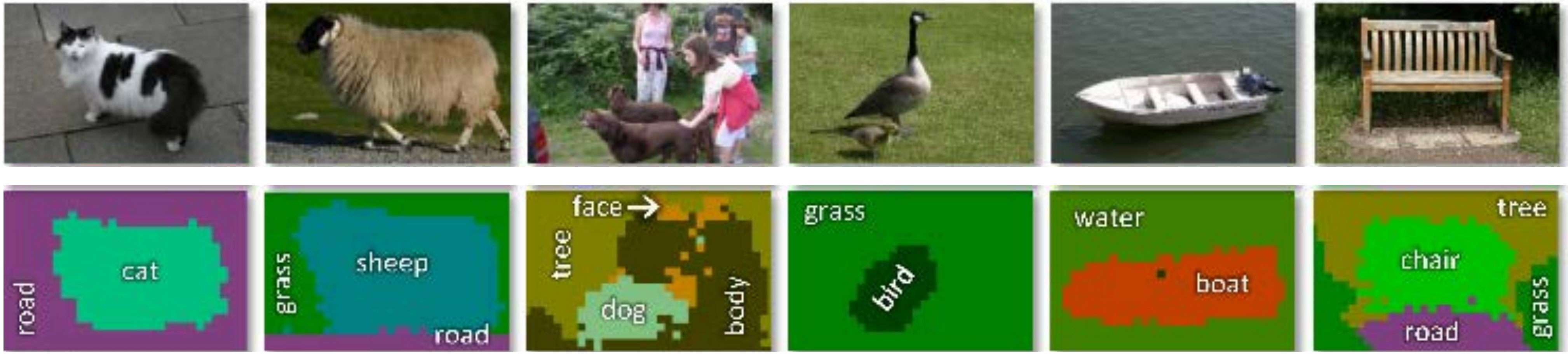
ML to the Rescue: Texton Boost



Use AdaBoost to perform classification on the output of Gabor filters.

Shotton et al., ECCV'06

ML to the Rescue: Texton Forests

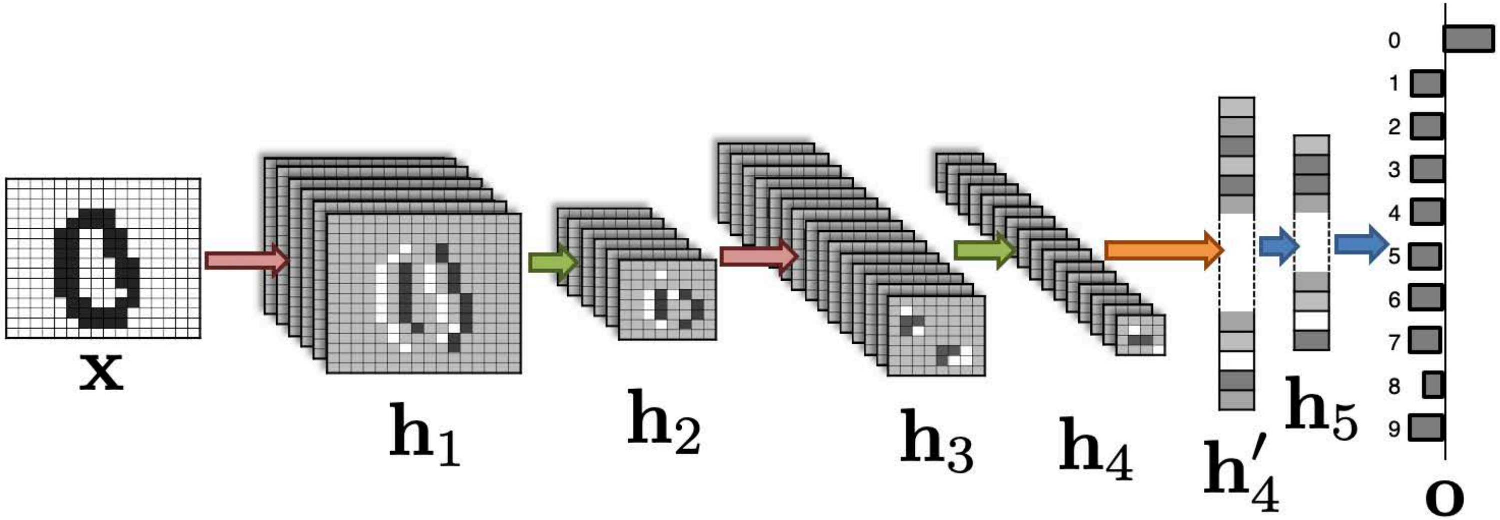


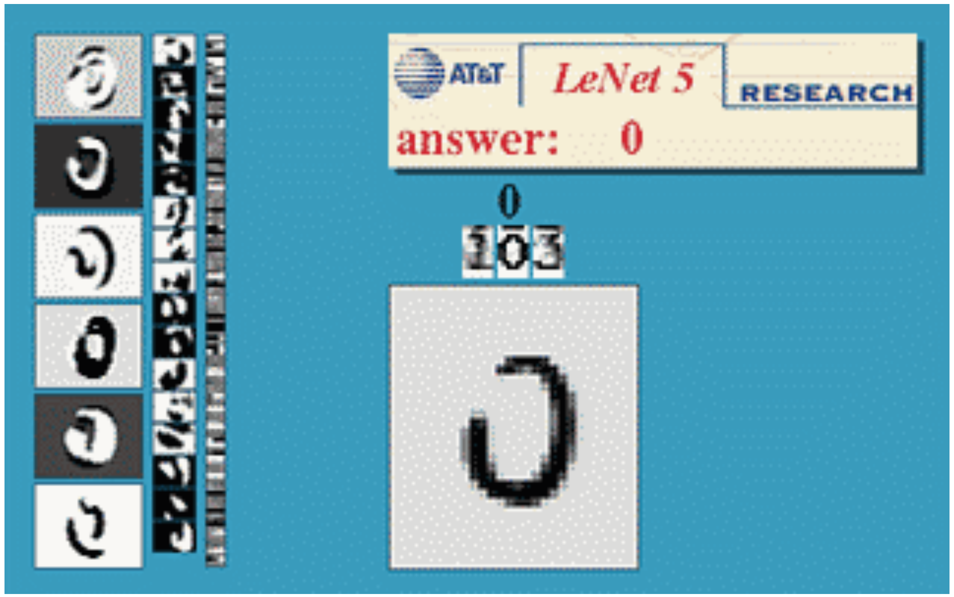
- Using Decision Forests to perform classification on the output of Gabor filters works better in this case.

- But what works even better, is ……

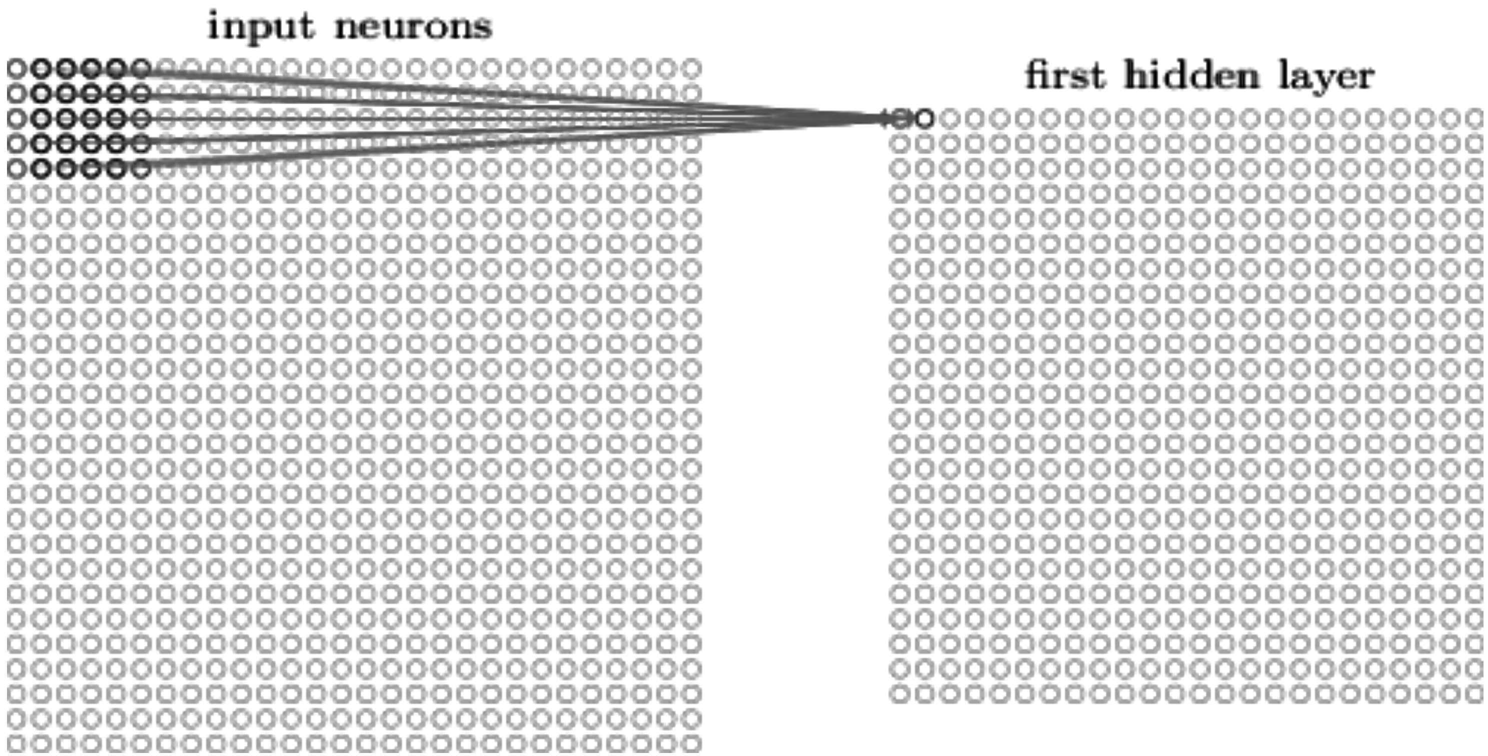
Shotton et al., CVPR'08

Reminder: ConvNets





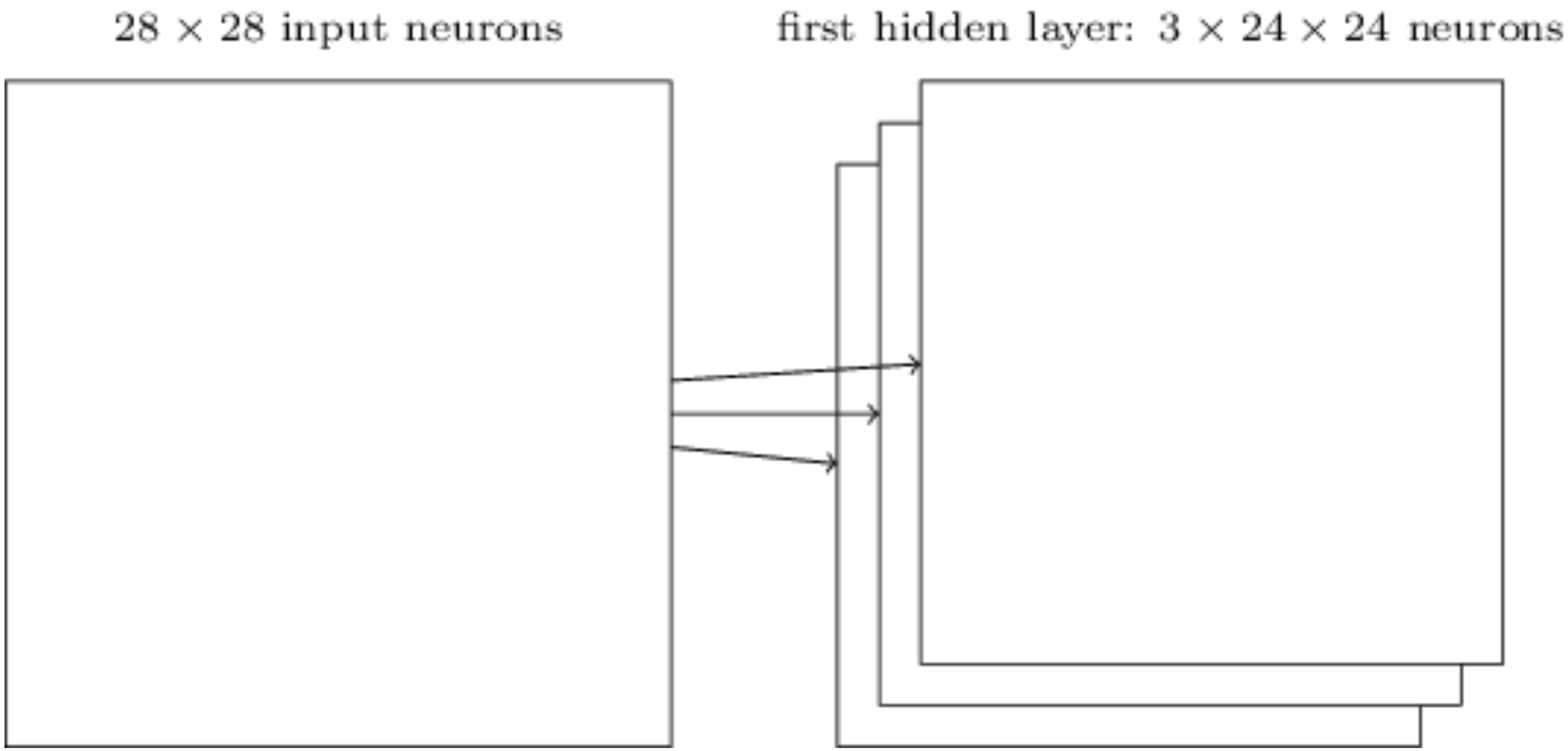
Reminder: Convolutional Layer



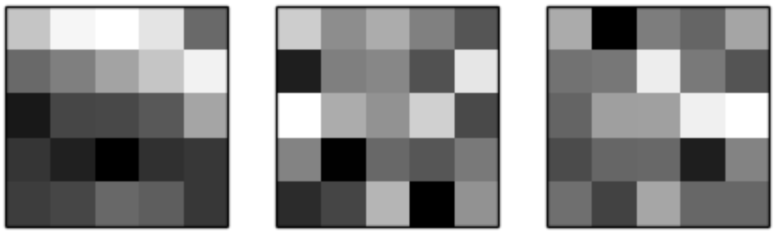
$$\sigma\left(b + \sum_{x=0}^{n_x} \sum_{y=0}^{n_y} w_{x,y} a_{i+x,j+y}\right)$$

EPFL

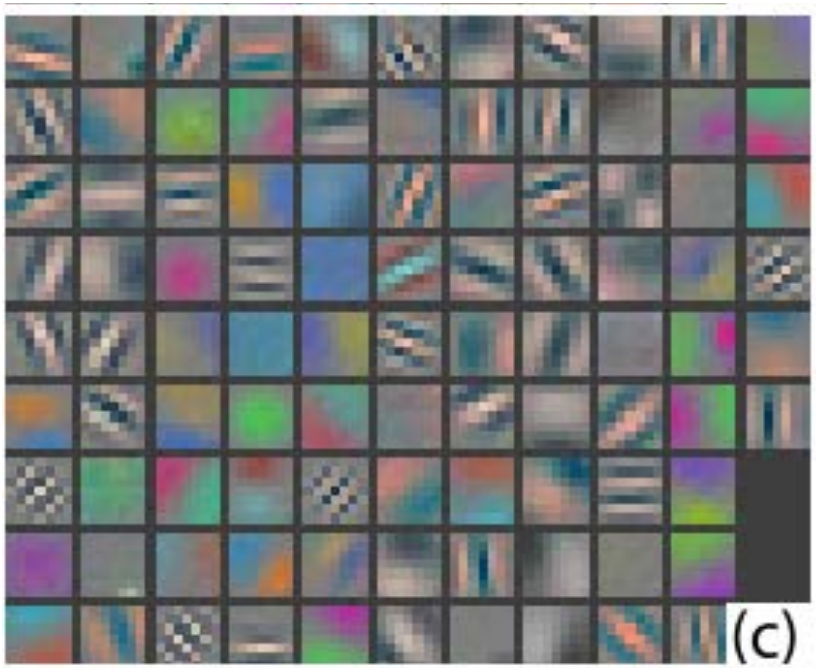
Reminder: Feature Maps

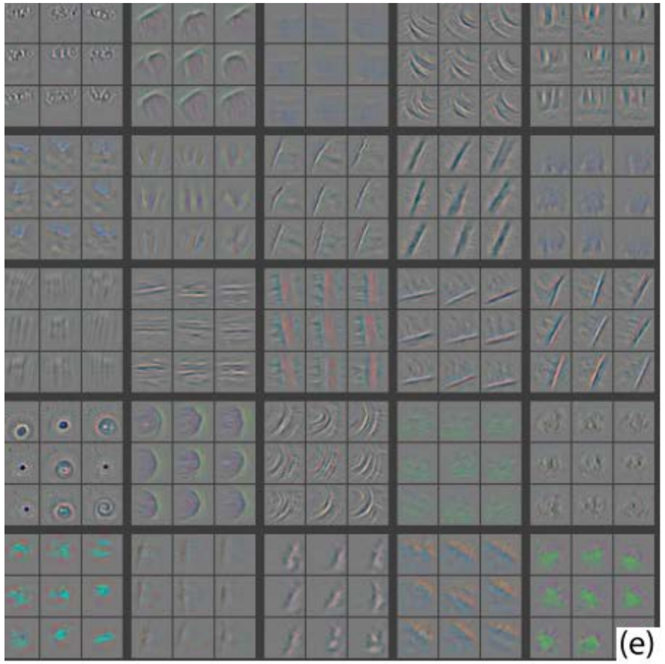


Filters:



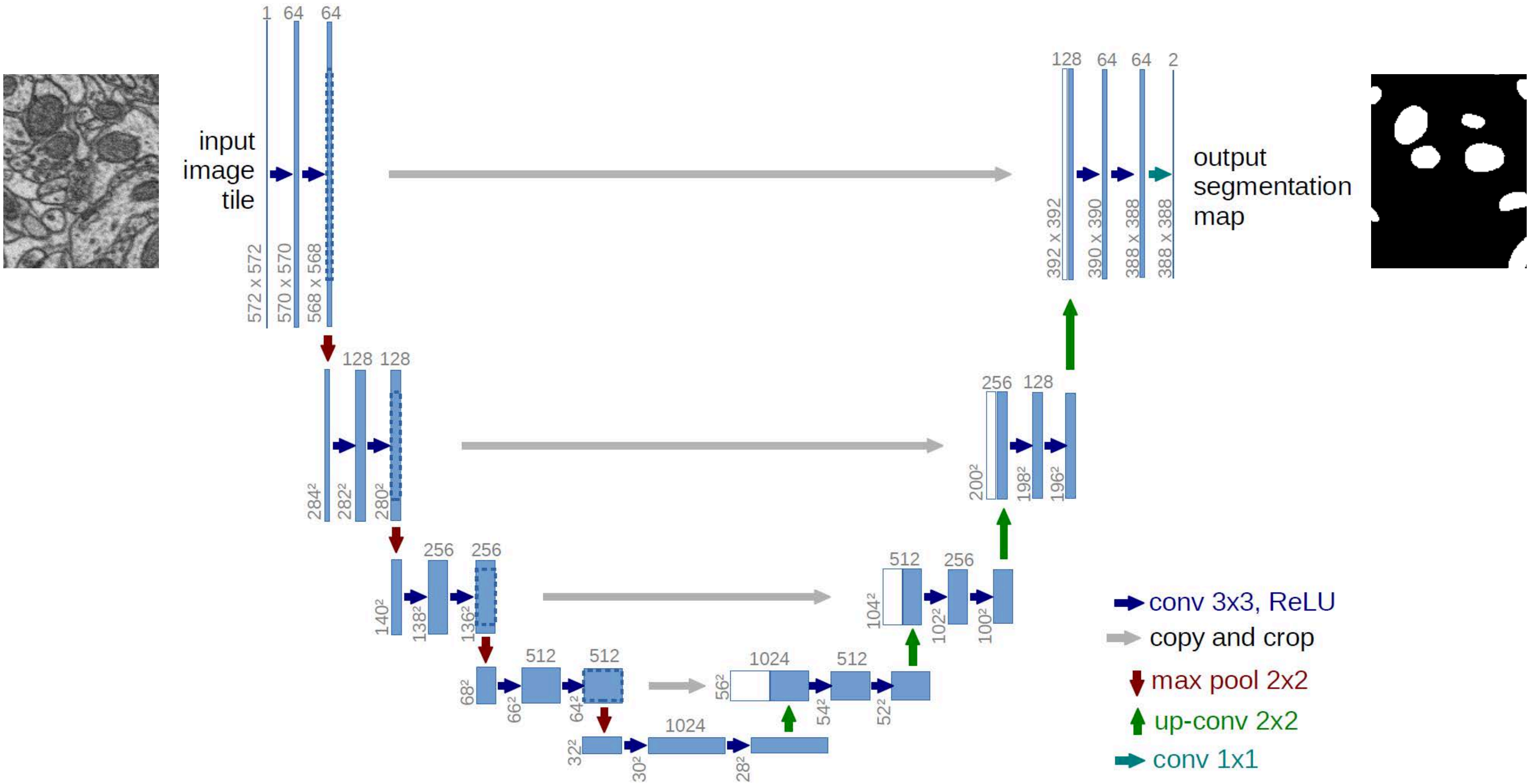
Learned Feature Maps



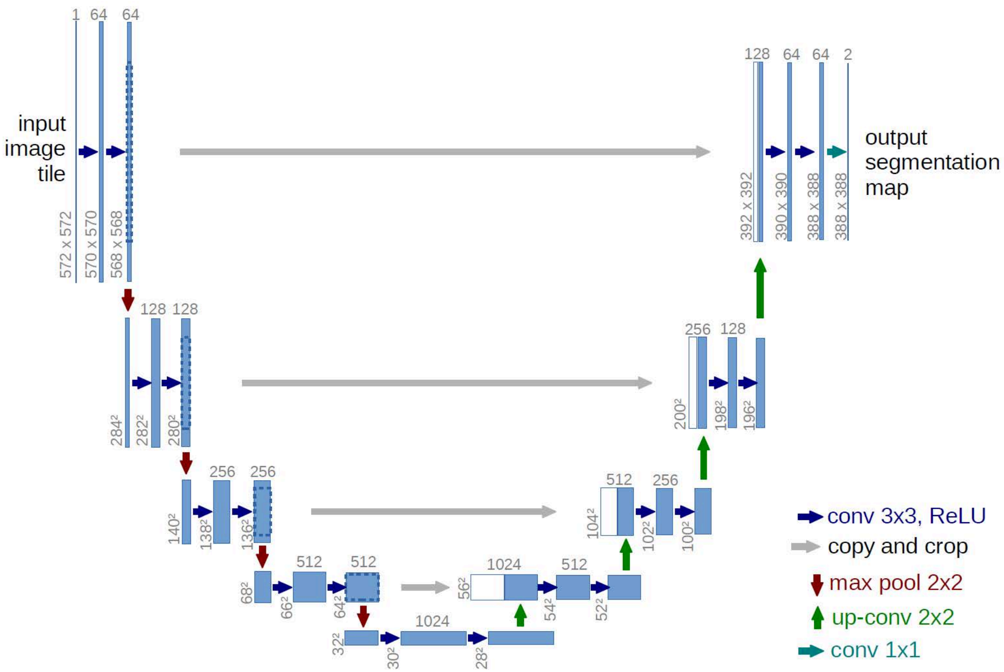


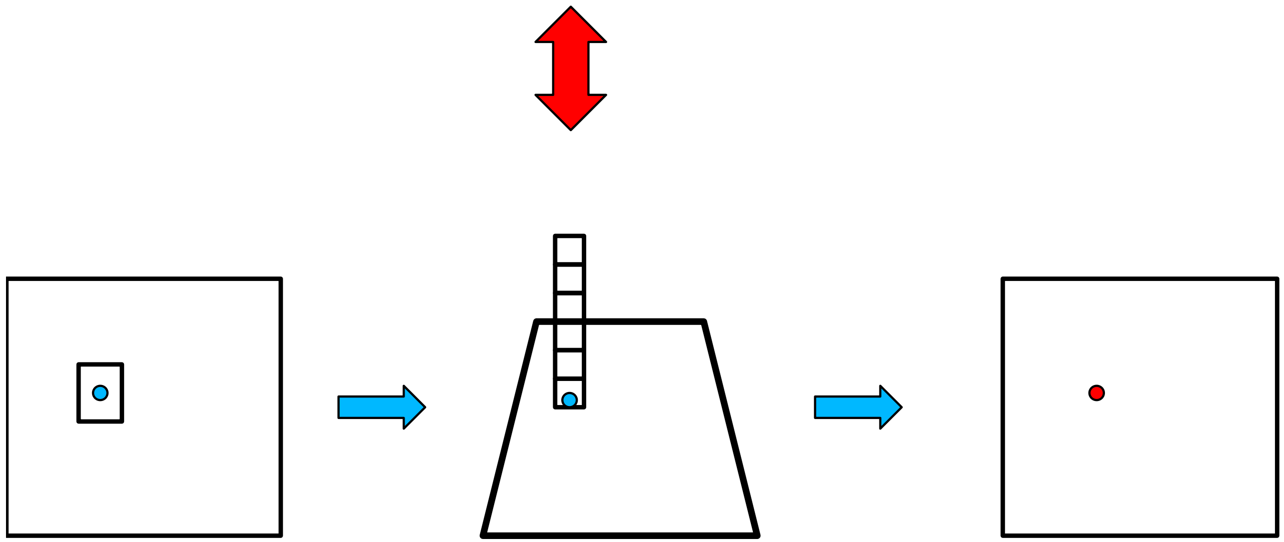
- Some of these convolutional filters look very Gabor like.
- The network requires a large training set to learn an effective filter bank.
- The older techniques still have their place in the absence of such training sets.

Reminder: U-Net Architecture



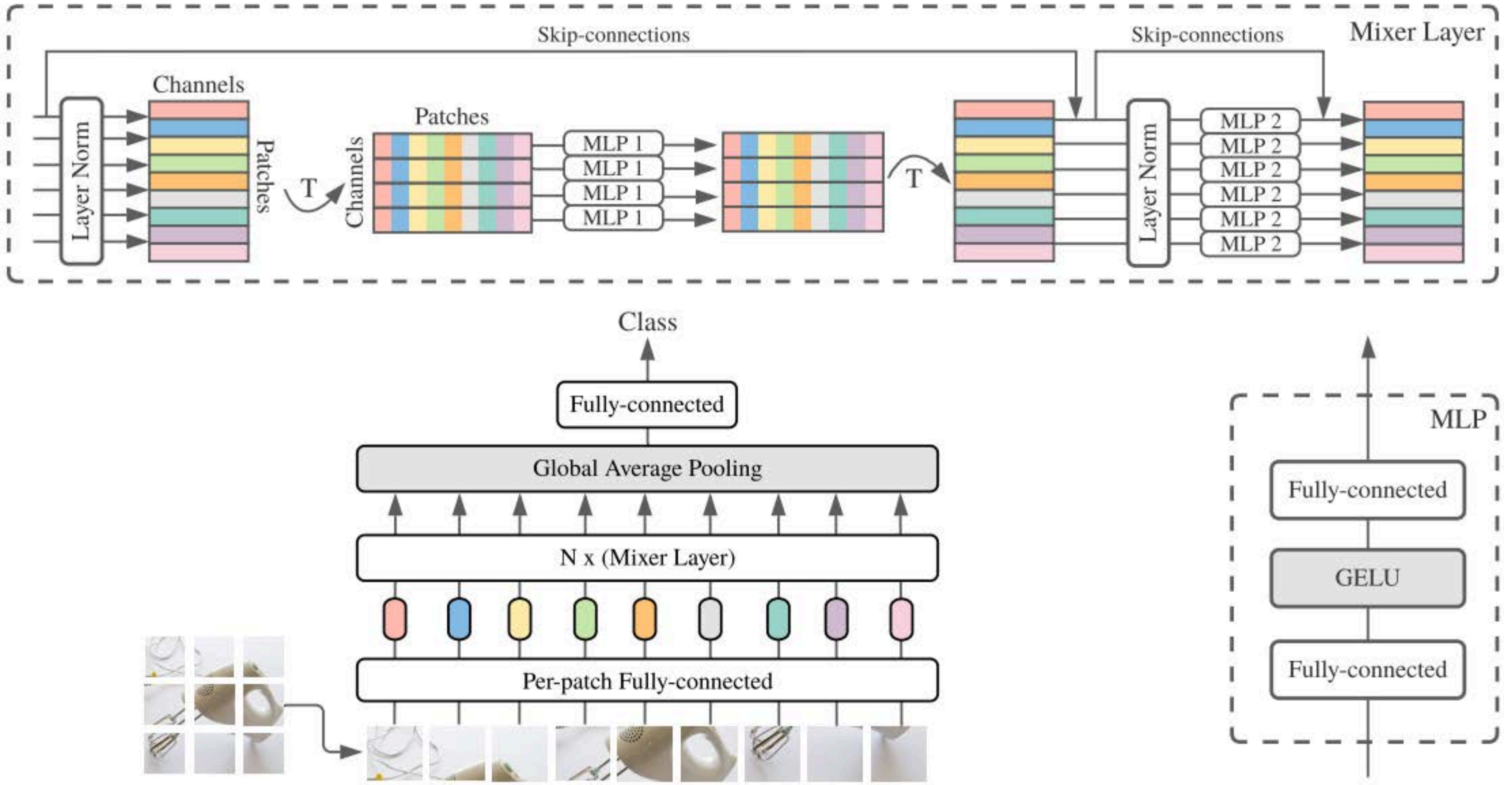
Reminder: Potential Interpretation





A key role of the ConvNet is to generate for every output pixel a feature vector containing the output of all the intermediate layers.

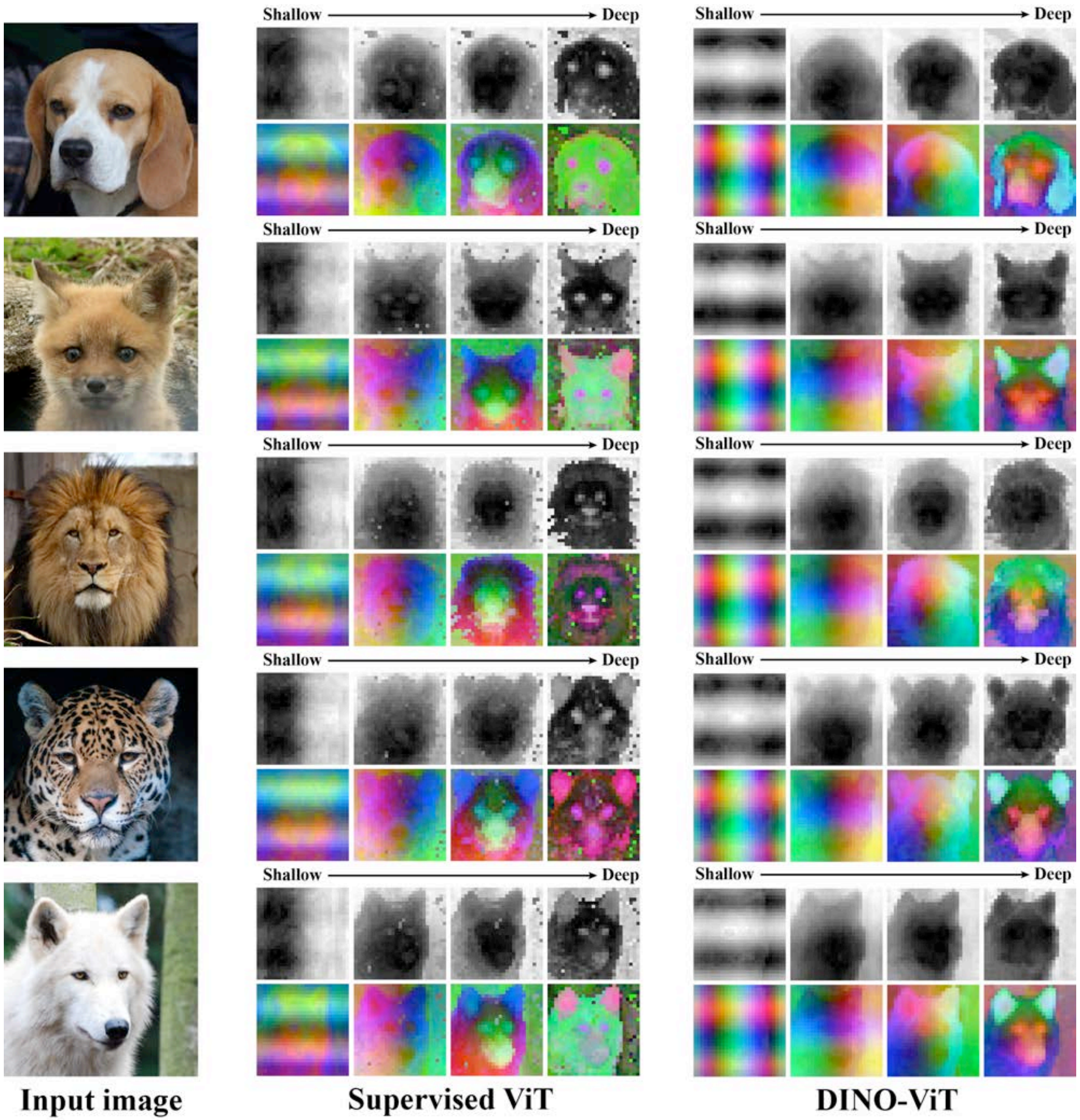
Reminder: Vision Transformers



- Break up the images into square patches.
- Transform each path into a feature vector.
- Feed to a transformer architecture.

Tolstikhin et al. , ArXiv'21

From Interpretation to Practice



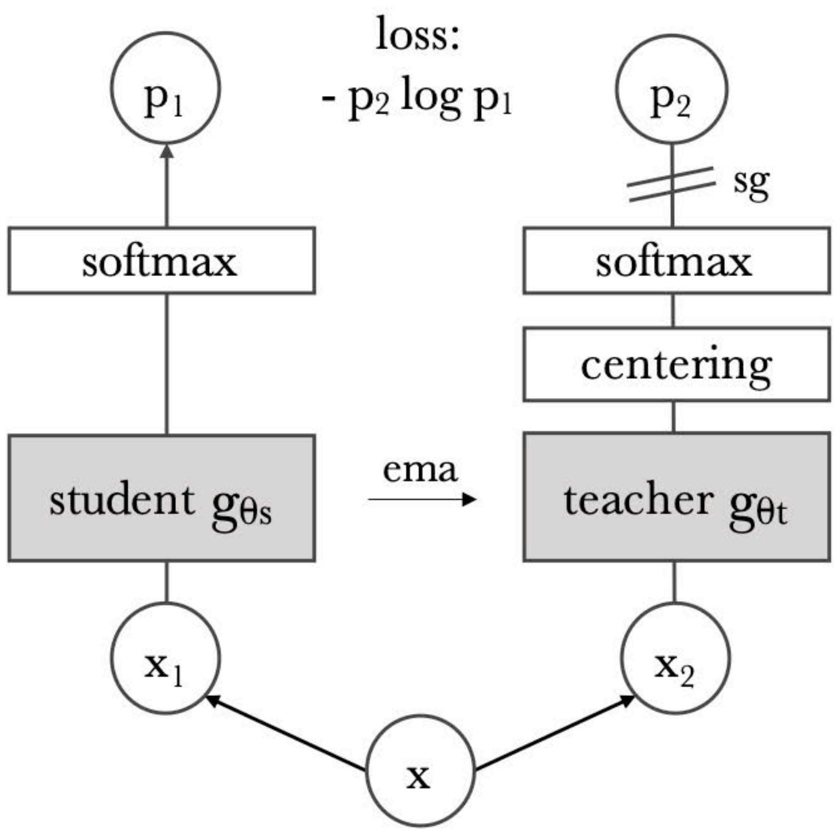
- Run a Vision Transformer on all images.
- Display first four PCA components of the feature vectors at each pixel.
- There is clearly enough information to perform a segmentation.
- But how should the network be trained?

  - Supervised. Possible but risk of bias if the training database is not adapted.

  - Self-Supervised. Potentially more generic.

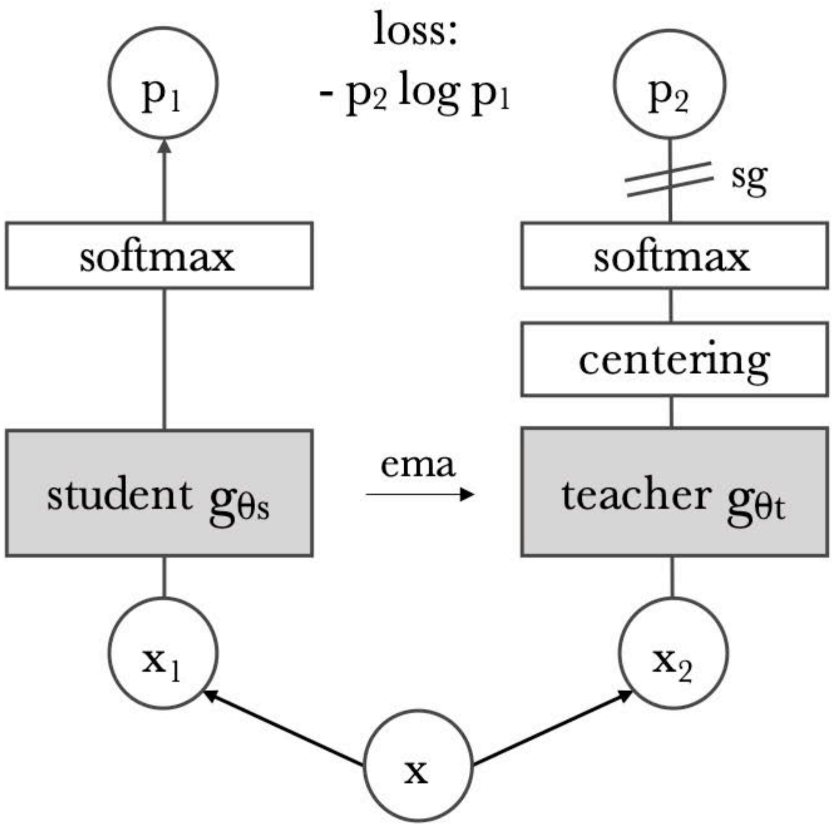
Caron et al. , CVPR'21

Self-Supervised Training



- Two networks—the student and the teacher—with the same architecture but different parameters are trained to output similar results.
- Each networks outputs a K dimensional feature that is normalized with a softmax over the feature dimension.
- Output similarity measured by a cross-entropy loss.
- To break the symmetry
  - Perturb the input to each network in a random way.
  - The output of the teacher network is centered with a mean computed over the batch.
  - Stop-gradient (sg) operator on the teacher to propagate gradients only through the student.
  - The teacher parameters are updated with an exponential moving average (ema) of the student parameters.

Self-Supervision Code



Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Very simple but works amazingly well!

What Makes it Tick?

- The network—student or teacher—is trained to output the same feature representation given different distorted views of the same image.
- As a result, it learns invariant features.

26 authors!

Oquab et al., ArXiv'23

In Short

Texture is a key property of objects that is

- Non local
- Non trivial to measure
- Subject to deformations

➡Hard to characterize formally but deep nets can do a good job it.

➡This helps explain the unreasonable effectiveness of deep nets for segmentation.