Computational Media Research

CMPM 202, W2019

Prof. Angus Forbes (instructor)
https://creativecoding.soe.ucsc.edu
angus@ucsc.edu

Reading Responses

a) Break into small groups, take a few minutes to read each others responses.

b) Come up with one question (or a set of closely related questions), and lead short discussion with your group around that question.

- What are the goals of humanities inquiry? Do they fundamentally different from research activities in science and engineering?
- How can research in CM bridge disciplinary approaches and utilize humanities methodologies?
- Will ML someday shape human culture? Does it already? How?
- “Surface” data or “deep” data or “deep surface” data? Which do you use in your own research or media creations?

Deep Learning has lead to SOTA for many research areas:

- Realtime text/speech translation
- Identifying/Segmenting objects in photos
- Self-driving cars and drones
- Predictive keyboards
- Gesture recognition
- Lip reading
- Product recommendation
- Tumor detection
- Speech synthesis
- Artistic style transfer
- Image processing
- Chess, Go, Starcraft, Montezuma’s Revenge

Machine learning

Overarching idea = learn from data

- Complex systems have too many rules, or rules that are difficult to quantify.

- Rather than try to come up with all of the rules, create a system that can learn the meaningful rules automatically, through examining lots of data where examples of the rules are expressed.

- Train your system on examples where you know the right answer, and then test to see if it works on new examples.

Machine learning

Discriminative: Detecting events, Finding patterns, Classifying objects, Recognizing elements

Generative: Synthesizing data, Inferring examples, Interacting with models

Machine learning

Classification and Generation are inverses of each other:

If I have the knowledge of what features determine whether or not a specific sample belongs or doesn't belong to a particular category or class,

Then I can also use those features to create new samples that are examples of a particular category or class

Machine learning

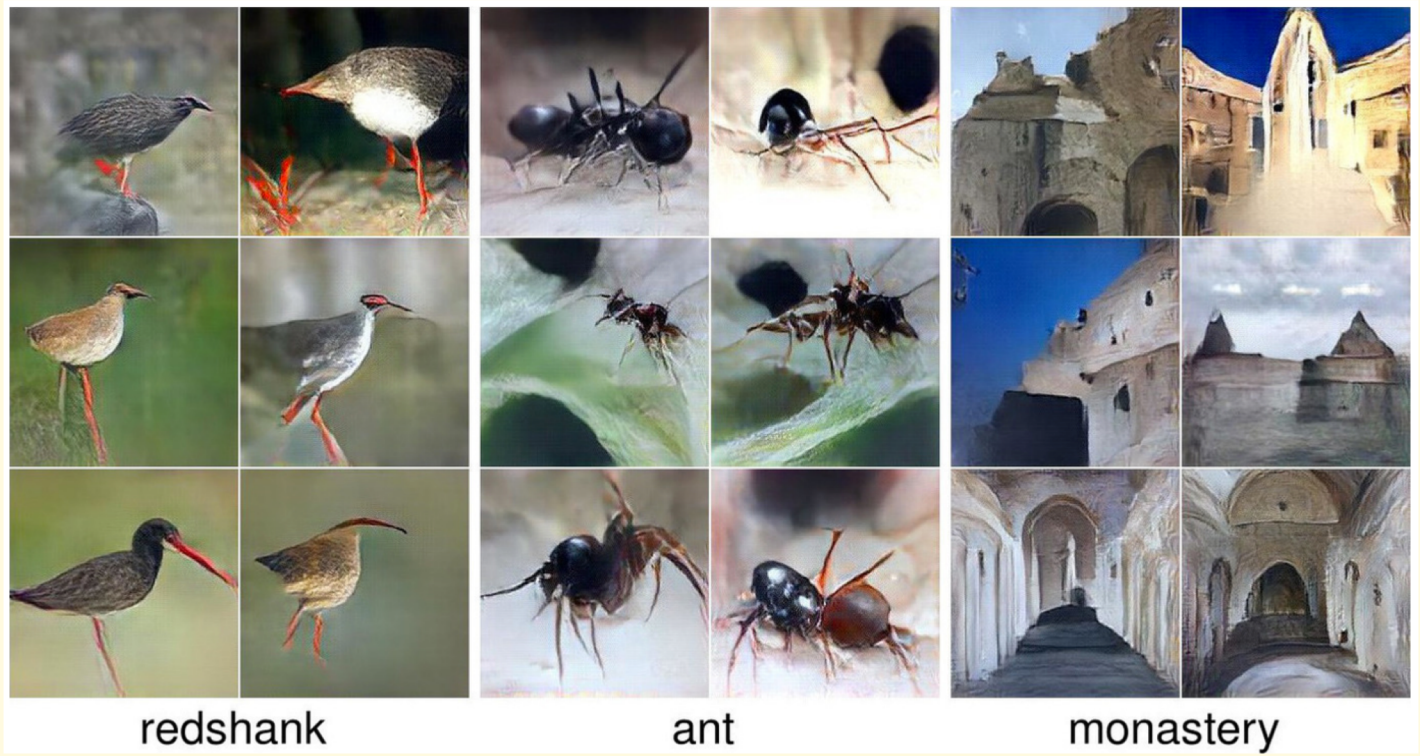
Online classifier for handdrawn objects (Google's A.I. Experiments) –

https://aiexperiments.withgoogle.com/quick-draw

GAN Paint application –

https://gandissect.csail.mit.edu/

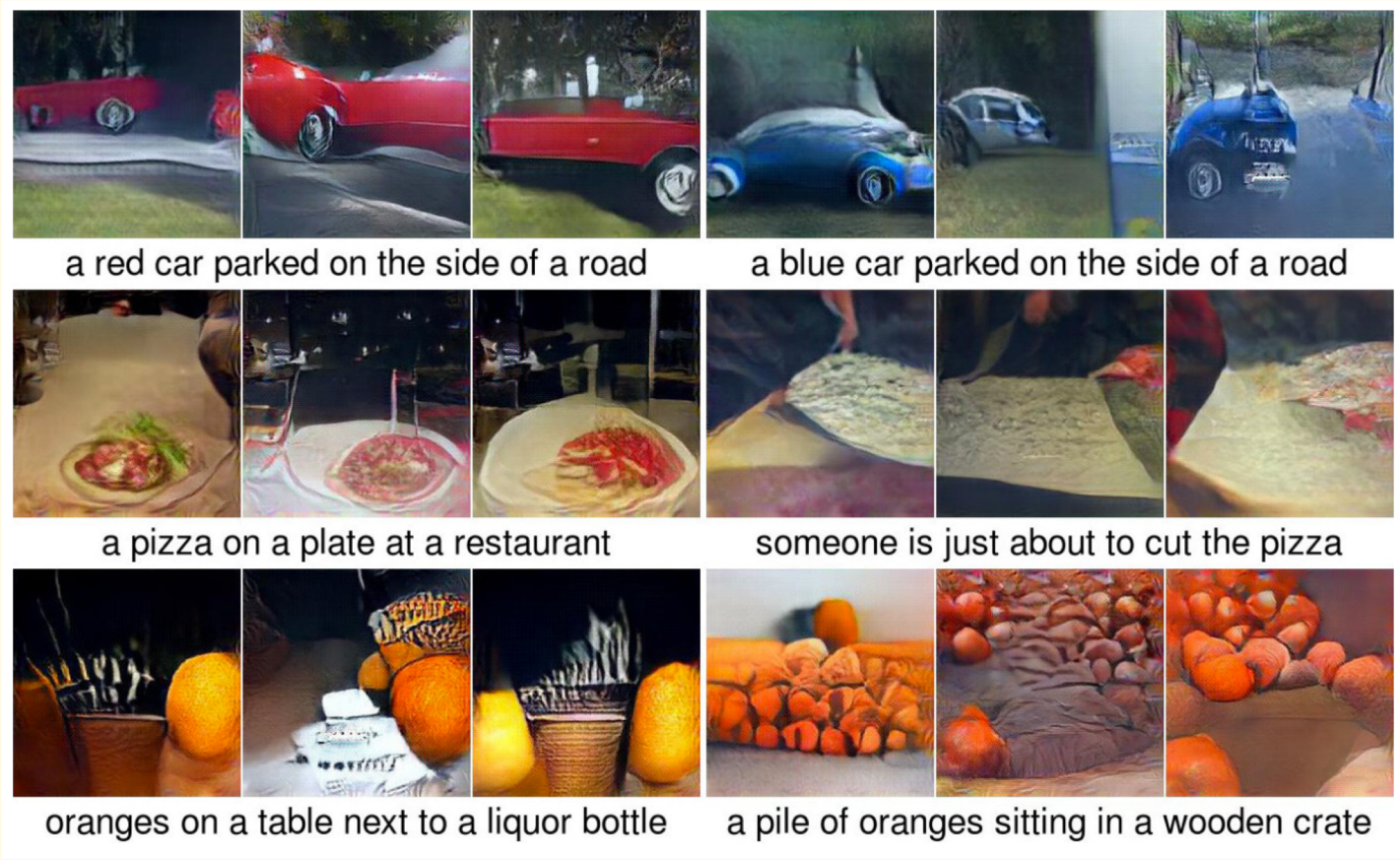
PPGAN (Plug and Play Generative Networks)



http://www.evolvingai.org/ppgn

PPGAN (Plug and Play Generative Networks)

PPGAN (Plug and Play Generative Networks)



StackGAN: Image Synthesis From Text

https://www.youtube.com/watch?v=rAbhypxs1qQ

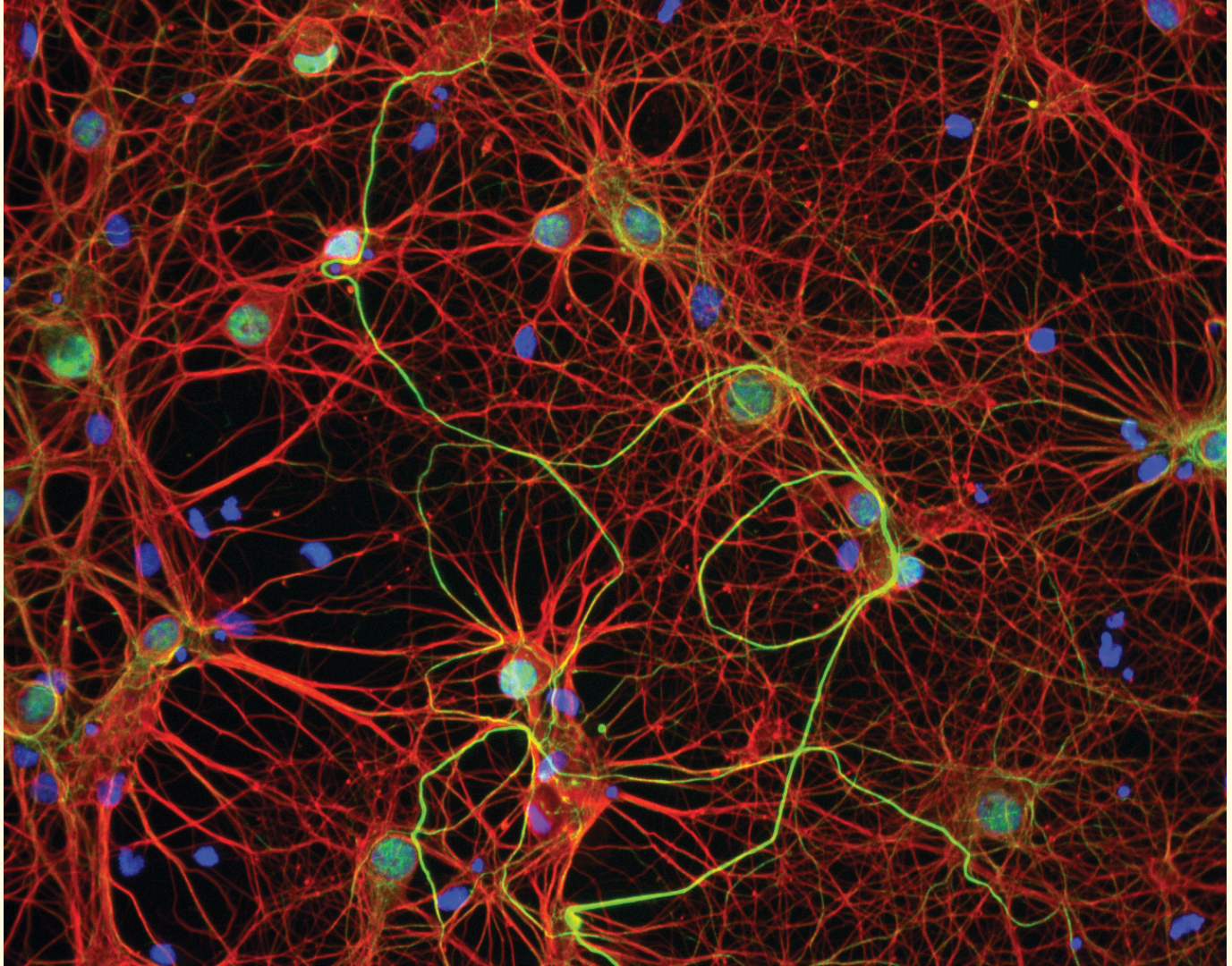
Overview of Neural Networks

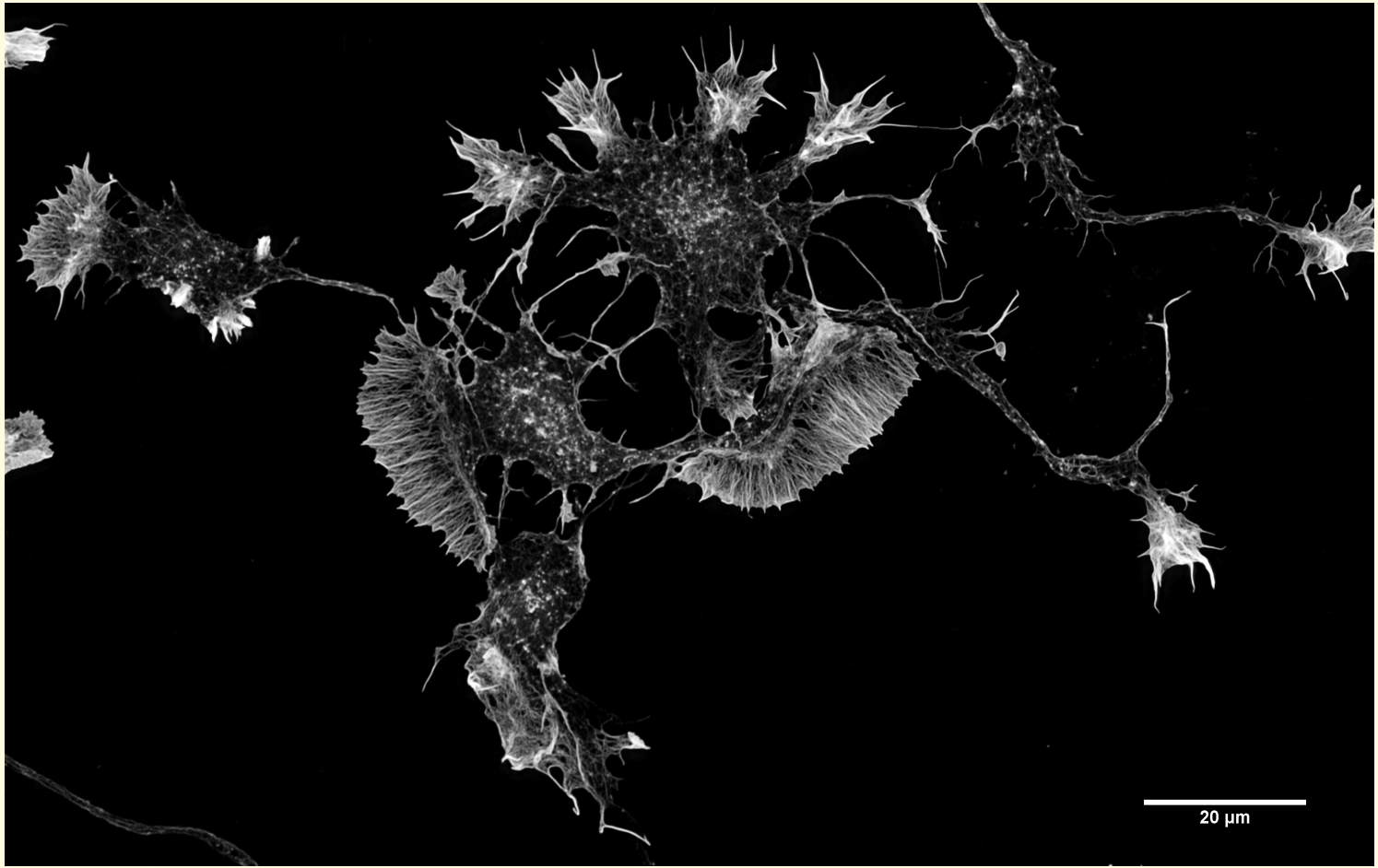
- Artificial Neural Networks (ANNs, or just NNs), and their many flavors, are all inspired from our understanding of how the brain works.

- Although biological networks are extremely complex, even the very oversimplified representations used in ANNs have shown to be very useful for interpreting (and synthesizing!) really world complex data.

- ANNs provide an approach for “learning” to approximate various types of functions: real-valued, discrete valued, and vector-valued.

Tom Mitchell, Machine Learning (1997)




20 µm

Overview of Neural Networks

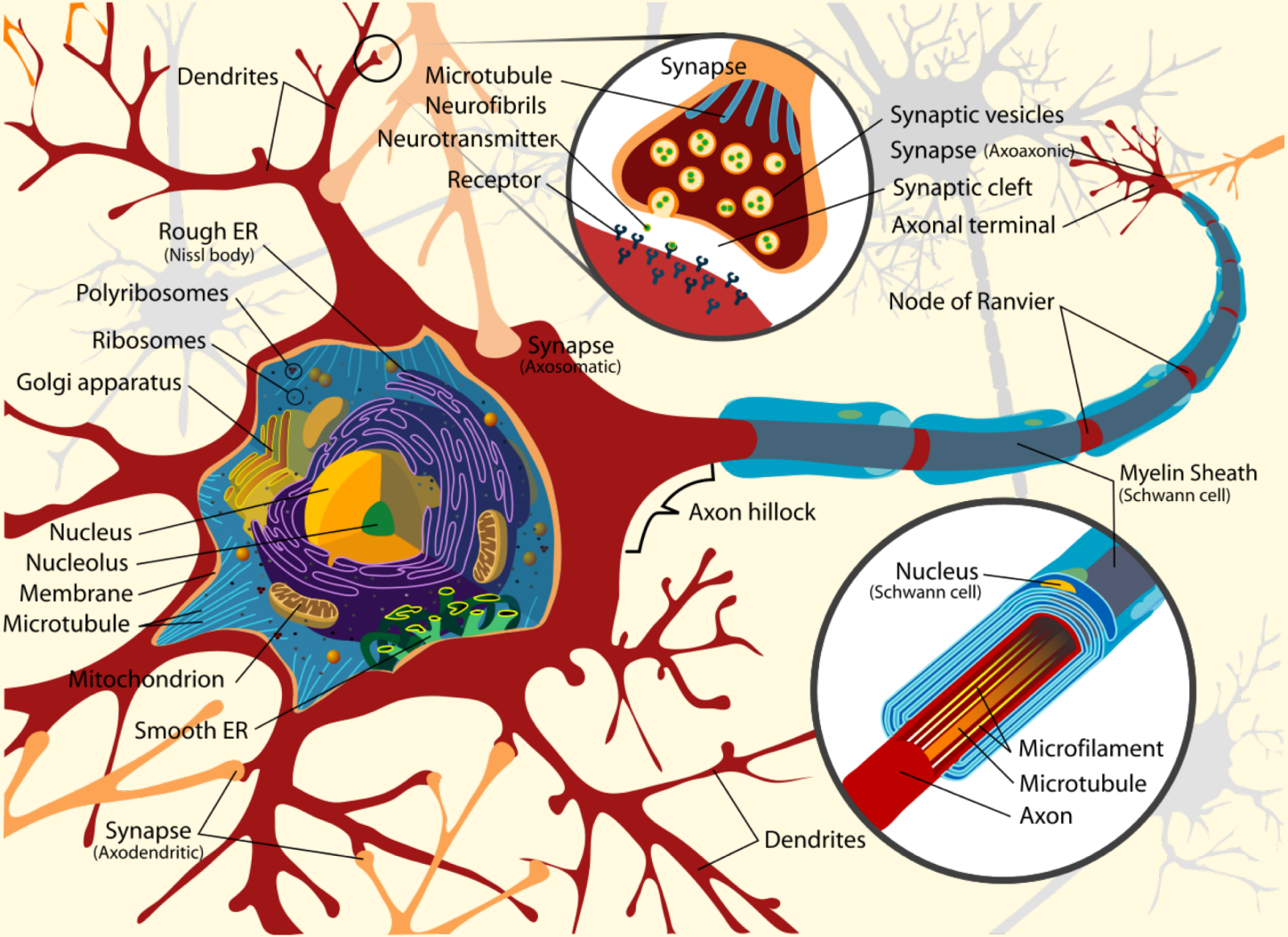
Human brain contains ~100,000,000,000 neurons, each of which is connected to between 10,000 to 100,000 other neurons. The process of responding to sensory input, managing our various bodily systems, controlling our motor functions, and thinking involves the constant firing of neurons.

Yet, these neurons fire relatively slowly - about 1/1000th a second— much, much slower that the speeds of your CPU. For example, it takes ~1/10th of a second to recognize somebody.

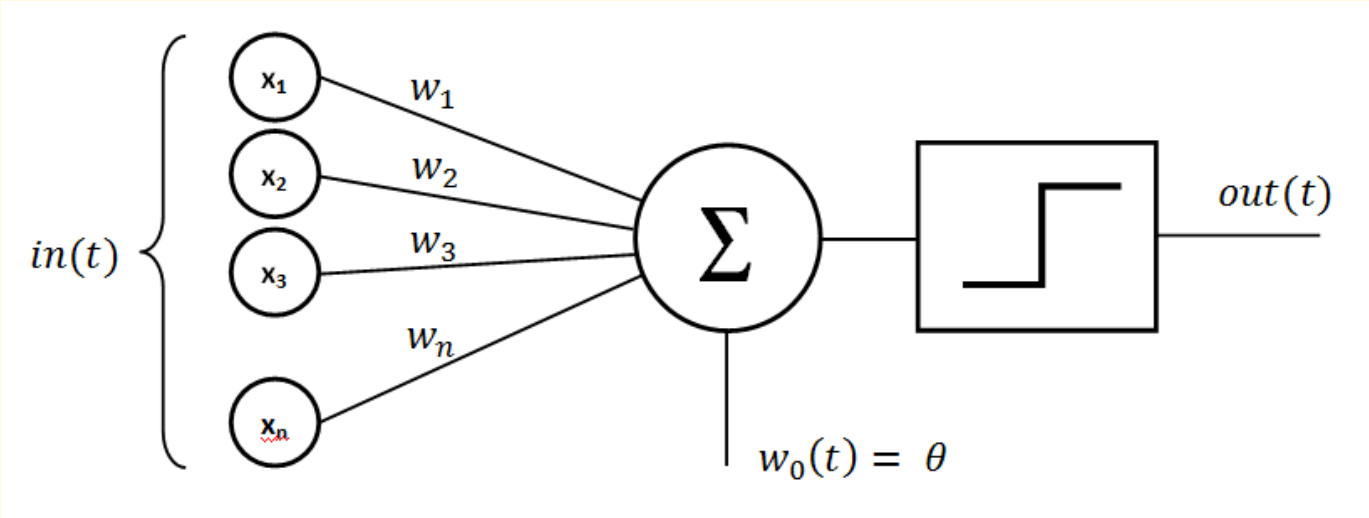
Overview of Neural Networks

Because the speeds are so slow, it seems clear that the information processing abilities of the brain is due to highly parallel processes operating on neurons distributed across the brain. Recent advances in brain imaging (MRI, Calcium Imaging, etc.) have illustrated this clearly.

The field of ANN started with some simple formulations to emulate the processing power of neurons. Described most simply, a neuron “fires” (outputs an electrical charge) once it has reached a particular threshold of input electrical activity from it’s neighbors.


Dendrites
Microtubule
Neurofibrils
Neurotransmitter
Receptor
Synapse
Synaptic vesicles
Synapse (Axoaxonic)
Synaptic cleft
Axonal terminal
Rough ER
(Nissl body)
Polyribosomes
Ribosomes
Golgi apparatus
Synapse
(Axosomatic)
Node of Ranvier
Myelin Sheath
(Schwann cell)
Axon hillock
Nucleus
Nucleolus
Membrane
Microtubule
Mitochondrion
Smooth ER
Nucleus
(Schwann cell)
Microfilament
Microtubule
Axon
Synapse
(Axodendritic)
Dendrites

Perceptron (a NN with a single node)



Overview of Neural Networks

Warren McCulloch (UIC) and Walter Pitts (UChicago) defined the first mathematical formulations of neural networks in 1943.

- “A Logical Calculus of Ideas Immanent in Nervous Activity,” McCulloch and Pitts, 1943

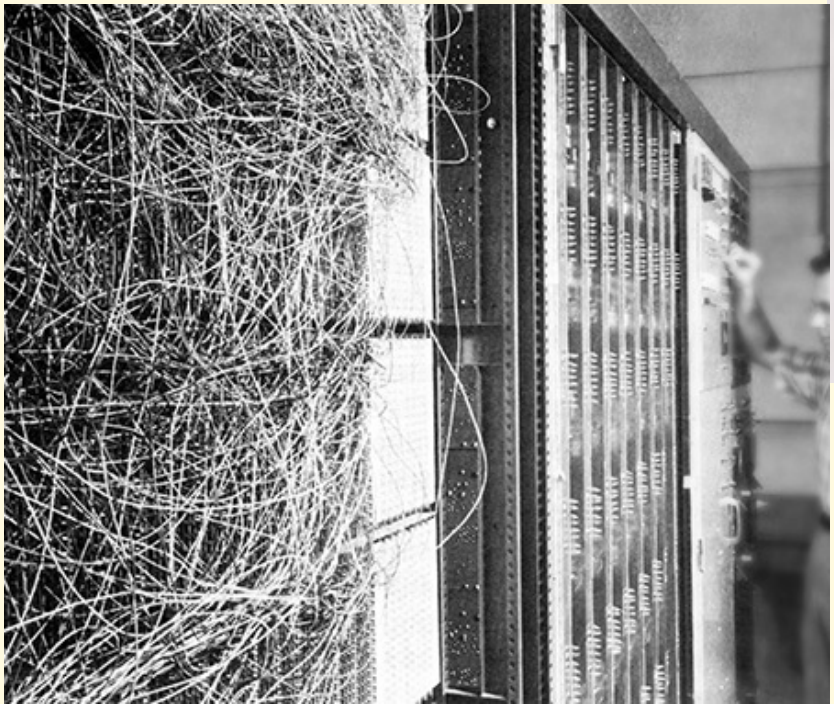
- http://nautil.us/issue/21/information/the-man-who-tried-to-redeem-the-world-with-logic

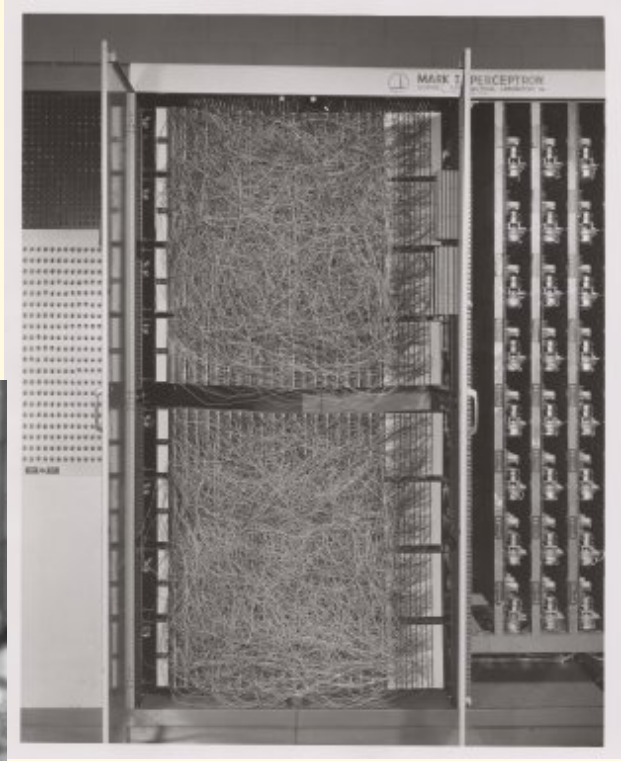
Overview of Neural Networks

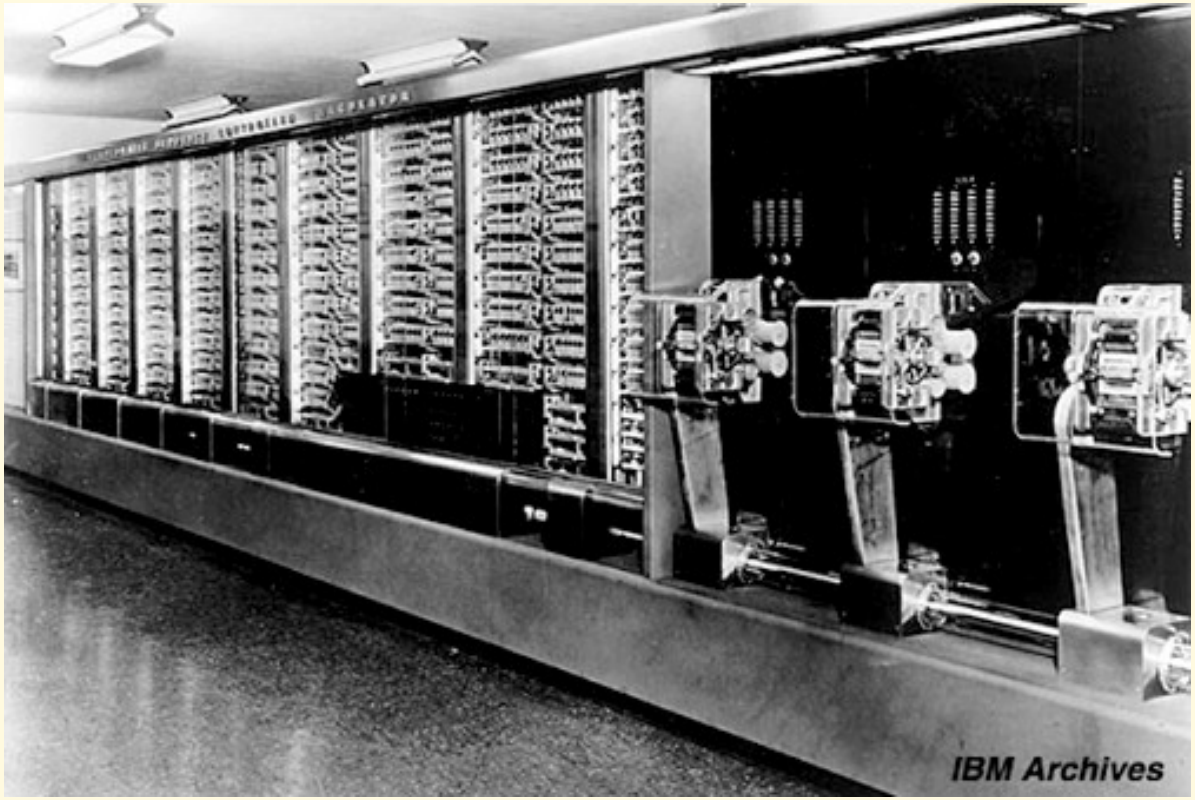
Based on McCulloch and Pitts work (and the work of many others), Frank Rosenblatt designed the first “perceptron” in the late 1950s, and then built it with analog circuits.

The perception is capable of taking multiple inputs and classifying them into two different classes. If the data is linearly separable, then it can always learn to converge on an accurate solution (although not always an optimal solution).

Mark 1 Perceptron

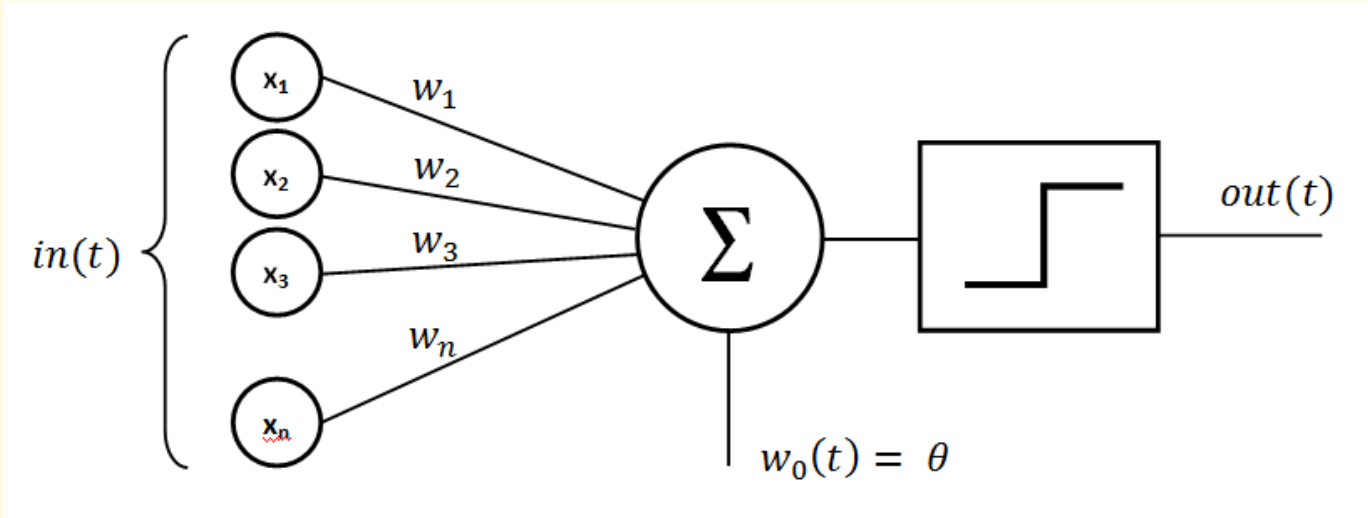





IBM Archives

Overview of Neural Networks

The perceptron is a simple structure that takes in an “input vector”, which consists of some number of elements (e.g., for image recognition, an array of pixels in an image). A weight are attached to the link between each input element and the “perceptron” itself. The perceptron “learns” these weights during the training session.


$in(t)$
x_1
x_2
x_3
x_n
w_1
w_2
w_3
w_n
Σ
$w_0(t) = \theta$
$out(t)$

Overview of Neural Networks

In the training session, you start with labeled data — data for which you already know the correct answer.

The perceptron encodes a simple step function that outputs either a 0 or a 1, based on a simple summation: output = for (1 to n), o += input(k) * weight(k). if the o value > 0, then output (“fire”) a “1". Otherwise do not.

Overview of Neural Networks

The weights originally will be set at random, and so you might as well flip a coin. But during training, the perception changes the weights based on how far off the answer was. The weights are updated so that they more closely approximate the correct answer.

```
newWeight = prevWeight +
                ((desiredOutput - prevOutput) * prevInput)
```

Overview of Neural Networks

If this is done enough times, and the data is linearly separable, then the neural network will converge so that the perceptrons output always matches the desired output for data from the training set.

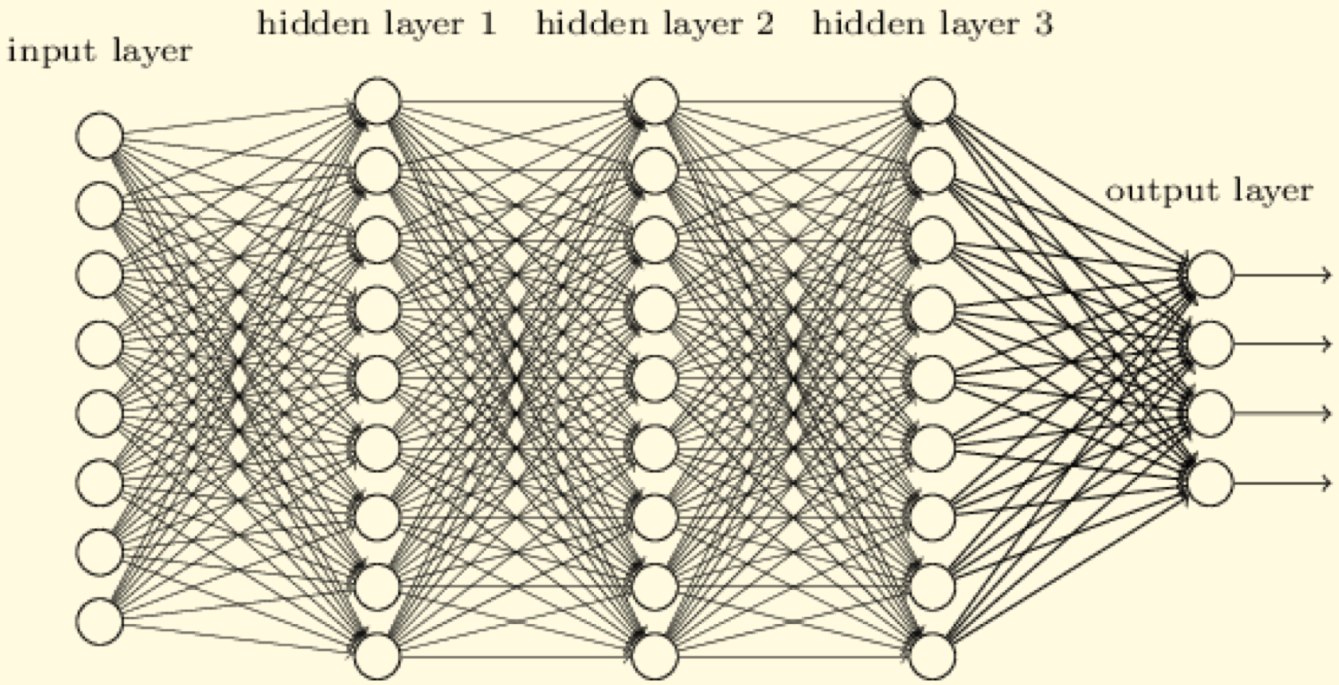
And, assuming that your training set is representative of real-world conditions, inputting any new data will also result in the correct answer.

Overview of Neural Networks

The limitations of the original perceptron NN is that it can only learn linear functions, and that it can only distinguish between two categories.

To enhance the power of NNs, these artificial preceptors can be chained together to create “multilayer” networks.

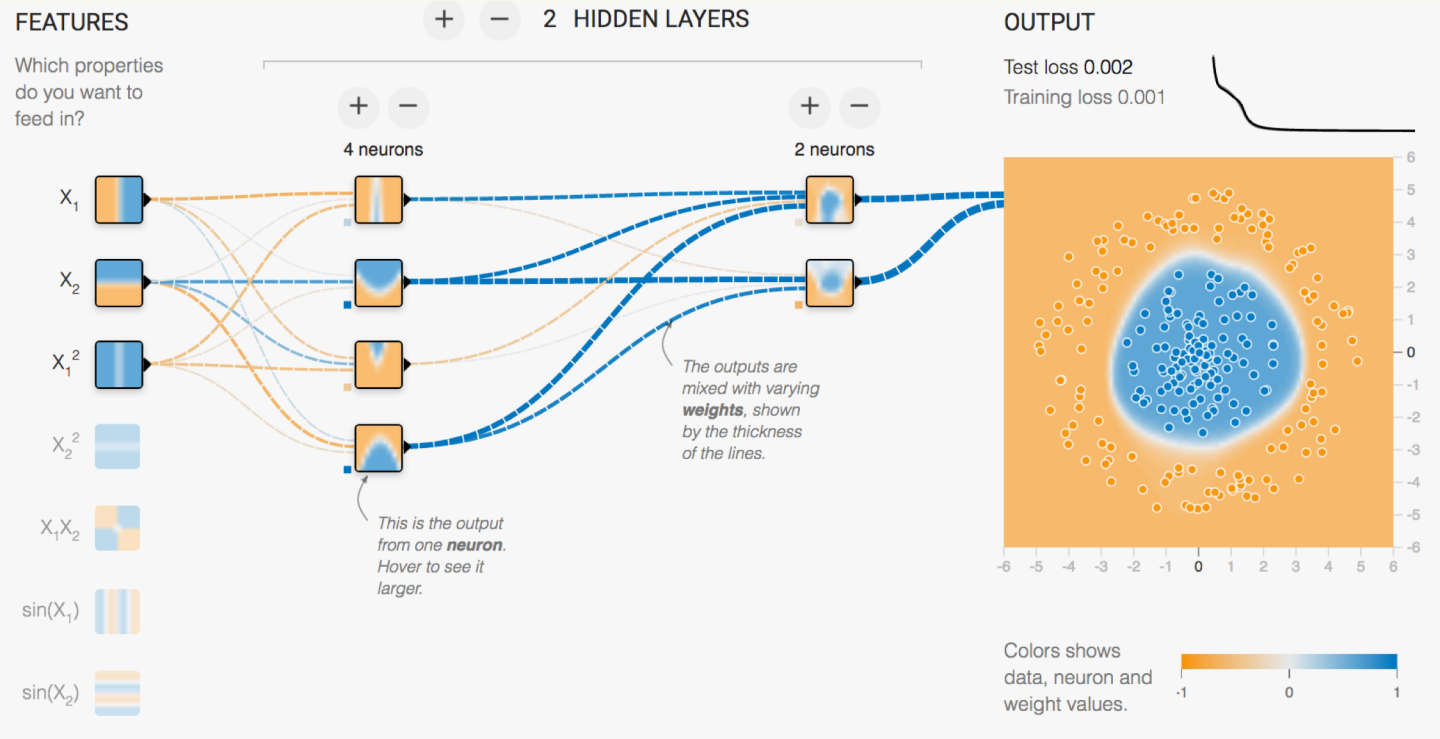
The “traditional” NN that is the basis of all of the NNs we’ll look at is a multilayer network that uses backpropogation to update weights attached to the links between nodes.


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

Overview of Neural Networks

- uses a different “perceptron” - a “threshold unit” that is a sigmoid and tanh function instead of a step function, which makes it easier to encode non-linear functions.
- uses multiple “hidden” layers.
- uses gradient descent to minimize error, updating weights incrementally.
- applies different strategies to attempt to not fall into local minima (i.e. to not converge on functions that aren’t appropriately expressive of the range of training data).
- each node in the multilayer NN, (sometimes called a BPNN, or more recently a Deep Learning NN) is trained to detect particular features.

http://playground.tensorflow.org/



Overview of Neural Networks

- hidden layers:

can learn to invent new features not explicit to the data. One common example sometime used in intro NN classes shows an 8 x 3 x 8 network for mapping a number to itself. The hidden layer learns to encode binary numbers (3 bits are required to encode 8 binary numbers)

- overfitting:

too much training / too many neurons / layers - can have an adverse effect, causing the NN to be highly sensitive to the original training data, and to fail to be generalizable to other, similar data.

Overview of Neural Networks

- implicit biases in training data:

there are *lots* of examples where the mapping of training data to the real world is incomplete or fails– it can be tricky to find a dataset that has wide enough coverage to generalize to real-world situations.

- Camouflaged tanks in trees (https://www.gwern.net/Tanks)

- http://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist (@TayTweets)

- http://www.usatoday.com/story/tech/2015/07/01/google-apologizes-after-photos-identify-black-people-as-gorillas/29567465/

- https://www.nytimes.com/2016/07/01/business/self-driving-tesla-fatal-crash-investigation.html

Overview of Neural Networks

Convolutional NNs (CNNs)

- Attempt to model visual processing, which has been suggested to be hierarchical or “pyramidal” in nature
- Detects high-level features first, then more fine grained features
- Uses convolution to create “feature maps”
- Math is similar to commonly used mathematics in image processing kernels

Convolutional Neural Networks

- Have had enormous success at image classification tasks

- Widely used by Google, Facebook, Instagram for identifying both general categories and individual elements in photos and videos; core of self-driving car algorithms, etc.

- Have been adapted to wide range of other types of problems, even those that don't involve images, such as search tasks and recommendation systems

Vision neurons

- In 1962, Hubel & Wiesel inserted microscopic electrodes into the visual cortex of experimental animals to read the activity of single cells in the visual cortex while presenting various stimuli to the animal's eyes.

- Nearby cells in the cortex represented nearby regions in the visual field

- The visual cortex represents a spatial map of the visual field.

Vision neurons

- Individual cells in the cortex respond to the presence of edges in their region of the visual field.

- Some of these cells fire only in the presence of a vertical edge at a particular location in the visual field, while other nearby cells responded to edges of other orientations in that same region of the visual field.

- Orientation-sensitive cells, called "simple cells", are found all over the visual cortex.

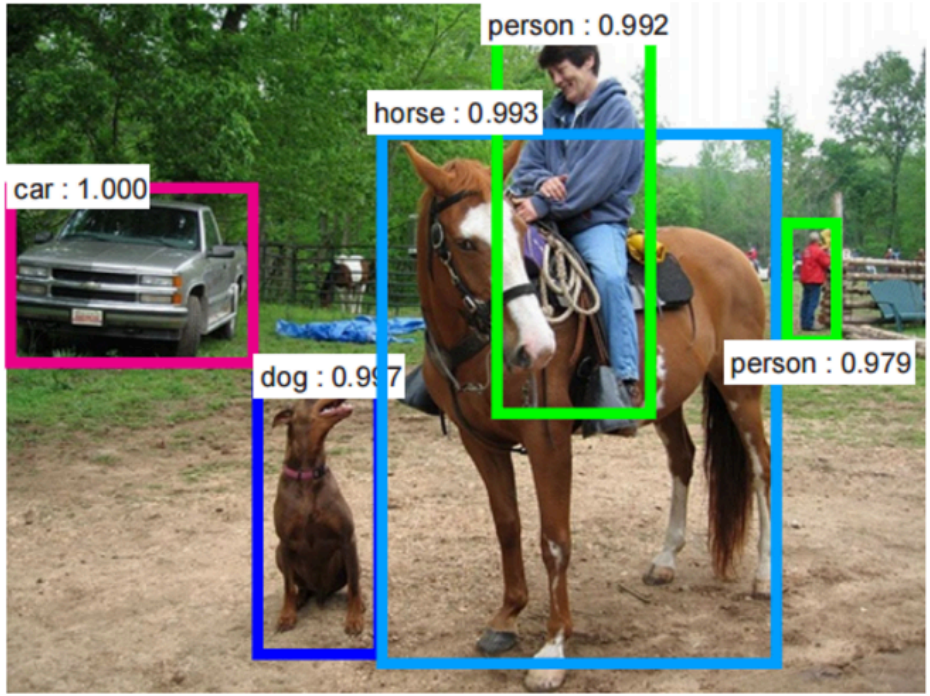
Vision neurons

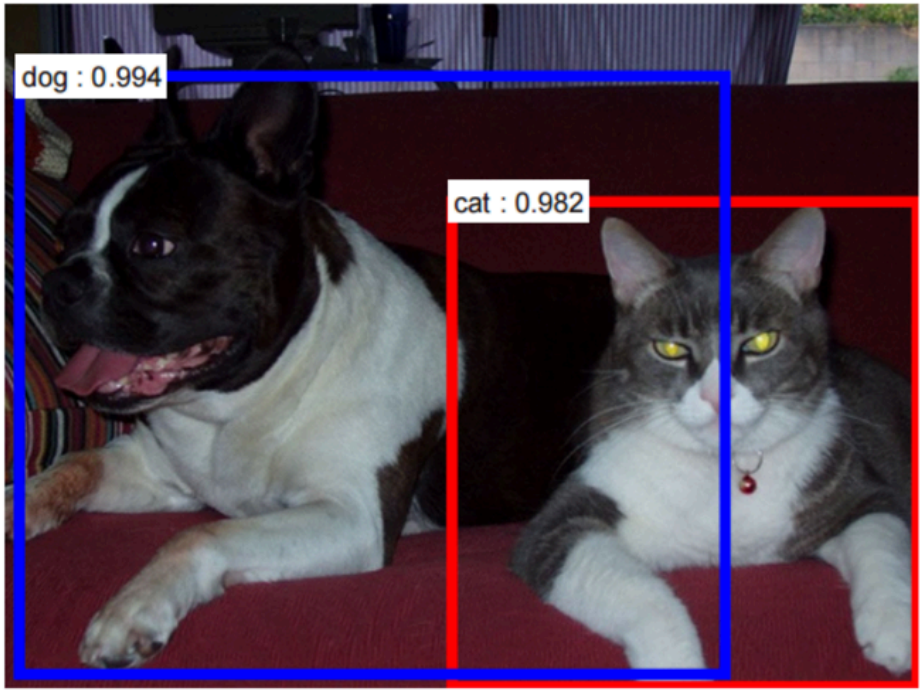
- “Complex cells” also respond to edges of a specific orientation at a specific location in the visual field, in a contrast-insensitive manner.

- Complex cells respond to their edges in a larger region of the visual field. This suggested that complex cells received input from lower level simple cells by way of a receptive field.

- Higher level "hypercomplex cells” respond to more complex combinations of the simple features, for example to two edges at right angles to each other in a yet larger region of the visual field.

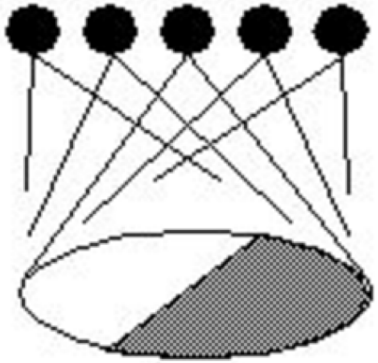
- See “Hubel & Wiesel” by Steven Lehar: http://cns-alumni.bu.edu/~slehar/webstuff/pcave/hubel.html


person : 0.992
horse : 0.993
car : 1.000
dog : 0.9
person : 0.979

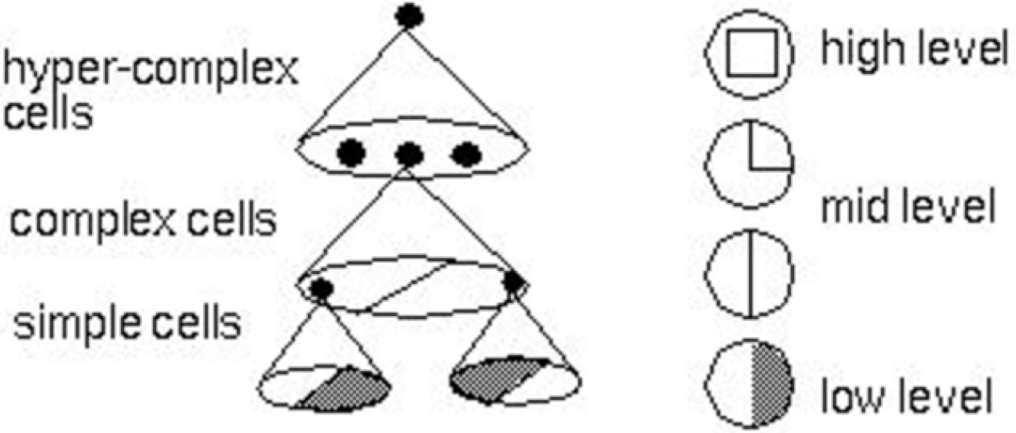

dog : 0.994
cat : 0.982

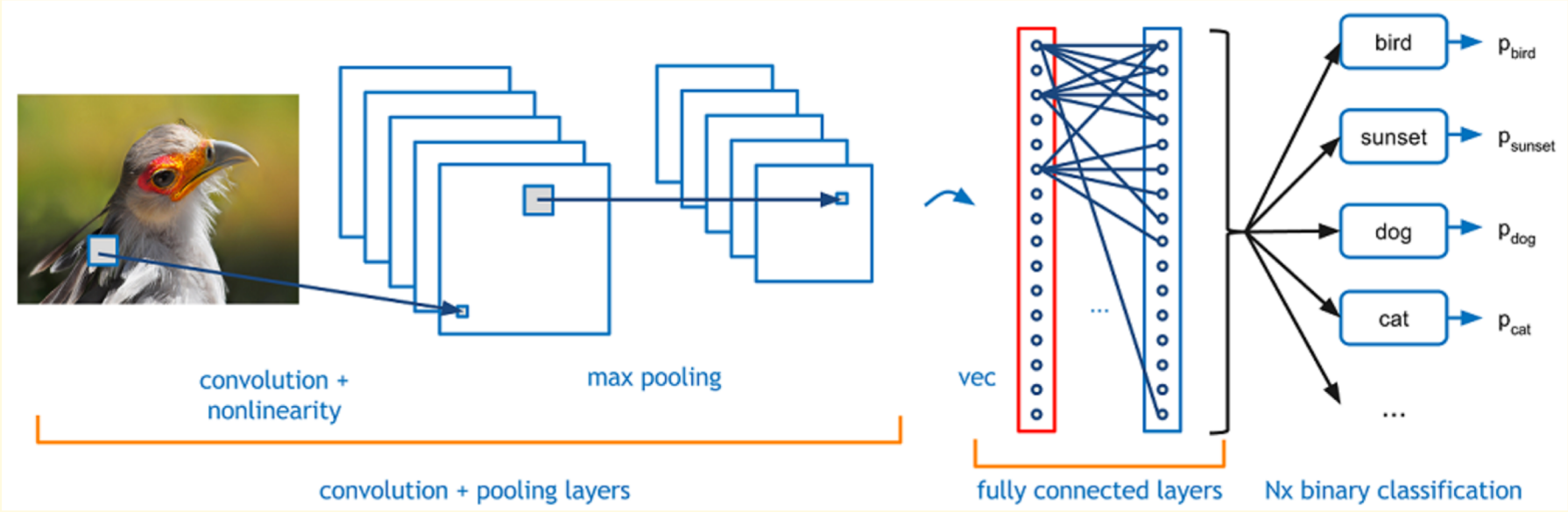
Hubel & Weisel

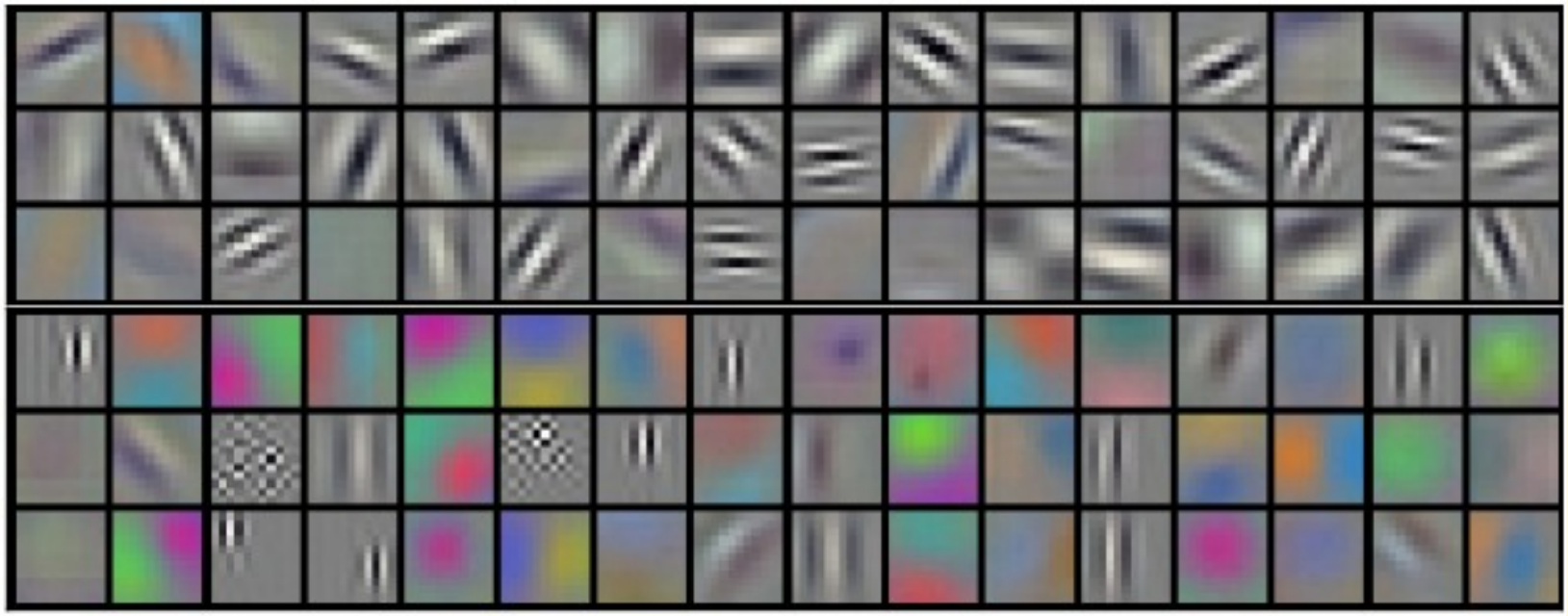
topographical mapping



featural hierarchy




convolution +
nonlinearity
max pooling
vec
bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + pooling layers
fully connected layers
Nx binary classification



- Krizhevsky, et al., “ImageNet Classification with Deep Convolutional Neural Networks”

CNNs

Neurons within the neural network “fire” (that is, contributes to the inputs of neurons in the next layer) if its value was greater than a threshold value.

CNNs additionally include an initial type of layer, called a “convolution layer,” in which a convolution operation is applied (often successively) to the input data (most often pixel data).

CNNs

A convolution filter in image processing transforms each of the input pixels into a new "convolved" output pixel:

for each pixel:

- place the convolution filter so that it is centered on the pixel
- where the filter overlaps a pixel overlaps, multiply the the element in the filter by that pixel, and divide it by the number of overlaps
- add them all together
- that's the convolved value of your output pixel

The CNN *learns* these filters – that is, it learns the most effective setting for the values on each of the cells of the convolutional kernel. Only values that end up being successful in distinguishing features that help to classify the training data correctly will survive the training session.

http://cs231n.stanford.edu/

Homework

- Reading tutorials on Convolutional Neural Networks

Next class:

- Reading Responses + Homework for week 4 will be assigned on 1/14
- Image classification with TensorFlow
- Training a CNN to classify custom categories