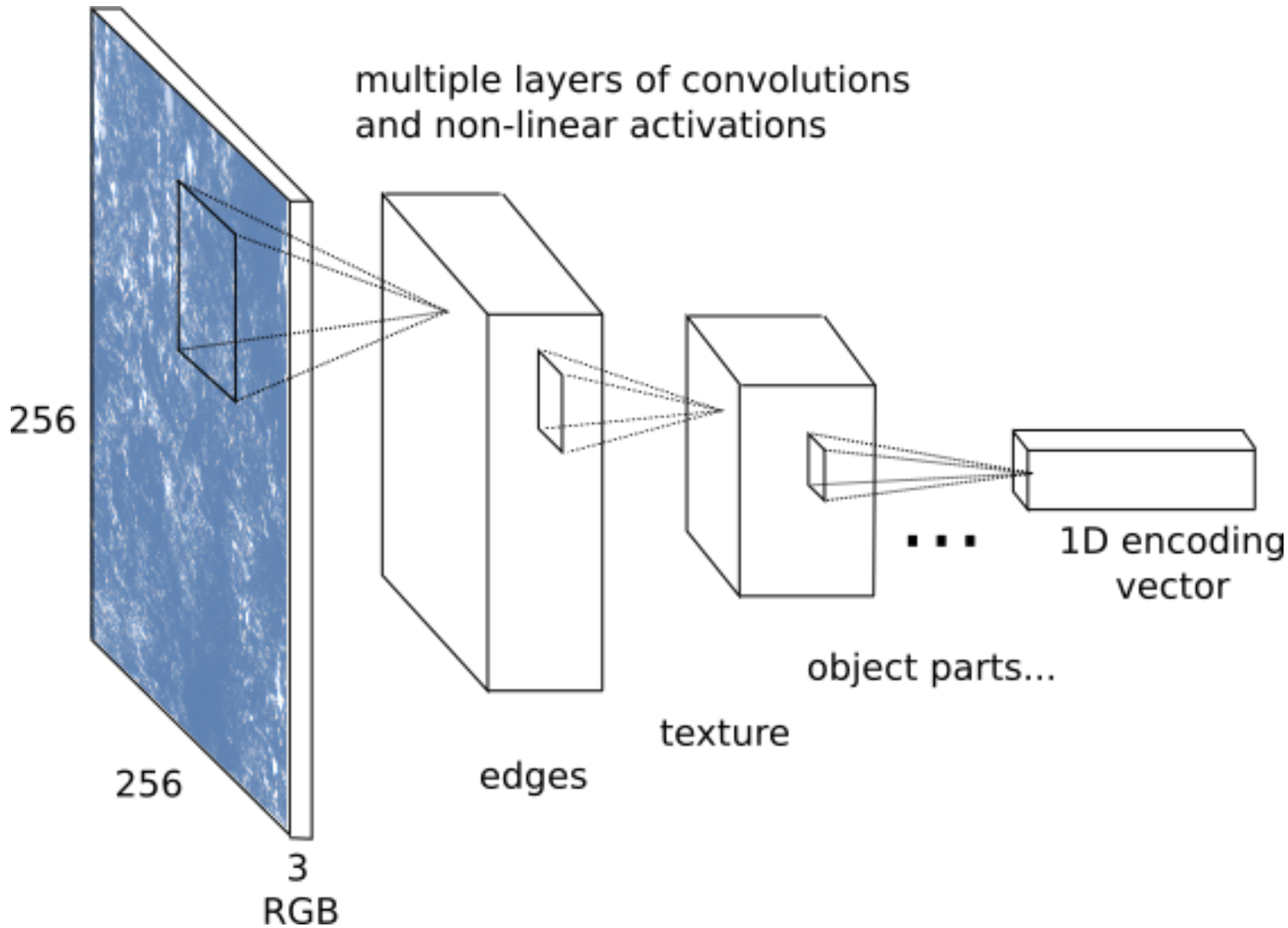
# Convolutional networks

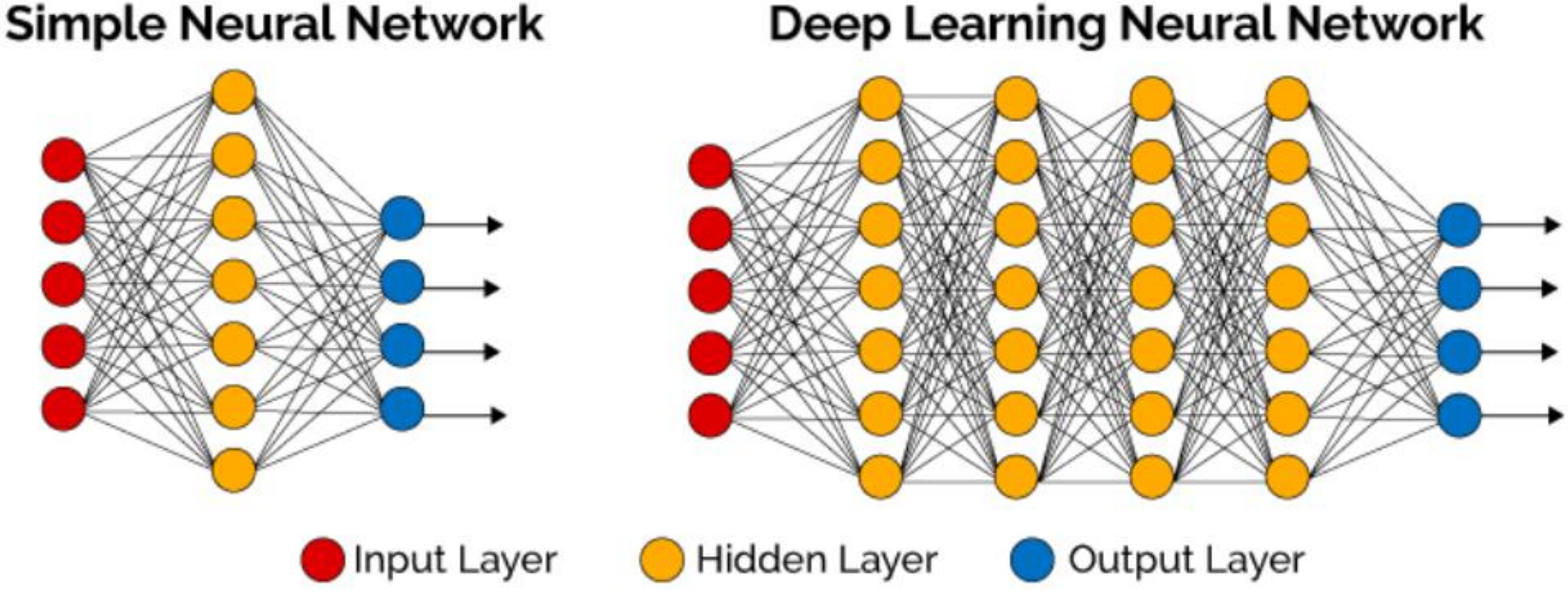



- How do I produce a neural network that behaves like this?

# What is deep learning?

# What is deep learning?

Uses **layers** of “neurons”, output from each layer connects to the next

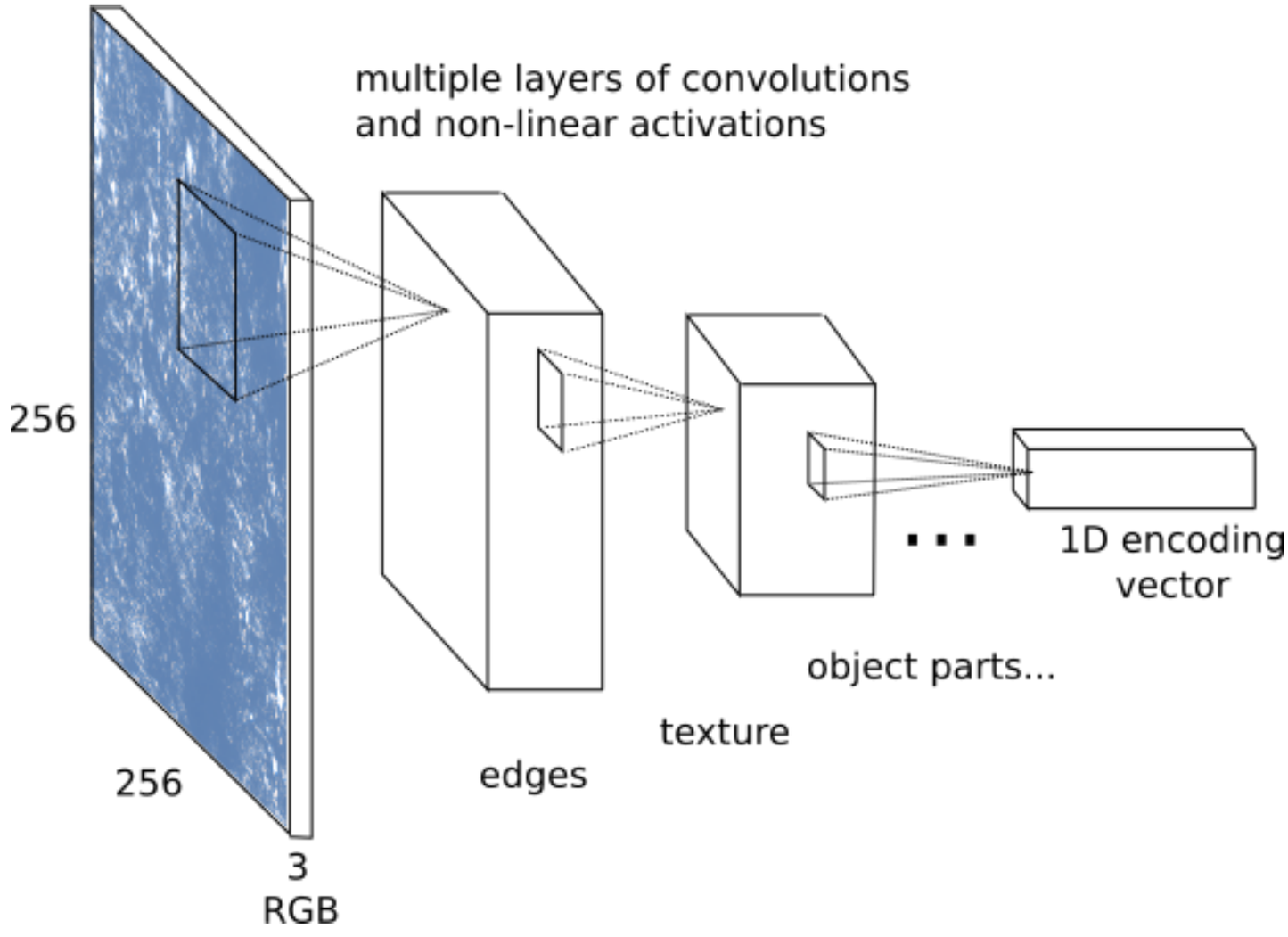


# What is deep learning?

- Many ways layers can be connected (*fully-connected, max-pooling, convolutions…*), which forms the model **architecture**
  - Key point: certain architectures are now known to work for specific applications

- *Convolutional* network are very (*very)* good for working with images
  - What are they: Combine *layers* of *convolutions* which have a finite *stencil width*, i.e. span only a finite number of points

# Convolutional networks

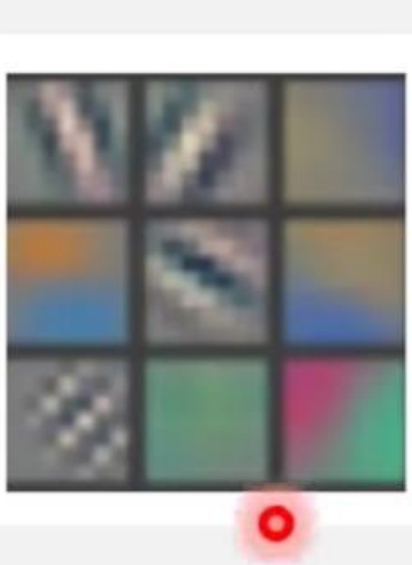


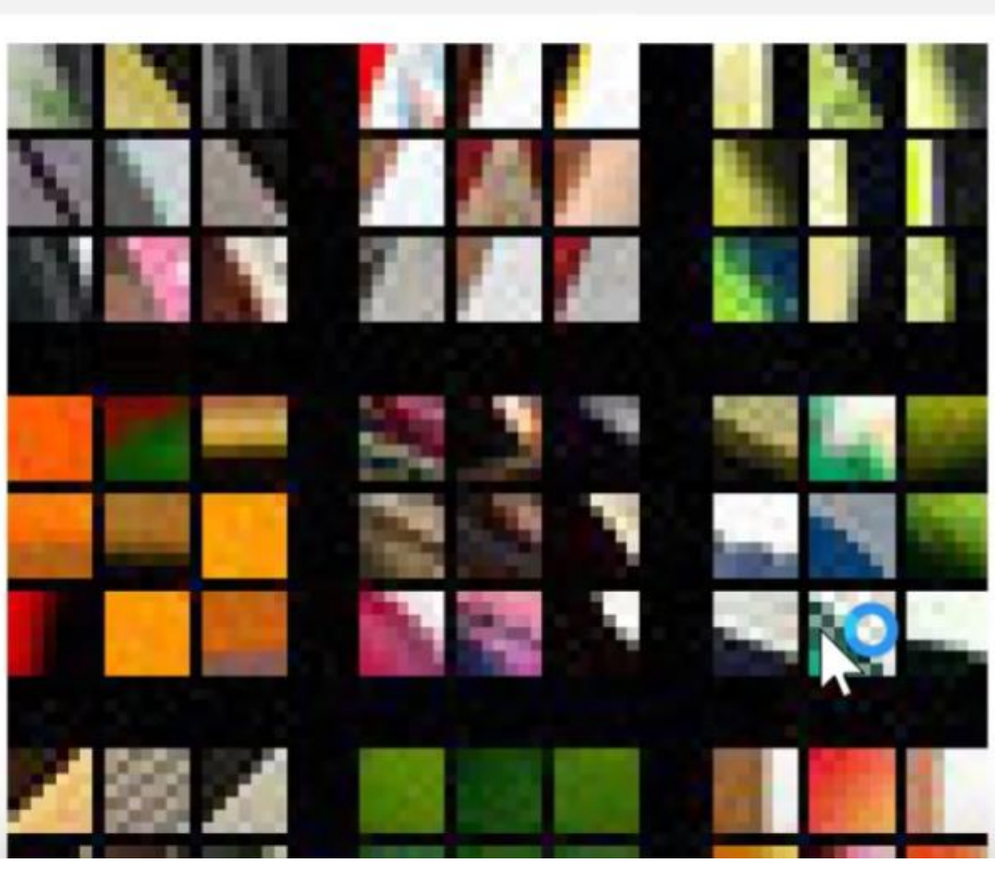

- Each layer contracts information from a finite part of image into a single value
  - These are composited over multiple layers to produce more complex features

# Visualizing and Understanding Convolutional Networks

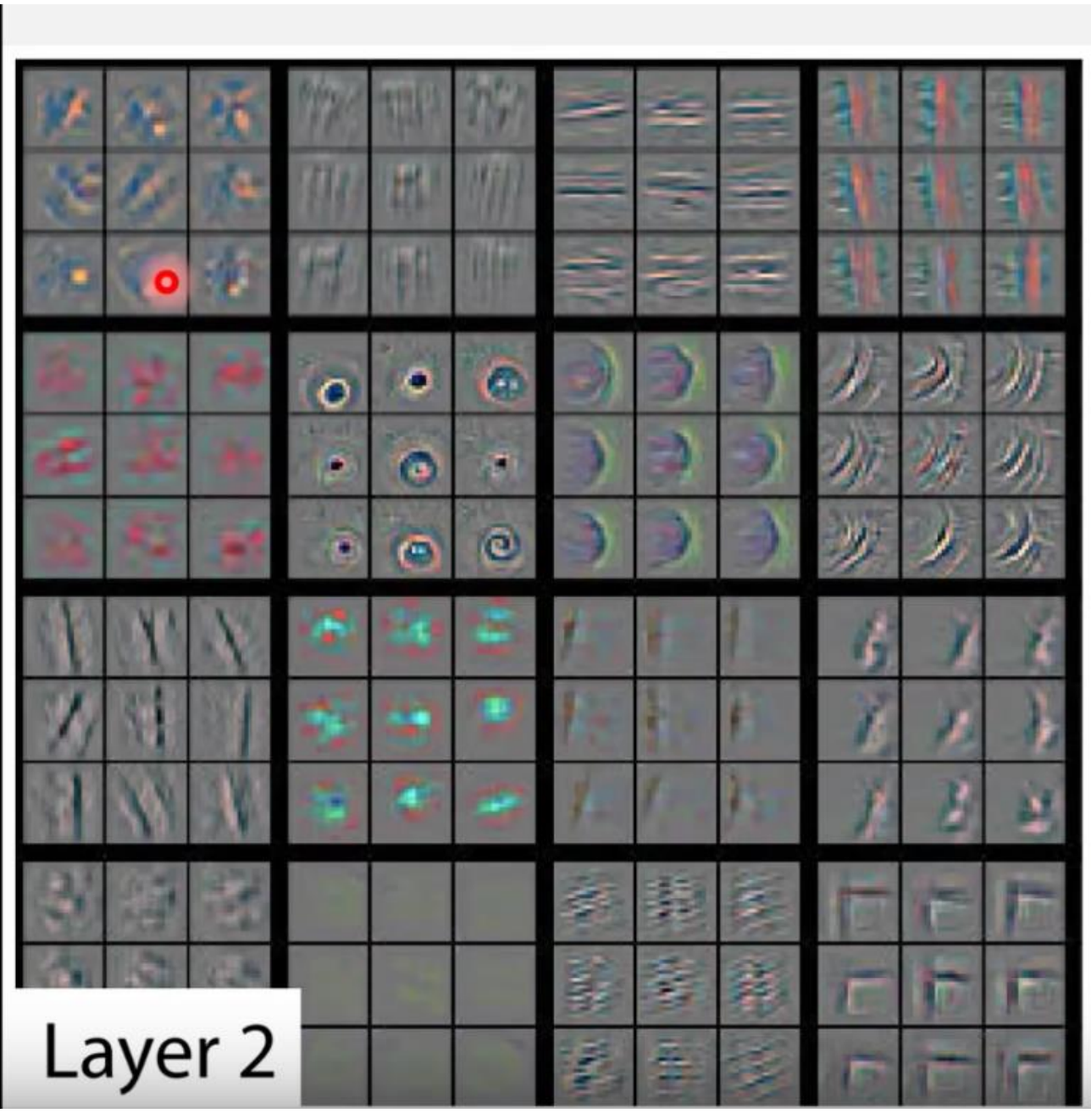
**Matthew D. Zeiler** ZEILER@CS.NYU.EDU
Dept. of Computer Science, Courant Institute, New York University

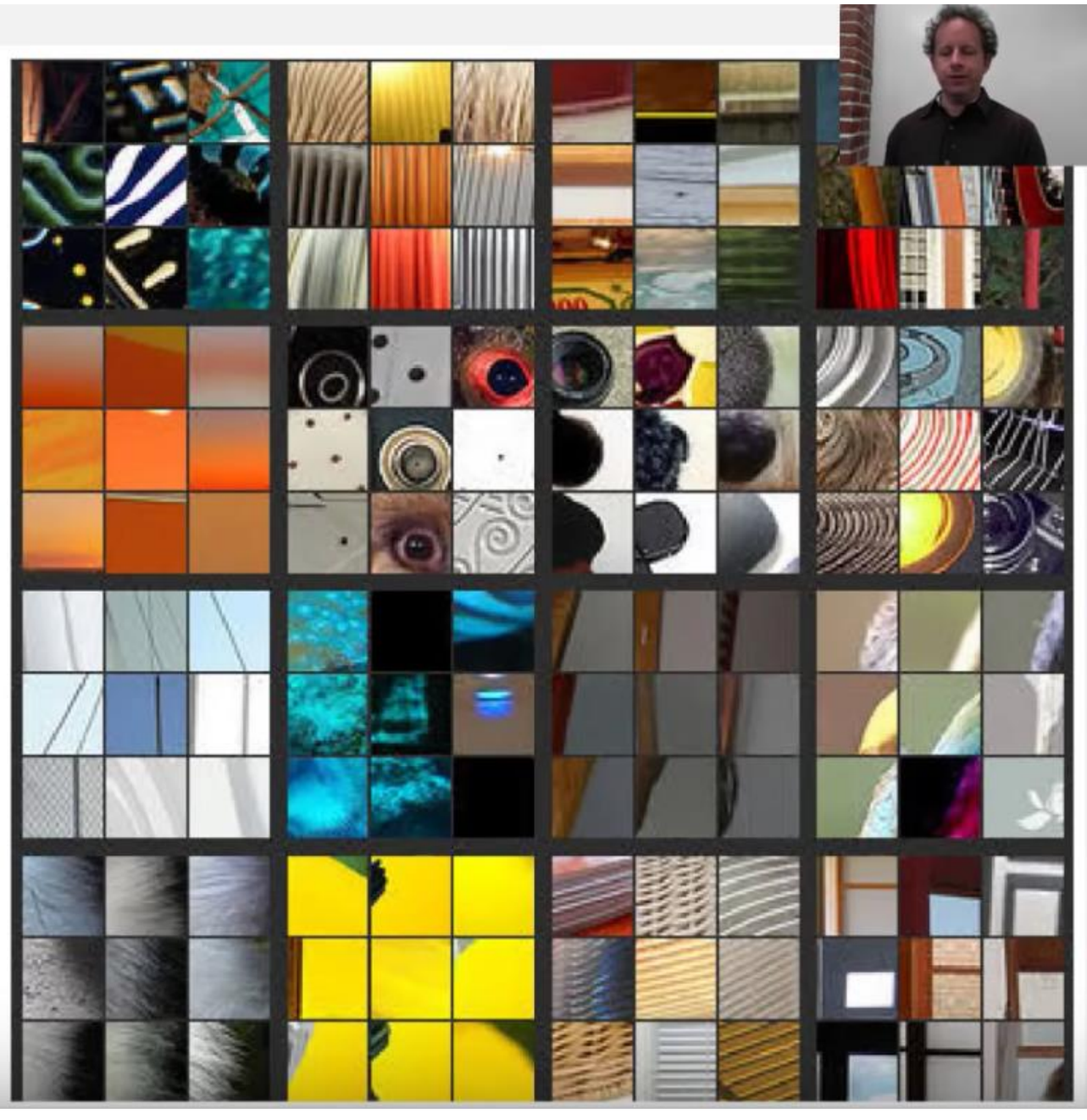
**Rob Fergus** FERGUS@CS.NYU.EDU
Dept. of Computer Science, Courant Institute, New York University



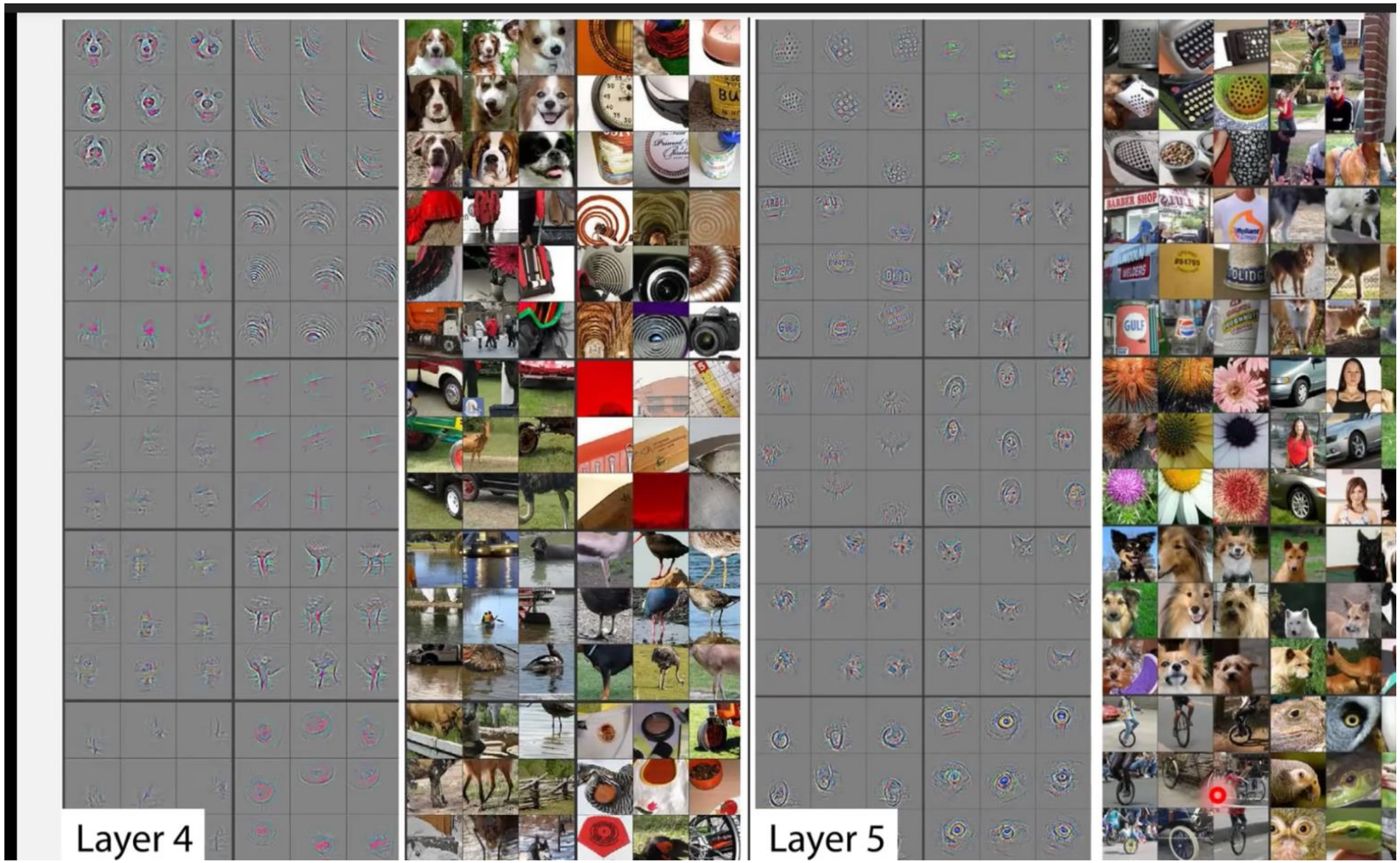



# Convolutional networks



# Convolutional networks



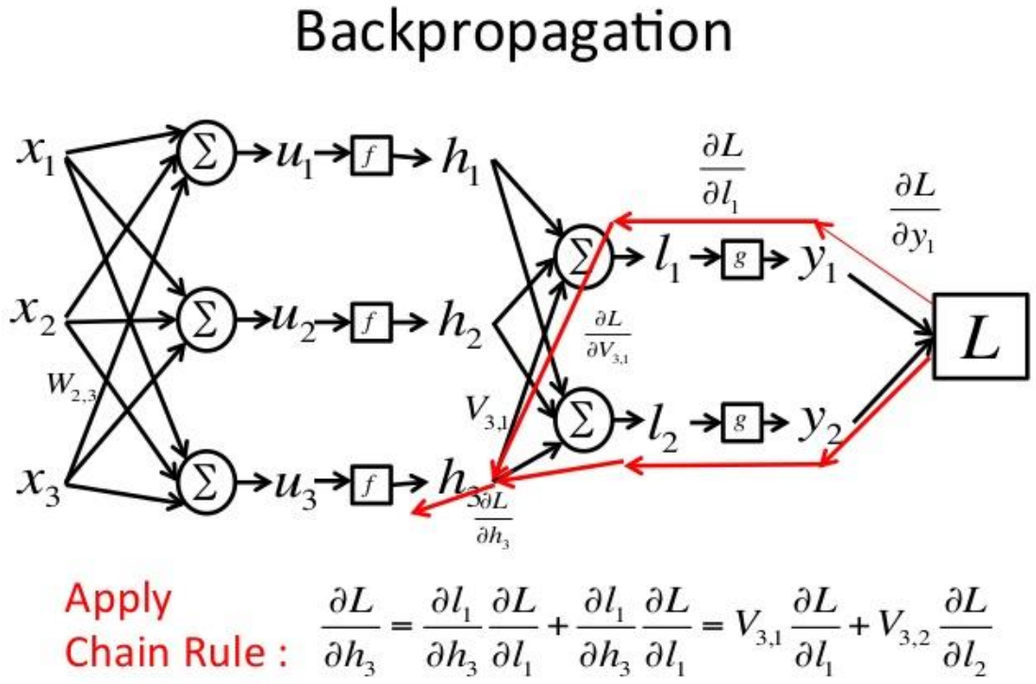
# How do you train a neural network?

- Model needs a *goal*, measure how close model is to goal with *loss function, L*

- Use *chain rule* to calculate what changes to *weights* in network *will reduce loss* for a given (set of) training example(s)

# How do you train a neural network?

- Model needs a *goal*, measure how close model is to goal with *loss function, L*
- Use *chain rule* to calculate what changes to *weights* in network *will reduce loss* for a given (set of) training example(s)




Learning Deep Learning (Sonse Shimaoka)

# What can neural networks predict?

Anything you want!

- As long as you can formulate a loss function for your problem

Examples

- Predicting location of something:
  - Need bounding box, four scalars y=[x0, y0, x1, y1]
- Predicting class between a predefined set (e.g. is it a cat, dog or fish)
  - Use “one-hot-encoding”, probability for each class is element of a vector, e.g. 3 scalars, y=[$p_{cat}$, $p_{dog}$, $p_{fish}$]
- Predicting temperature, concentration change:
  - Output is vector of increments, should probably normalize to ensure conservation, e.g. y=[$dT_0$, $dT_1$, $dT_2$, …]