CS 523: Multimedia Systems

Angus Forbes

creativecoding.evl.uic.edu/courses/cs523

Today

- Presentations
- Overview of Neural Networks
- Project 1
- Lab

Creative AI reports

What is creative, original, exciting about the project?

What do you like about it? What drew you to it?

What other creative works does it remind you of?

Describe in detail how what it does and how it works.

What generative, AI, or machine learning techniques does it build off of or make use of?

What would you have to learn in order to create a similar project?

How would you implement this project?

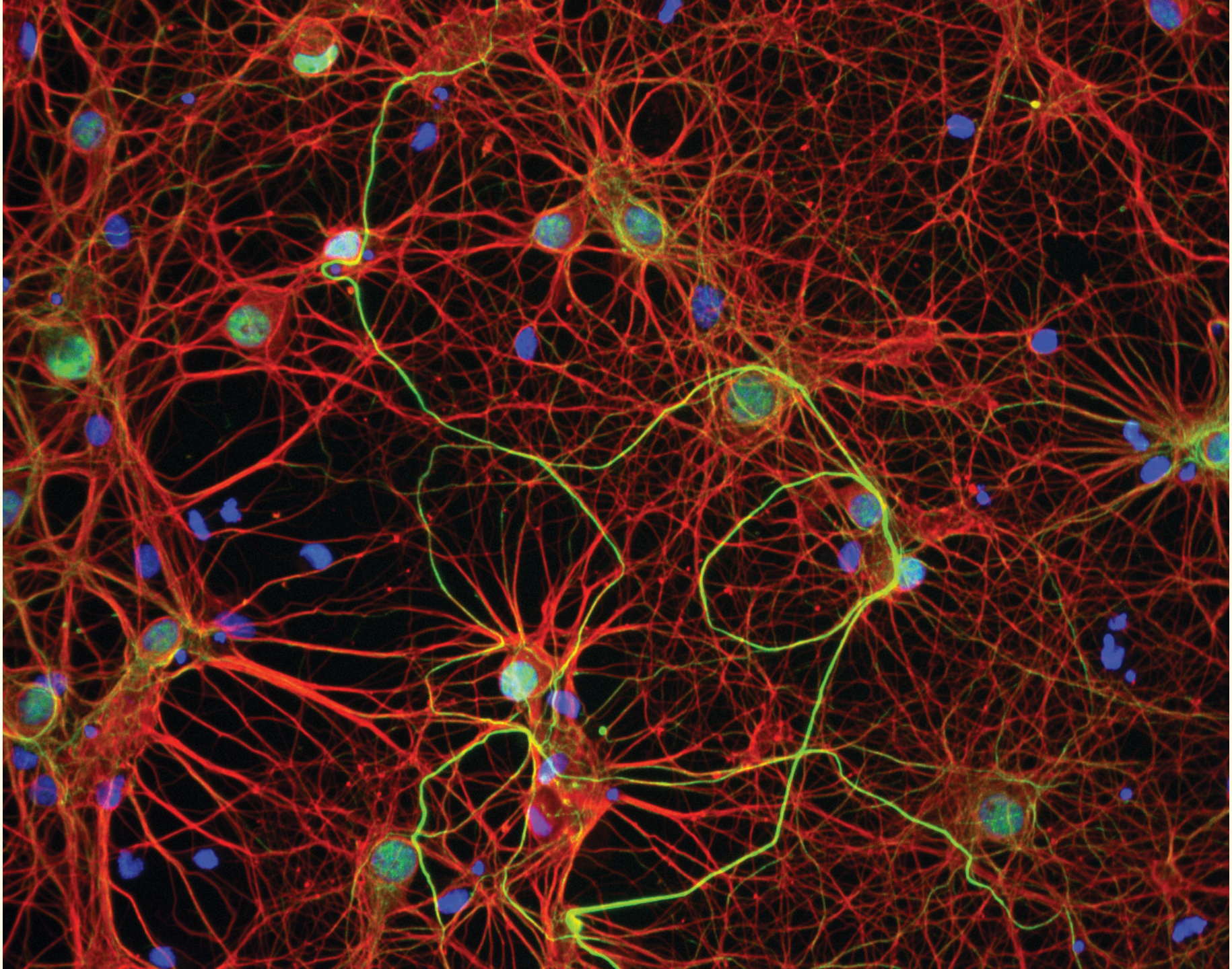
Overview of Neural Networks

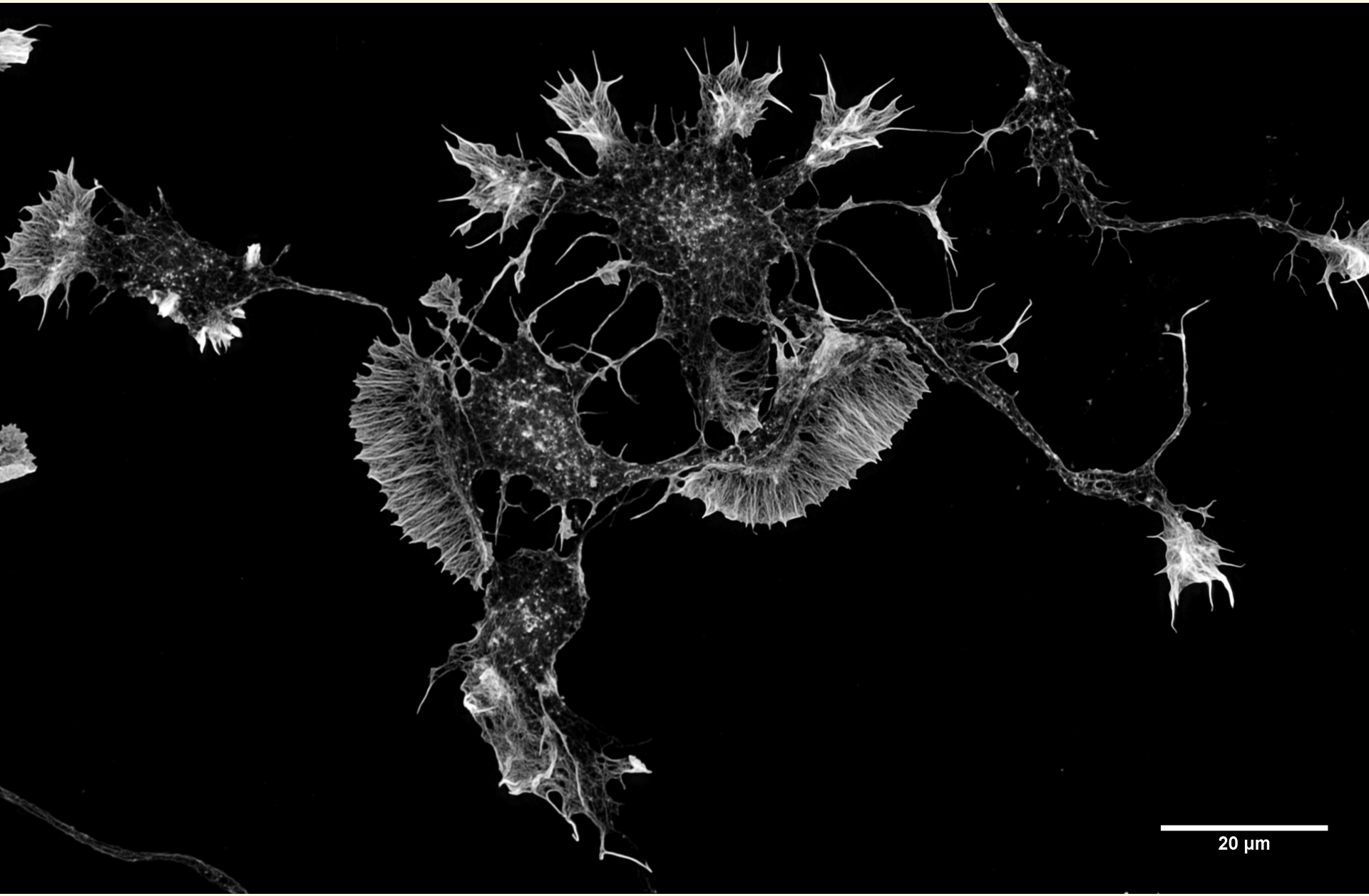
- Artificial Neural Networks (ANNs, or just NNs), and their many flavors, are all inspired from our understanding of how the brain works.

- 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

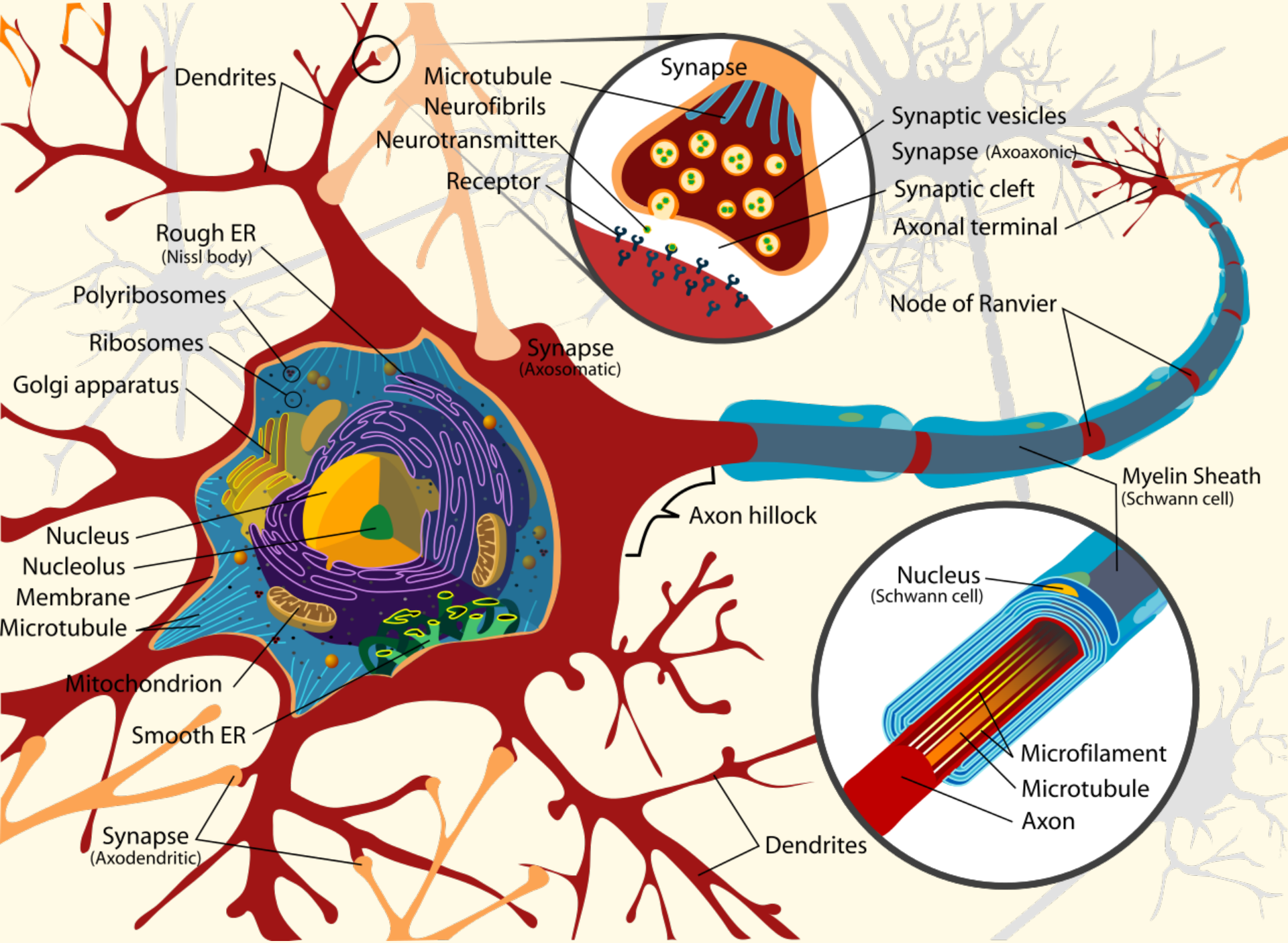
Human brain contains ~100,000,000,000 neurons, each of which is connected to between 10,000 to 100,000 other neurons. The process of responding to sensory input, managing our various bodily systems, controlling our motor functions, and thinking involves the constant firing of neurons.

Yet, these neurons fire relatively slowly - about 1/1000th a second (much slower that the speeds of your CPU.) 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. Advances in brain imaging (eg, MRI, Calcium Imaging) 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

Overview of Neural Networks

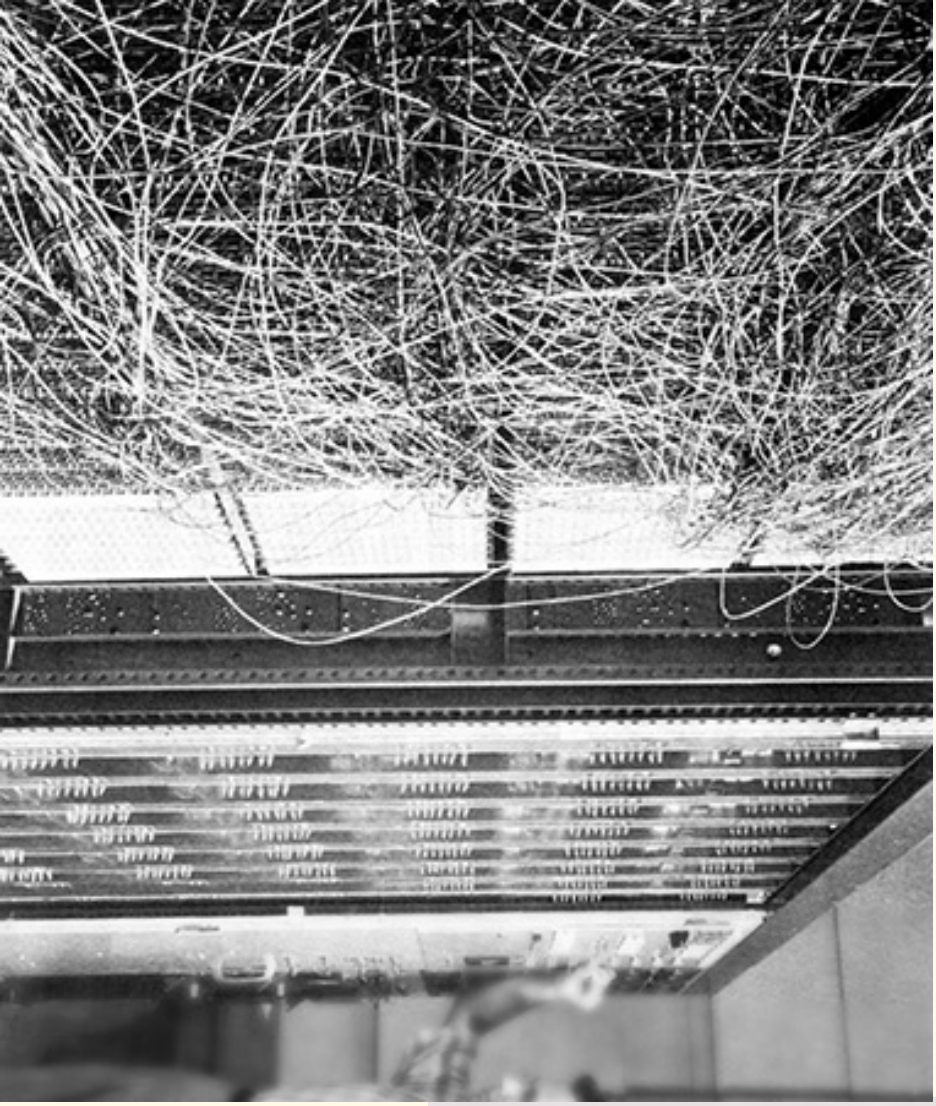
Interesting piece of history - one of the forefathers of NNs is Warren McCulloch, who came up with the first mathematical formulations of neural networks in 1943 while a professor at UIC in the Dept. of Psychiatry (in collaboration with Walter Pitts at UC).

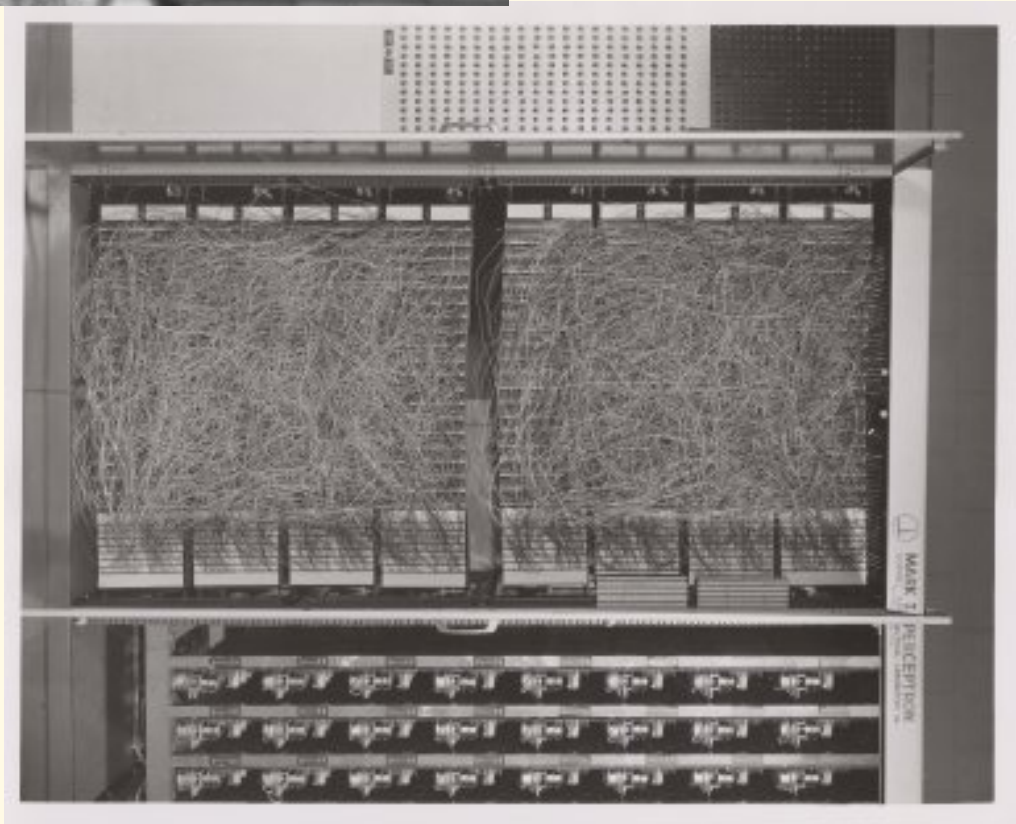
- “A Logical Calculus of Ideas Immanent in Nervous Activity,” McCulloch and Pitts, 1943

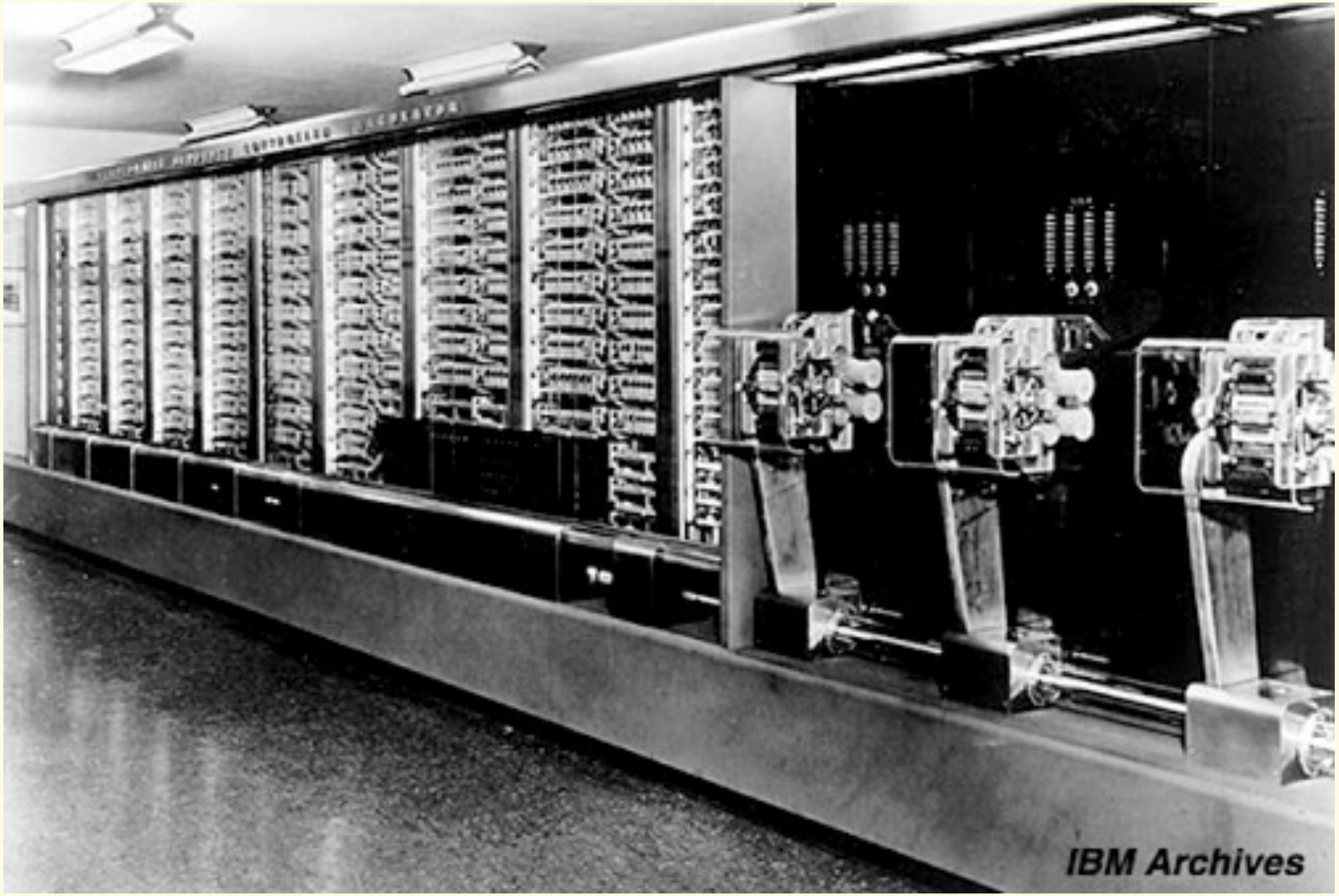
- http://nautil.us/issue/21/information/the-man-who-tried-to-redeem-the-world-with-logic

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 built it with analog circuits. It is capable of taking multiple inputs and classifying them into two different classes. If the data is linearly separable, then it will always converge to an accurate solution (although not always an optimal solution).

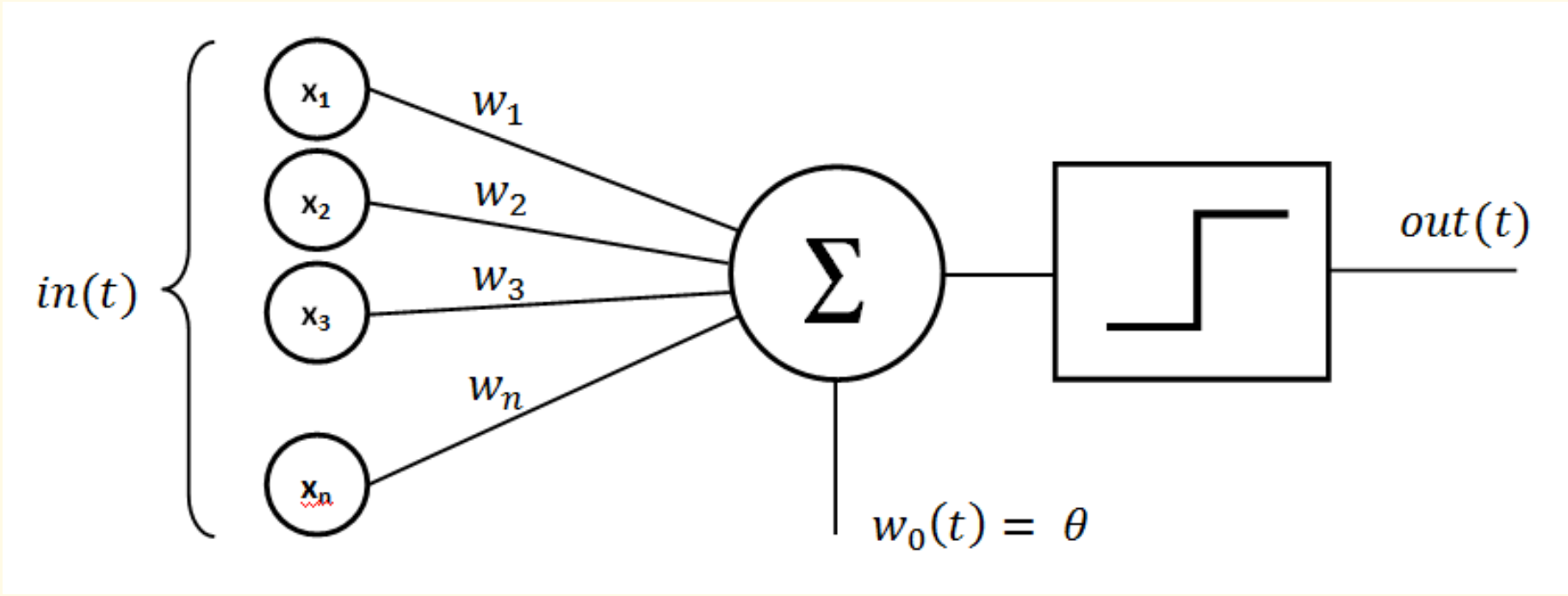



MASK 1
PERCEPTION


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. 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)
x1
x2
x3
xn
w1
w2
w3
wn
Σ
w0(t) = θ
out(t)

Overview of Neural Networks

In the training session, you start with labeled data — data for which you 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 what the perception does is change 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 preceptrons output always matches the desired output.

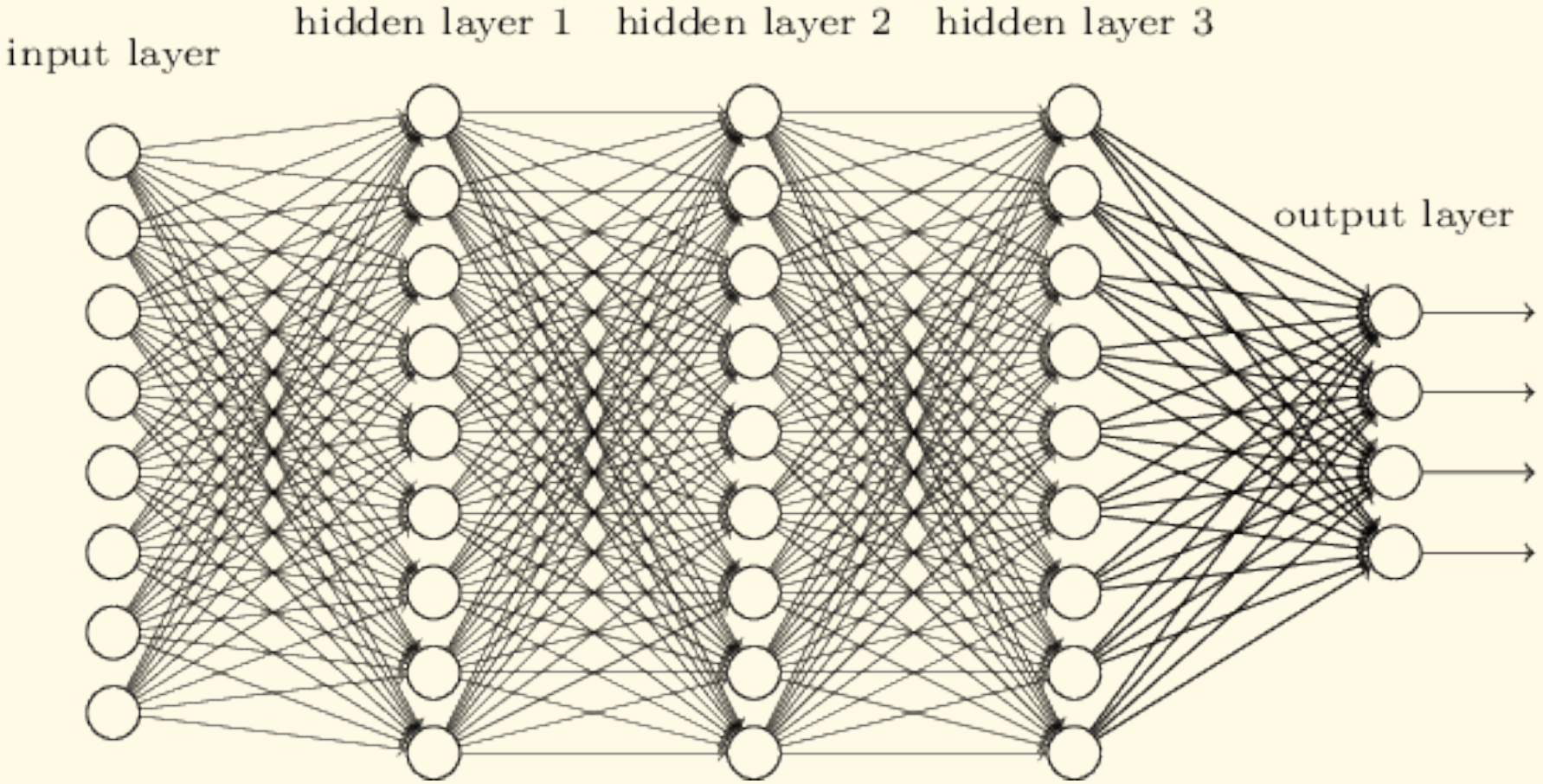
Then, assuming that your training set matches real-world conditions, inputting any new data should also result in the correct answer.

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 in this class is a form of the 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).

Overview of Neural Networks

- The Multilayer NN, (sometimes called a BPNN) is trained to detect particular features. If you have training data of many faces, you can label the data in terms of any features you want, and provide an output neuron for that feature.

- For example, one neuron for eye color, one for hair color, skin color, glasses, etc. The output vector will contain the amount that the NN thinks the photo is likely to be in each class.

Overview of Neural Networks

- For training, we can assume that a perfect answer would be [1,0,0,0] for correct eye color. And just as in the real-world, an item can belong to multiple categories, and our neurons would fire for these multiple categories, so can backpropogate multiple values simultaneously.

Overview of Neural Networks

- hidden layers:

can learn to invent new features not explicit to the data. One common example used in into 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 this assumption does not hold in the real-world, and it can be tricky to find a dataset that has wide enough coverage to generalize to real-world situations.

- Camouflaged tanks in trees

- 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

Recurrent NNs (RNNs)

- Attempt to model time series or sequential data (e.g., videos, texts, speech)
- Some of the *outputs* of the NN for one time period are used as *inputs* into the next time period. These neurons "learn" aspects of each time series and the changes between them.

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 identify rotation-invariant features
- Math is similar to commonly used mathematics in image processing kernels

Project 1

- Implement Deep Dream or Neural Style
- Learn by doing!
- Main goal is to understand:

  Tensorflow code (lots of examples, tutorials online to learn from)

  Neural Network architecture and algorithms
- Can work alone in groups of 2 or 3
- An interactive interface would be nice as well, plus adding your own twist to it... but main goal is get used to coding in python + TF

Lab

- Has everyone been able to run the MNIST example from the TF website?

- No?

    then work on it now!

- Yes?

    then:

        a) help others

        b) work through the other MNIST example

        c) figure out how to visualize neurons (the weights attached to the neurons) during (and at end of) training

        d) visualize the number that were misclassified – can you infer why the NN got confused?

Next Week

- Project 1 updates
- CNNs
- RNNs