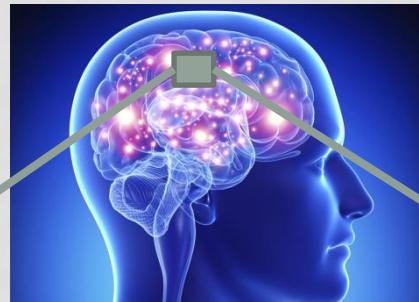




COMPUTING WITH RHYTHMS: THE SEARCH FOR OSCILLATORY DEEP NEURAL NETWORKS

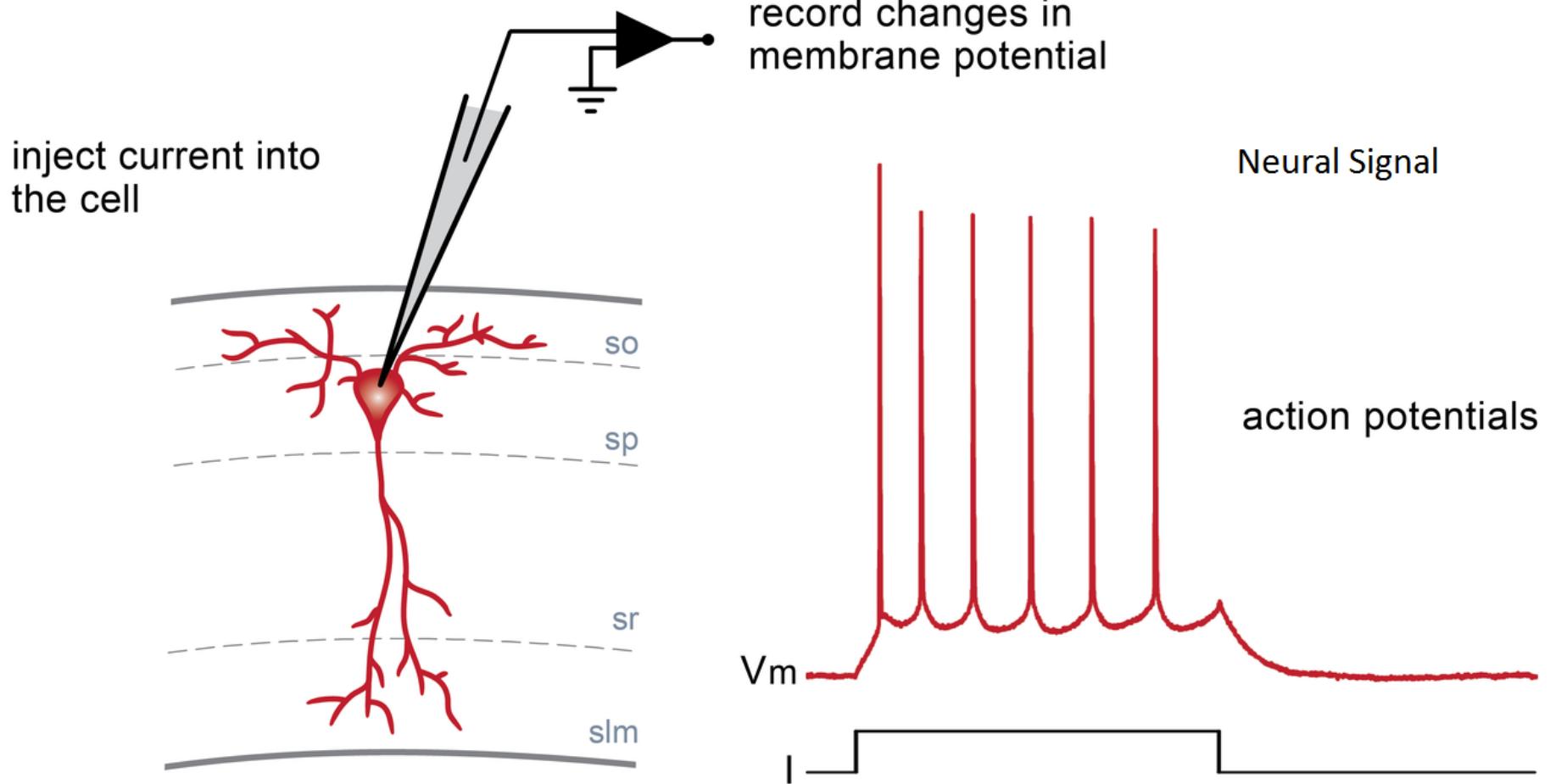
V. SRINIVASA CHAKRAVARTHY
IIT MADRAS.

PhD Student
Dipayan Biswas

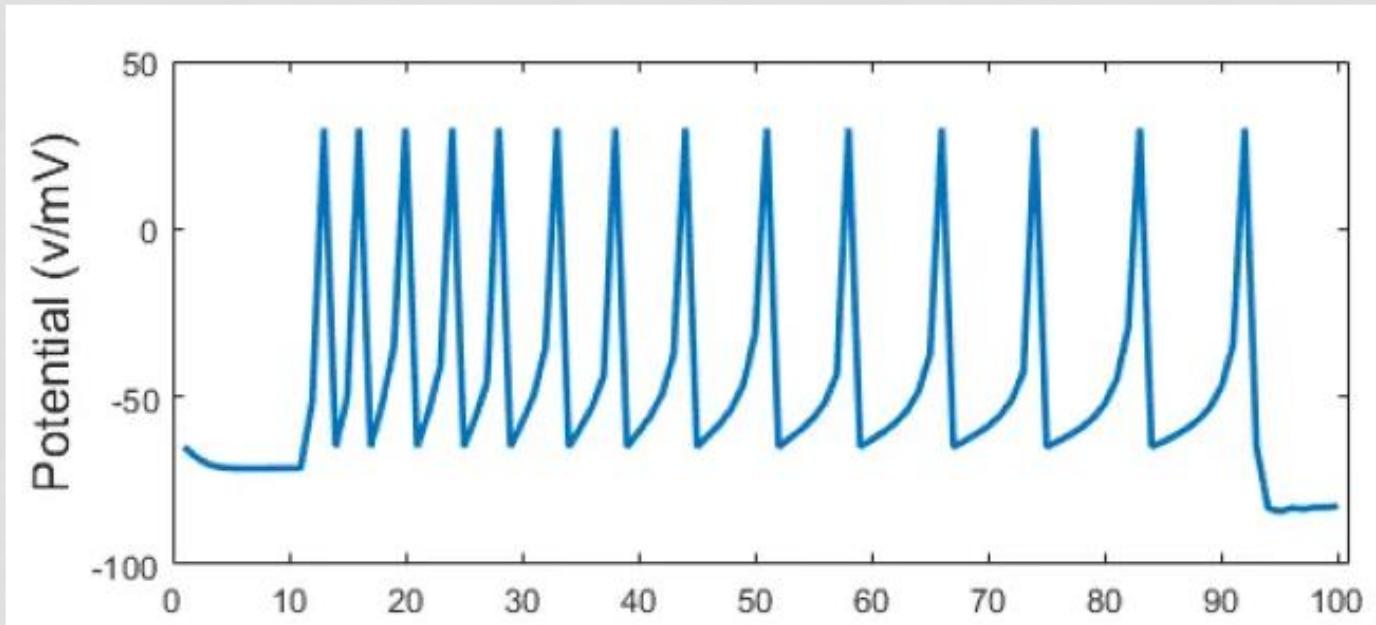


The Brain is a network of Neurons

Neurons communicate with each other by exchanging
electrical signals



WHAT IS THE NEURAL CODE?



How do you mathematically describe the information present in the electrical spiking activity of a neuron?

TWO THEORIES

Spike code

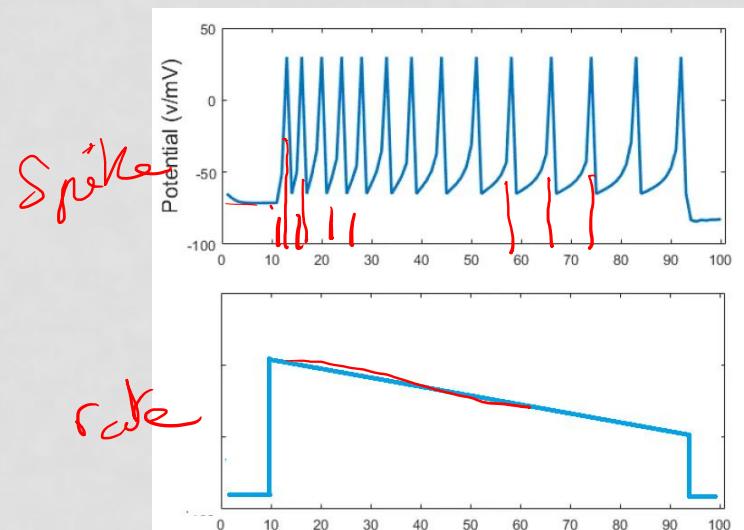
- Information is present in the times of occurrence of the spikes

$$s(t) = \sum_i \delta(t - t_i)$$



Rate Code

- Information is present in the frequency (rate) of the spikes



TWO KINDS OF NEURAL NETWORK MODELS

Spiking neuron networks

- Used often to model brain function

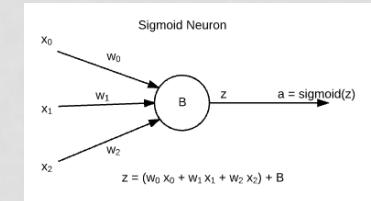
Rate coded neural networks

- Used more in artificial, engineering domains
- Eg. Deep neural networks

RATE-CODED NEURON MODELS

Used in
Deep
Neural
Networks

Sigmoidal neuron - static, rate-coded



$$\begin{aligned} y &= x, & x > 0 \\ &= 0, & \text{otherwise} \end{aligned}$$

Relu neuron - static, rate-coded

Long Short-Term Memory block:

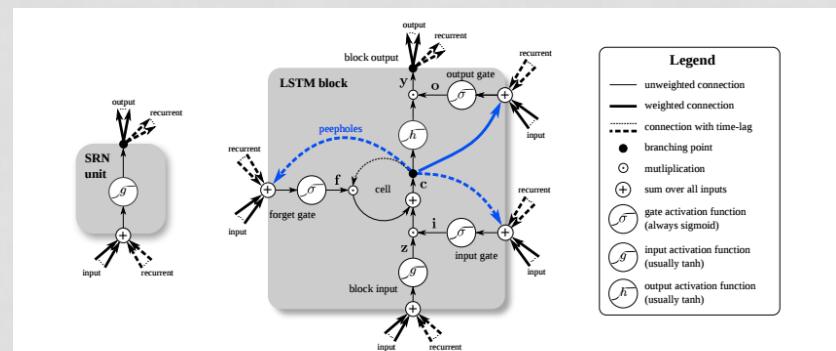


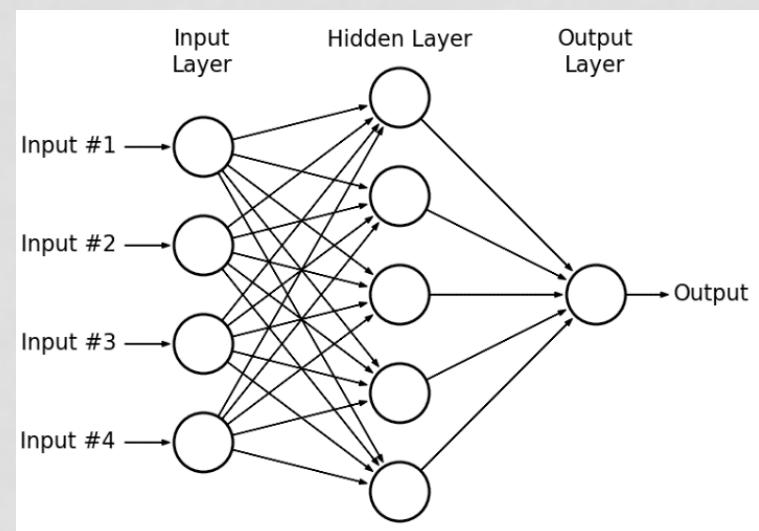
Figure 1. Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.

DEEP LEARNING/AI

- The aim of Deep learning/AI is to reproduce human intelligence.
- The approach is to achieve it through an implementation of the human brain
- Deep Neural networks are offered as an answer

NNS AND DEEP NNS

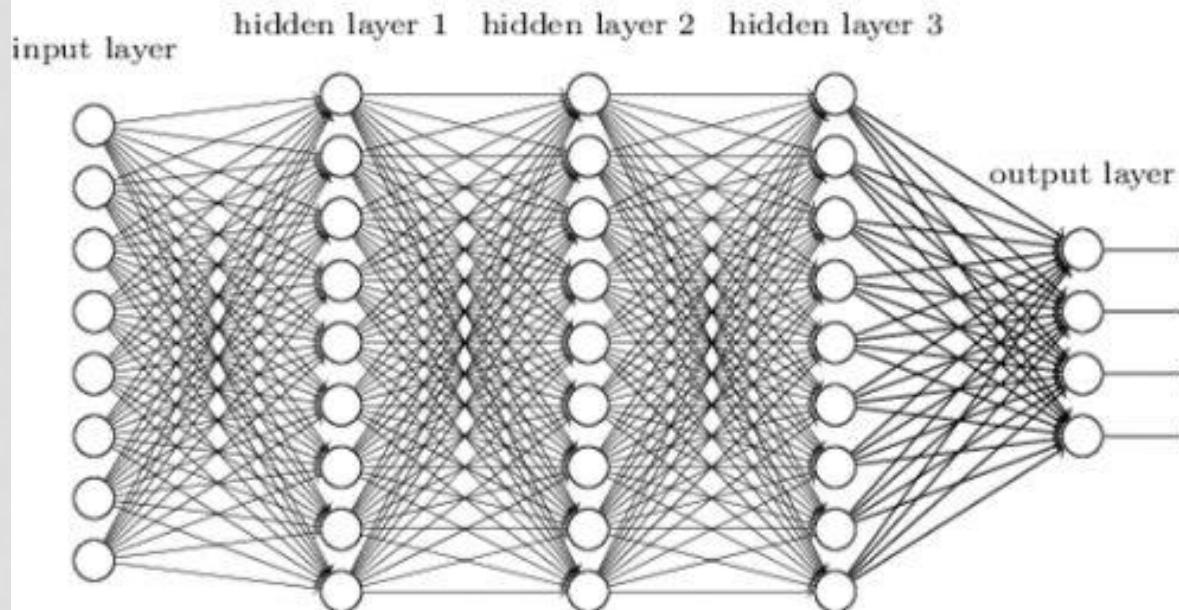
- 1980s-1990s: Multilayer perceptrons and backpropagation
- Deep learning – 2006
- Powerful computational properties.
- Universal approximation results



Multi-layer Perceptron

DEEP NEURAL NETWORK

Deep neural network

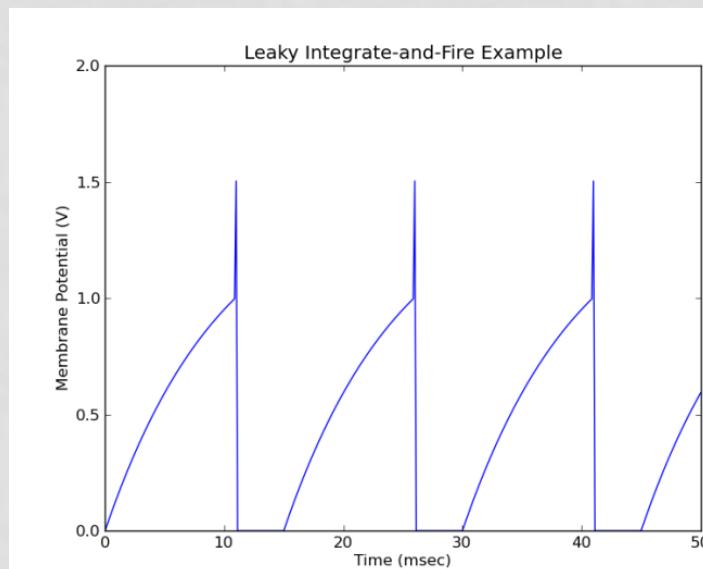
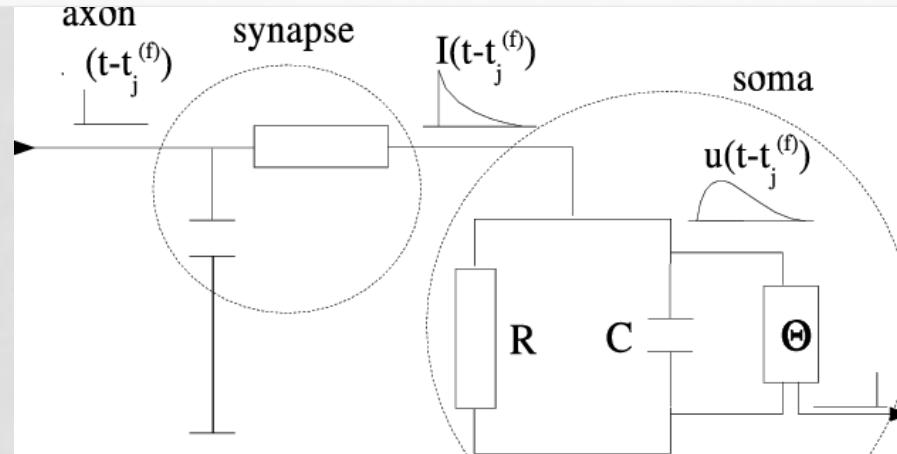


Has Universal Approximation properties
Can learn to map arbitrary input/output
Excellent applications in image pattern recognition,
Signal processing, robotics...

SPIKING NEURON MODELS

- Leaky Integrate and fire neuron (1 variable)
- Izhikevich neuron (2 variables)
- Hodgkin-Huxley neuron model (4 variables)
- Biophysical neuron models (large number of variables)

LEAKY INTEGRATE AND FIRE NEURON MODEL

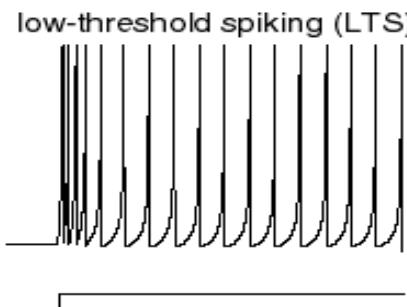
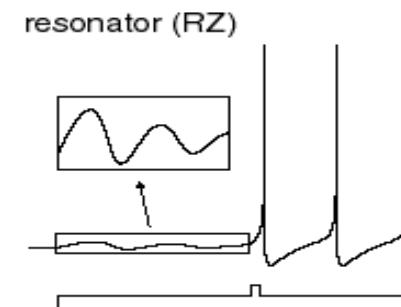
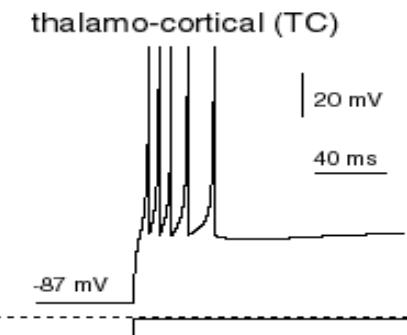
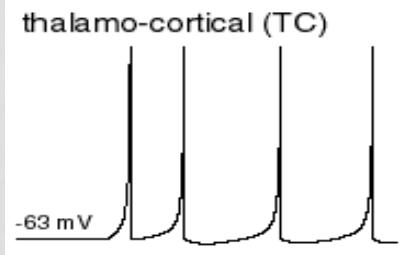
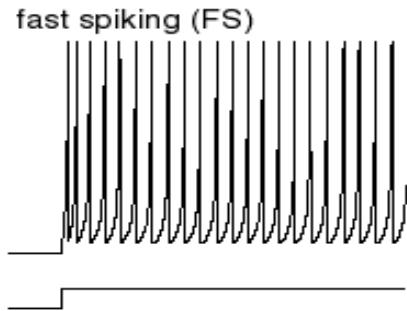
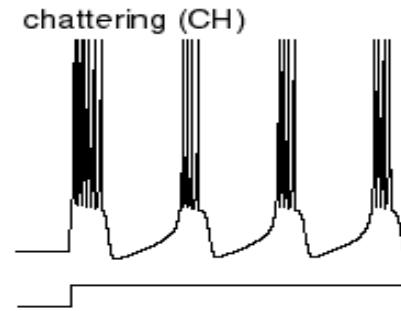
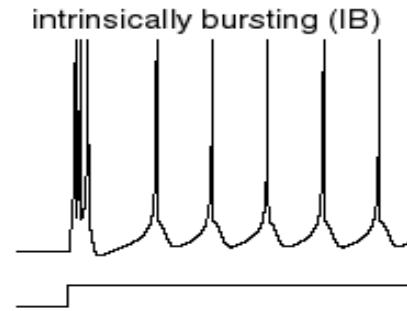
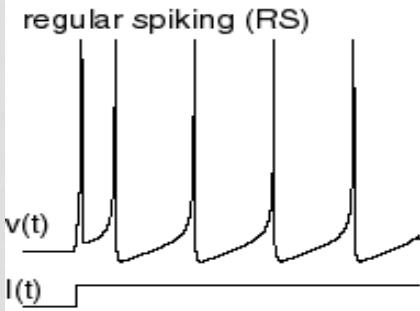
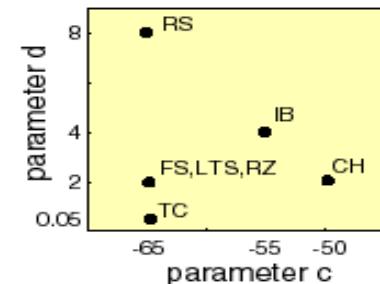
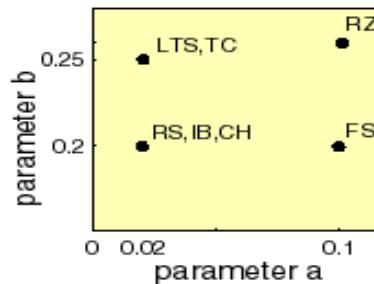
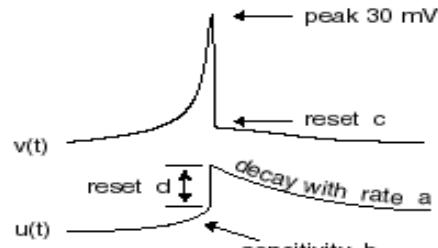


IZHIKEVICH NEURON MODEL

$$v' = 0.04v^2 + 5v + 140 - u + I$$

$$u' = a(bv - u)$$

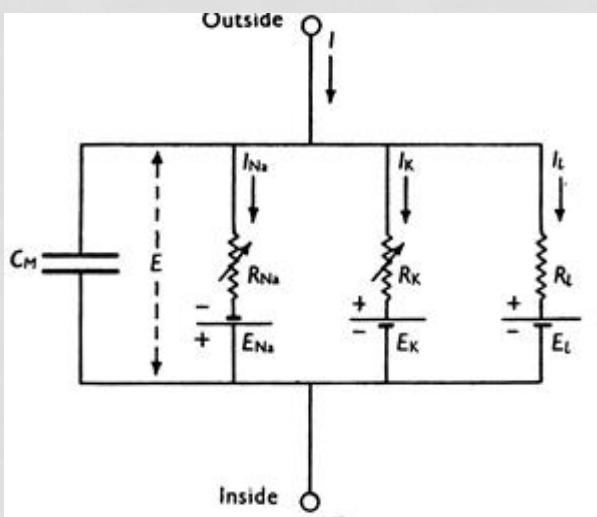
if $v = 30 \text{ mV}$,
then $v \leftarrow c$, $u \leftarrow u + d$



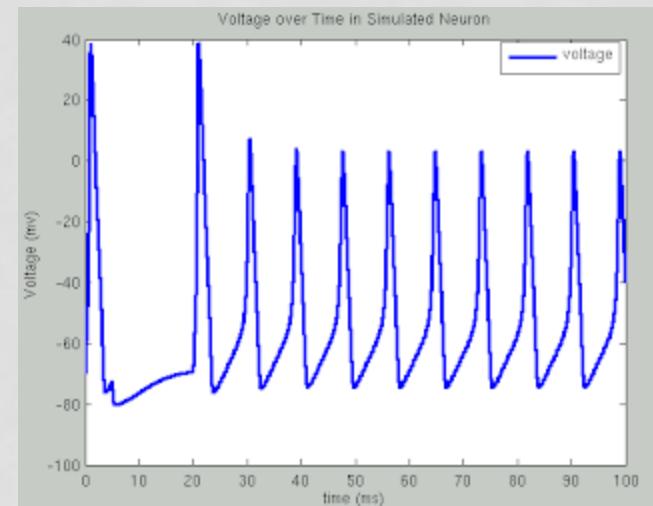
HODGKIN-HUXLEY NEURON MODEL

Hodgkin-Huxley Model (Example of a “point neuron” model)

$$\begin{aligned}\frac{dv}{dt} &= \frac{1}{C_m}[I - g_{Na}m^3h(v - E_{Na}) - g_Kn^4(v - E_K) \\ &\quad - g_L(v - E_L)] \\ \frac{dm}{dt} &= \alpha_m(v)(1 - m) - \beta_m(v)m \\ \frac{dn}{dt} &= \alpha_m(v)(1 - m) - \beta_m(v)m \\ \frac{dh}{dt} &= \alpha_h(v)(1 - h) - \beta_h(v)h\end{aligned}\tag{1}$$

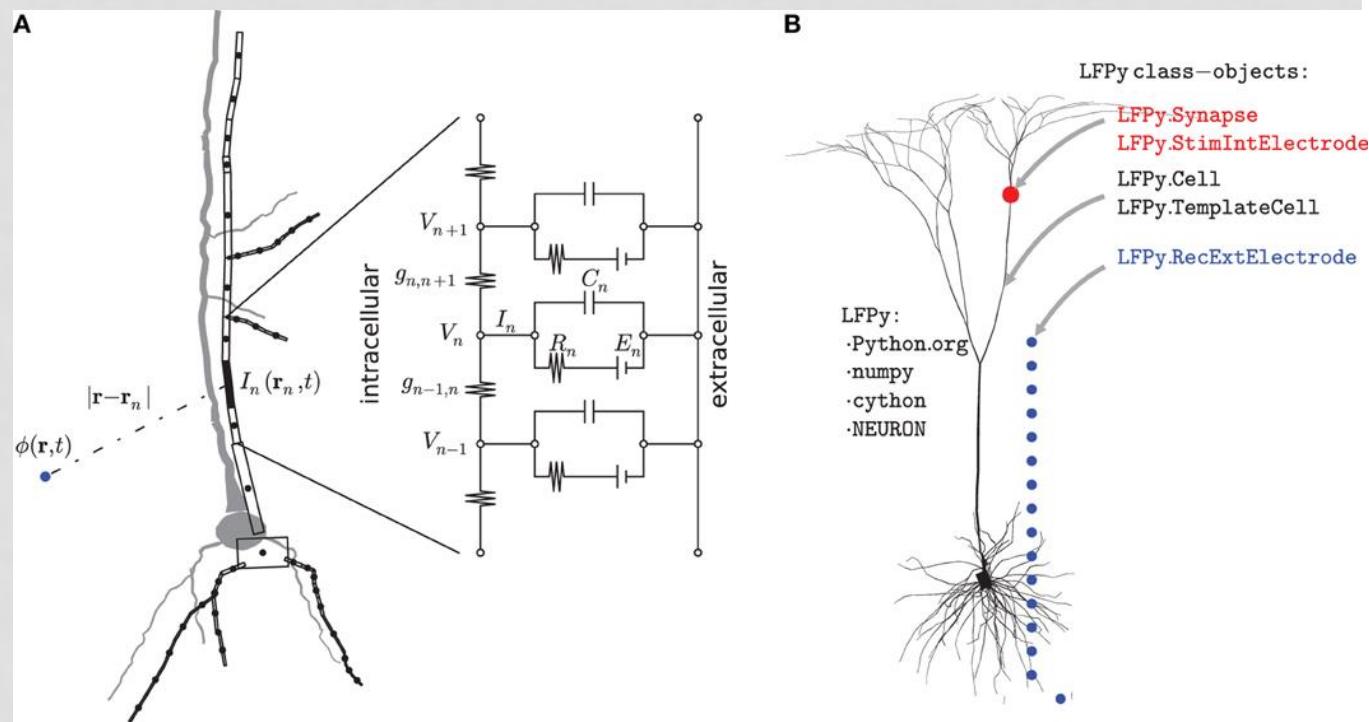


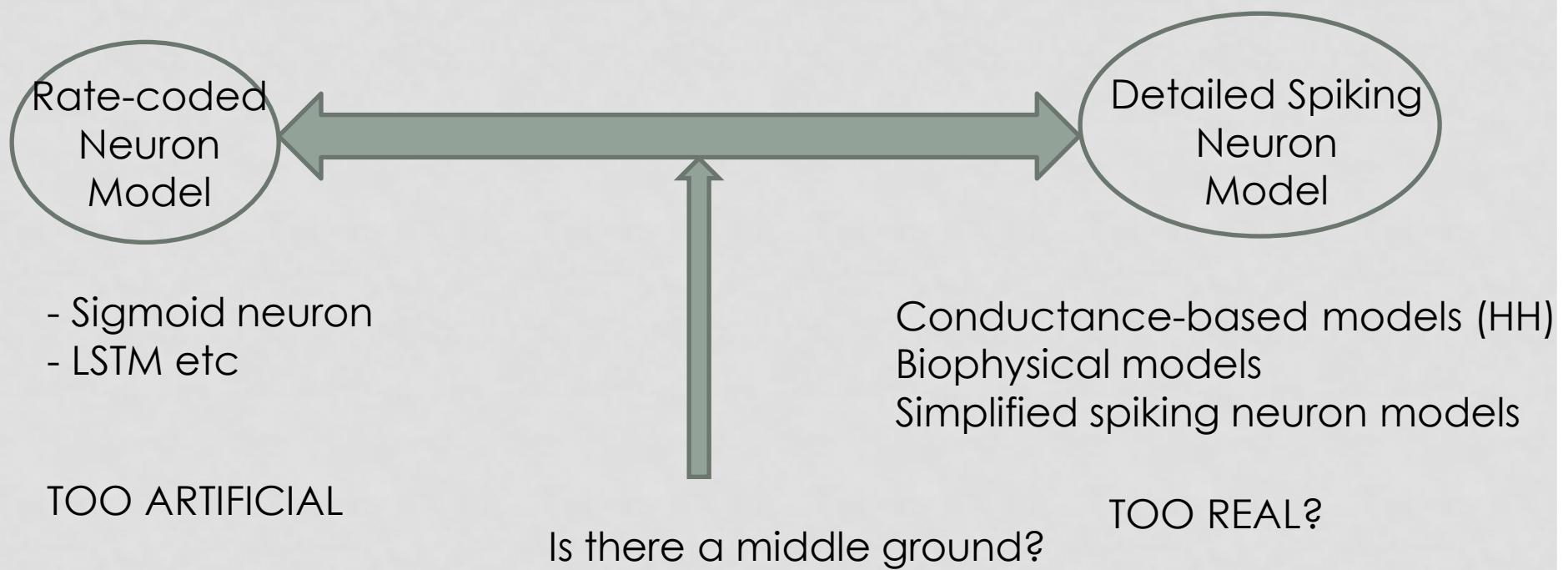
Hodgkin & Huxley
Were awarded the
Nobel Prize in 1963



BIOPHYSICAL NEURON MODEL

Simulate the entire neuronal arbor

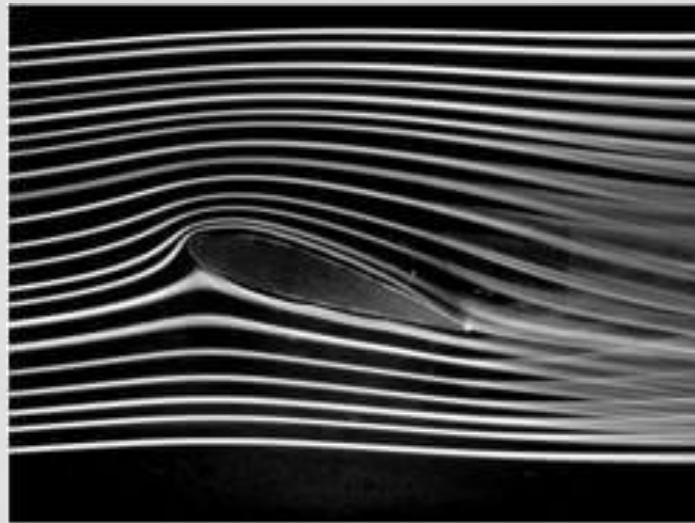




THE QUESTION OF THE RIGHT LEVEL

Example:

Aircraft wing in a wind tunnel

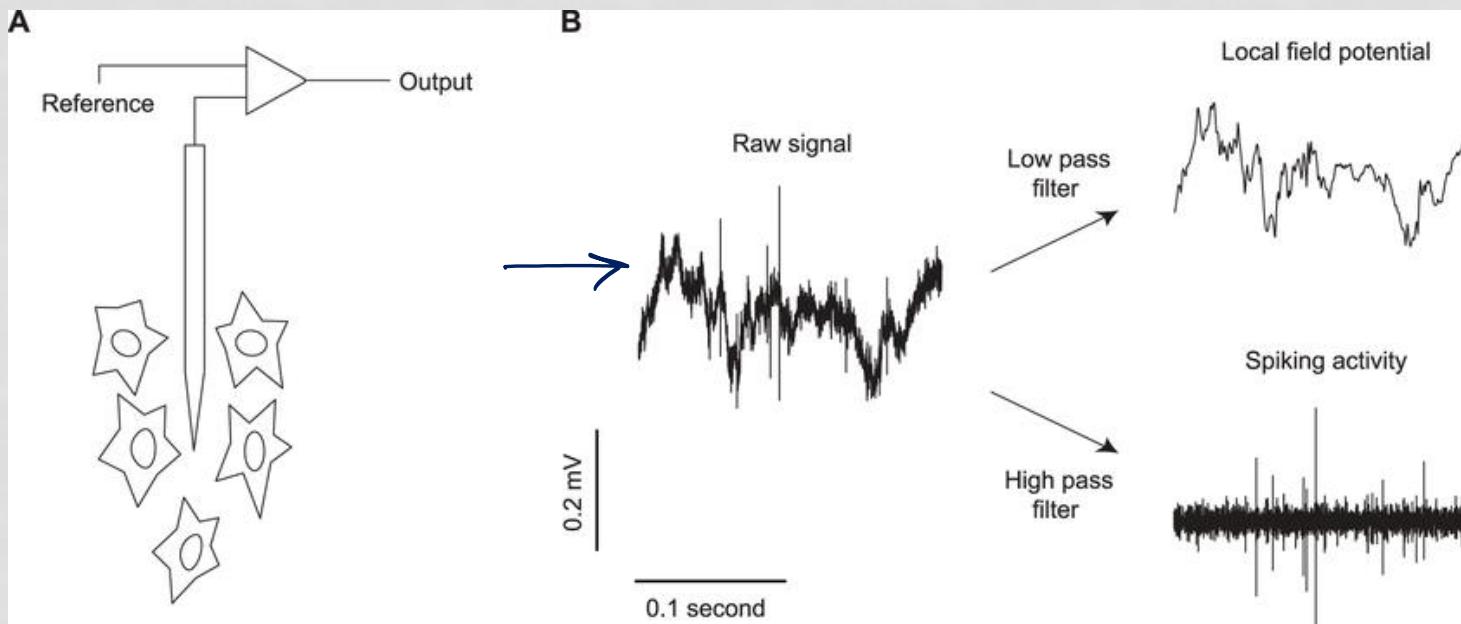


Study fluid flow using Navier Stokes Equation
FLUID is an abstraction

FLUID is NOT molecules

FROM SINGLE NEURON TO NEURAL ENSEMBLE

- Local field potential
 - activity of a local neural population
- Neural field models:
 - Coarse-grained models of spatio-temporal evolution of neural tissue (eg cortex)



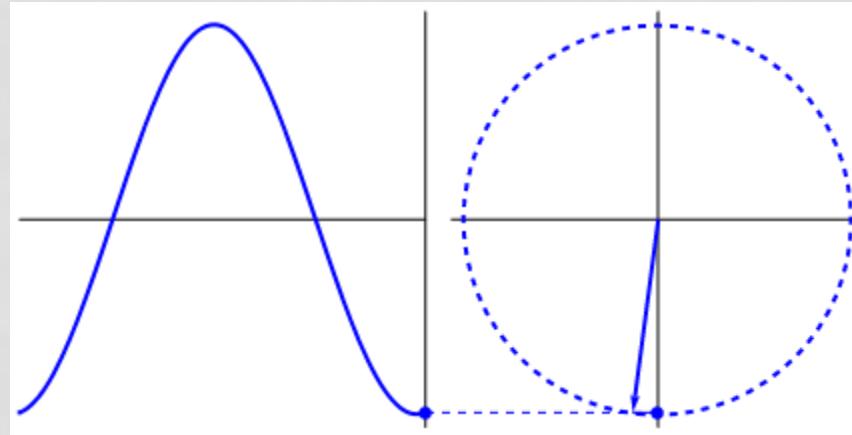
NEURAL "WAVES"

Electro Encephalo Gram (EEG)

Wave	Frequency (hertz)	Properties	EEG Trace
Delta	0.2-3	Lowest in frequency and arising in deep, dreamless sleep	
Theta	3-8	Important in strengthening synapses during learning	
Alpha	8-12	Predominant in a relaxed state when the brain is at rest and the eyes are closed	
Beta	12-30	Common when the brain is in an alert state, attentive and concentrating	
Gamma	30-120	Associated with information processing in the cerebral cortex, thinking and learning	

DO MODELS OF NEURAL OSCILLATORS OCCUPY THAT IDEAL MIDDLE GROUND?

“Synchronous activity of oscillating networks is now viewed as the critical ‘middle ground’ linking single-neuron activity to behavior” - (Buzsaki and Draghun, Science 2004)

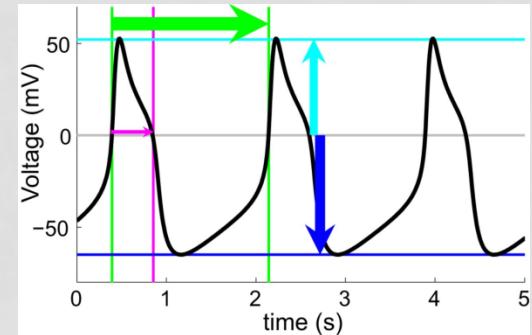


Ex
single ion channel
SINUSOID
Population of I.C.s

NEURAL OSCILLATOR MODELS (TYPICALLY 2 VARIABLE MODELS)

Wilson-Cowan model

$$\begin{aligned}\dot{x} &= -\alpha x + f(ax - by + \rho_x) \\ \dot{y} &= -\beta y + f(cx - dy + \rho_y).\end{aligned}$$



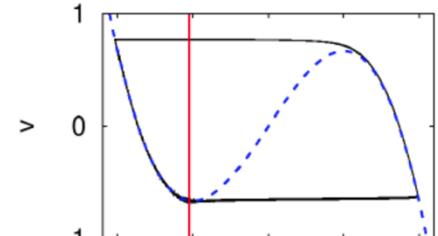
FitzHugh - Nagumo model

$$\frac{\partial V}{\partial t} = \frac{\partial^2 V}{\partial x^2} + (a - V)(V - 1)V - v \quad (1)$$

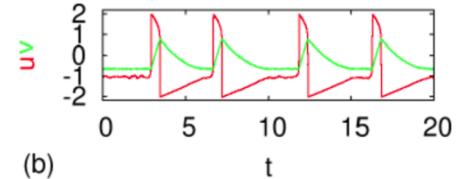
$$\frac{\partial v}{\partial t} = \epsilon(\beta V - \gamma v - \delta) \quad (2)$$

V = Voltage (fast variable)

v = v -gate (slow variable)

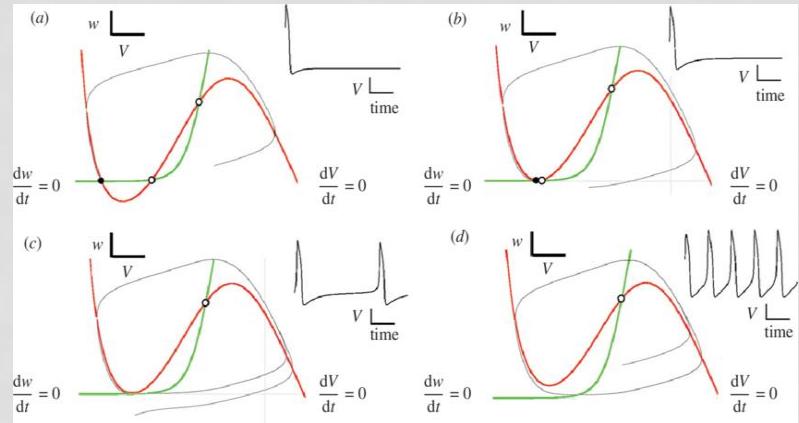


(a)



(b)

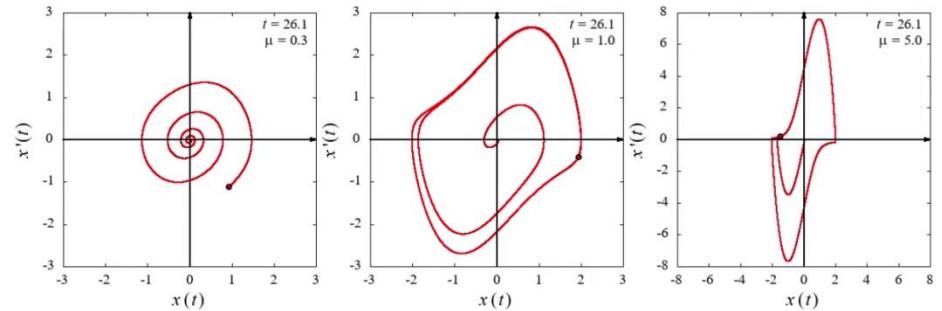
Morris-Lecarr model



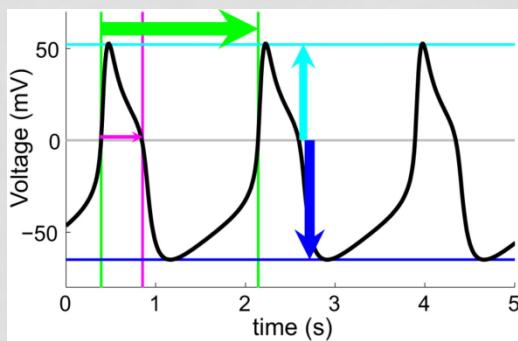
Van Der Pol model

Van der Pol equation

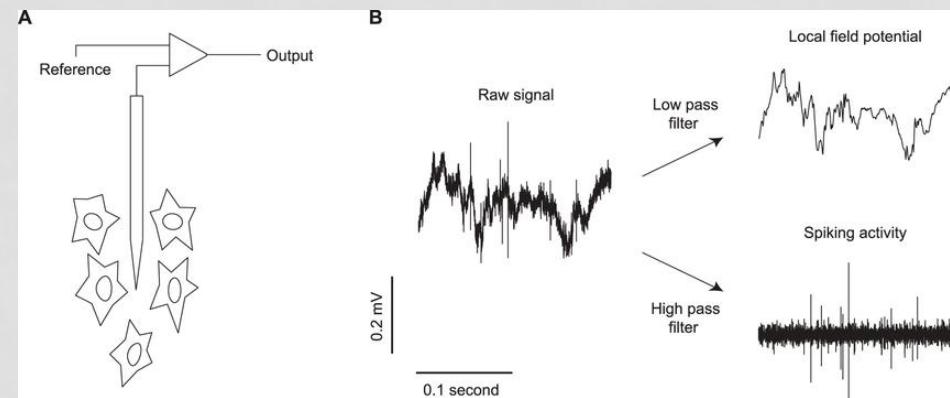
$$\frac{d^2x}{dt^2} - \mu(1-x^2) \frac{dx}{dt} + x = 0$$



Oscillation as a model of an Action Potential



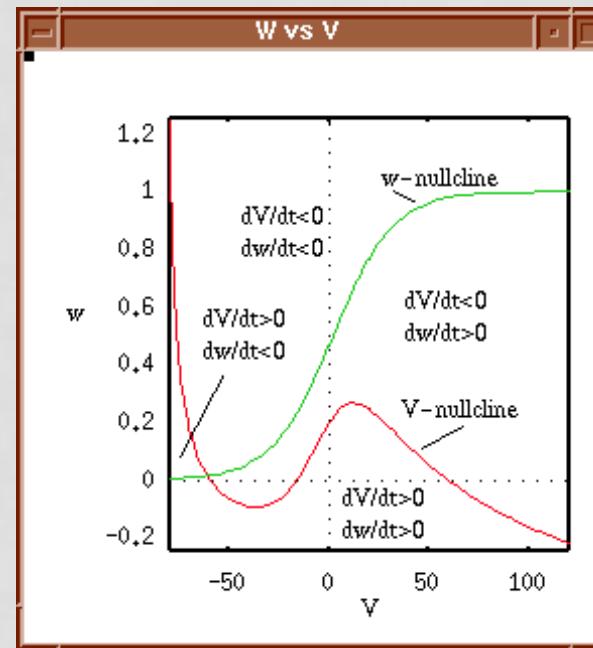
Oscillation as a model of Local Field Potential



The same models have been used for both

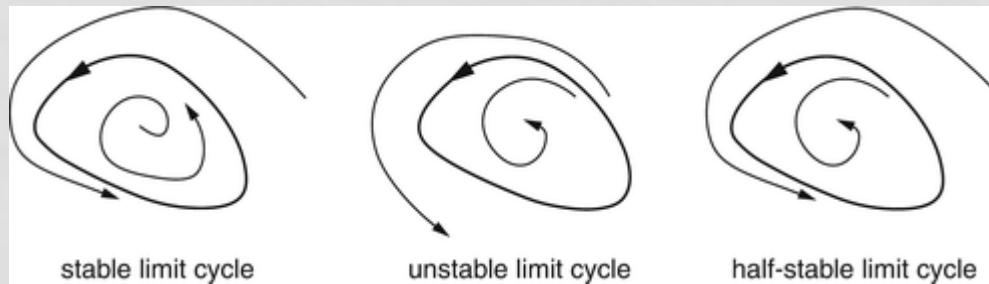
PHASE PLANE ANALYSIS

Two-variable form:
 $dx/dt = f(x,y)$
 $dy/dt = g(x,y)$



Can exhibit limit cycle oscillations

LIMIT CYCLES



Definition: A *limit cycle* is a closed trajectory in phase space having the property that at least one other trajectory spirals into it either as time approaches infinity or as time approaches negative infinity.

Can only be exhibited by nonlinear systems.

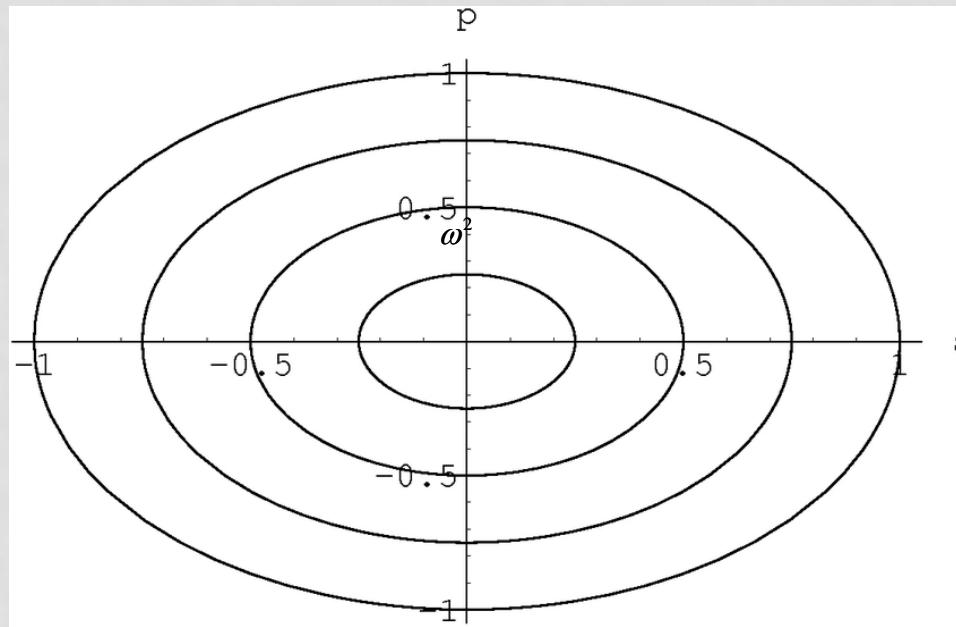
HARMONIC OSCILLATOR

Two-variable form:

$$\frac{dx}{dt} = f(x,y) = wy$$

$$\frac{dy}{dt} = -wx$$

$$\frac{d^2x}{dt^2} = -w^2 x$$



Solution is strictly dependent on the initial condition

Neural oscillations are Limit Cycle type

HOPF OSCILLATOR

$$\dot{x} = -y + \mu x(1 - x^2 - y^2)$$

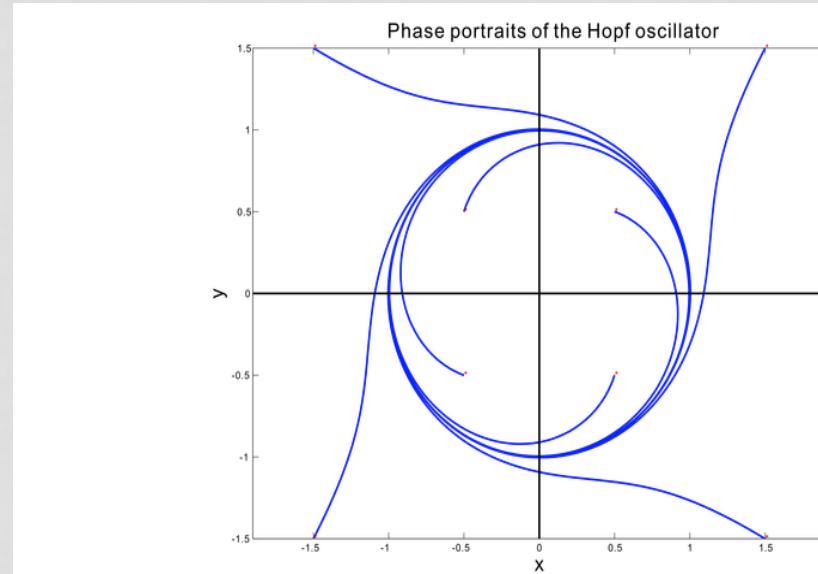
$$\dot{y} = x + \mu y(1 - x^2 - y^2)$$

In polar coordinates the equations have an elegant form.

Let $x = r \cos(\theta)$, $y = r \sin(\theta)$

$$\dot{r} = \mu r(1 - r^2)$$

$$\dot{\theta} = 1$$

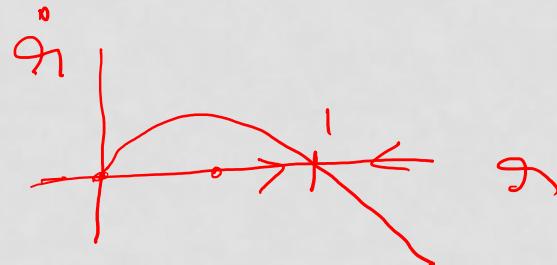


$$x\dot{x} + y\dot{y} = \mu(x^2 + y^2)(1 - x^2 - y^2)$$

$$= \mu r^2(1 - r^2)$$

$$\frac{1}{2} \frac{d(r^2)}{dt} = \mu r^2(1 - r^2)$$

$$\frac{dr}{dt} = \mu r(1 - r^2)$$



Now consider, $\theta = \arctan(y/x)$

- Differentiating both sides,

$$\dot{\theta} = \frac{1}{1 + \frac{y^2}{x^2}} \frac{x\dot{y} - y\dot{x}}{x^2}$$

$$x\dot{y} - y\dot{x} = x^2 + y^2 = r^2 \quad \dot{\omega} = 1$$

$$\dot{\theta} = \omega$$

IN COMPLEX VARIABLE FORM

$$\frac{dz}{dt} = i\omega z + (1 - |z|^2)z$$

$$z = x + iy$$

Let

$$z = re^{i\theta}$$

$$\frac{dz}{dt} = e^{i\theta} \frac{dr}{dt} + ire^{i\theta} \frac{d\theta}{dt}$$

$$e^{i\theta} \frac{dr}{dt} + ire^{i\theta} \frac{d\theta}{dt} = i\omega r e^{i\theta} + (1 - r^2) r e^{i\theta}$$

cancel $e^{i\theta}$,

$$\frac{dr}{dt} + ir \frac{d\theta}{dt} = i\omega r + (1 - r^2)r$$

$$\frac{dr}{dt} = (1 - r^2)r$$

$$\frac{d\theta}{dt} = \omega$$

Equating real and
imaginary parts:

NO NONLINEARITY → SHM

$$\frac{dz}{dt} = i\omega z + (1 - |z|^2)z$$

Eliminating nonlinearity...

$$\frac{dz}{dt} = i\omega z$$

$$\frac{dx}{dt} + i \frac{dy}{dt} = i\omega(x + iy) = \omega(ix - y)$$

$$\frac{dx}{dt} = \omega y$$

$$\frac{dy}{dt} = -\omega x$$

$$A = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}$$

$$\lambda = \pm i\omega$$

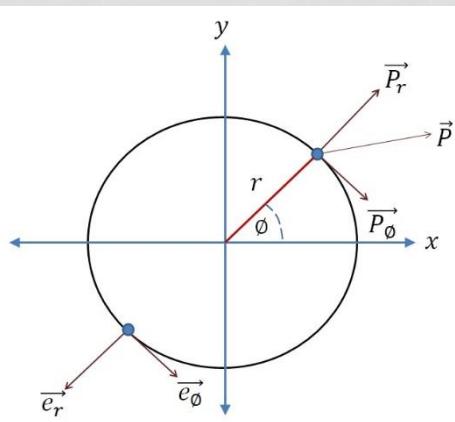
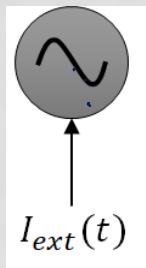
RESEARCH AGENDA

- To construct general oscillatory neural networks using Hopf Oscillators:
 - Oscillatory Hopfield networks (done) ✓
 - Oscillatory Auto Encoders (done) ✓
 - Oscillatory Self-organizing Maps (SOMs) ✓
 - Oscillatory Deep networks ✓
 - Oscillatory Convolutional Networks etc etc ✓

COMPLEX ADAPTIVE HOPF OSCILLATOR:

$$\dot{z} = z(\mu + i\omega - |z|^2) + \varepsilon I_{ext}(t)$$

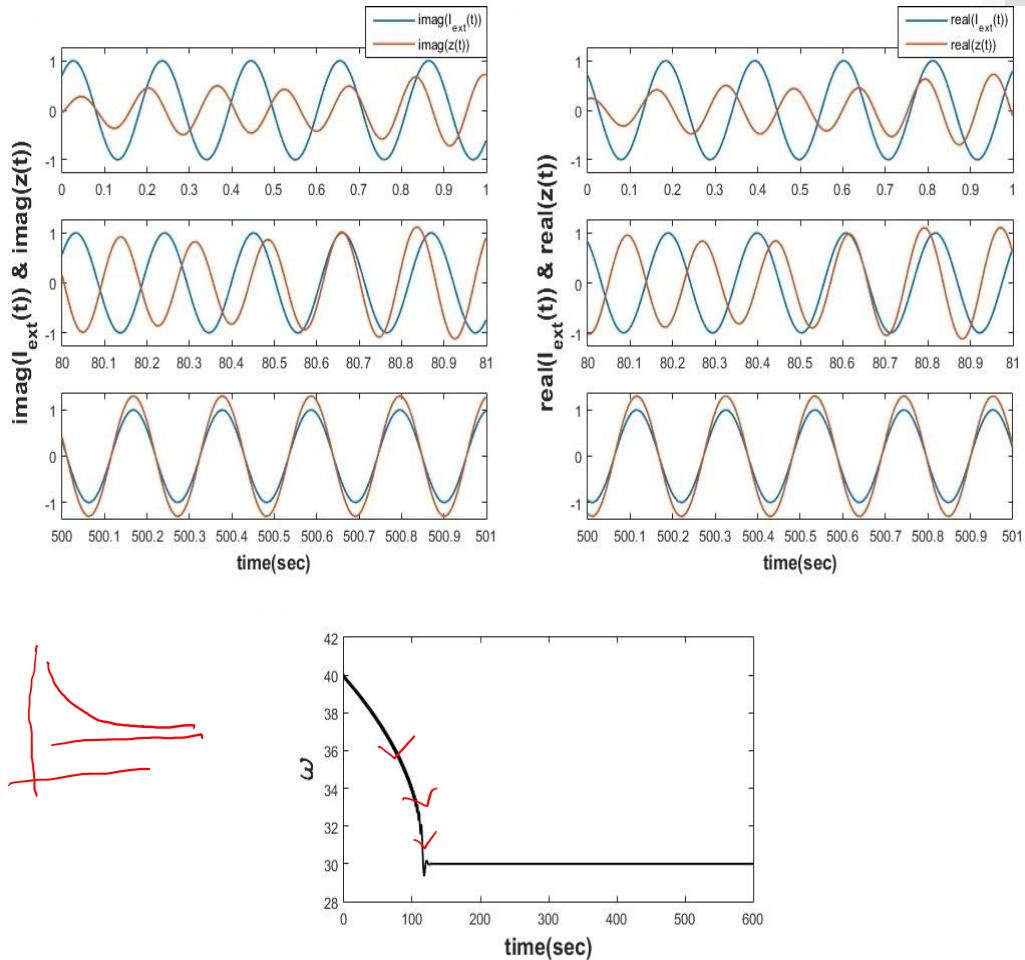
$$\dot{\omega} = -\varepsilon I_{ext}(t) \sin \phi$$



$$I_{ext}(t) = I_0 e^{i(\omega_0 t + \varphi)}$$

$$\dot{\omega} = -\varepsilon (\text{real}(I_{ext}(t)) \sin \phi - \text{img}(I_{ext}(t)) \cos \phi)$$

$$\dot{\omega} = -\varepsilon I_0 \sin(\phi - \omega_0 t - \varphi)$$



Righetti et al., 2005

(Biswas & Chakravarthy, 2020)

In this scenario, $I_{ext}(t)$ is a complex sinusoidal signal. It is straight forward to derive the learning rule for the natural frequency of the oscillator if eq. 2a is represented in the Cartesian and polar coordinate forms, respectively, as follows:

$$\begin{aligned}\dot{x} &= (\mu - r^2)x - \omega y + \varepsilon I_0 \cos(\omega_0 t + \varphi) \\ \dot{y} &= (\mu - r^2)y + \omega x + \varepsilon I_0 \sin(\omega_0 t + \varphi)\end{aligned}\tag{2a1}$$

$$\begin{aligned}\dot{r} &= (\mu - r^2)r + \varepsilon I_0 \cos(\omega_0 t + \varphi - \emptyset) \\ \dot{\emptyset} &= \omega + \underbrace{\frac{\varepsilon I_0}{r} \sin(\omega_0 t + \varphi - \emptyset)}\end{aligned}\tag{2a2}$$

In the phase plane representation, it can be observed from eq. 2a2 that the influence caused by the input perturbation on the oscillator phase is $\frac{\varepsilon I_0}{r} \sin(\omega_0 t - \emptyset)$. Whereas from the Cartesian coordinate system



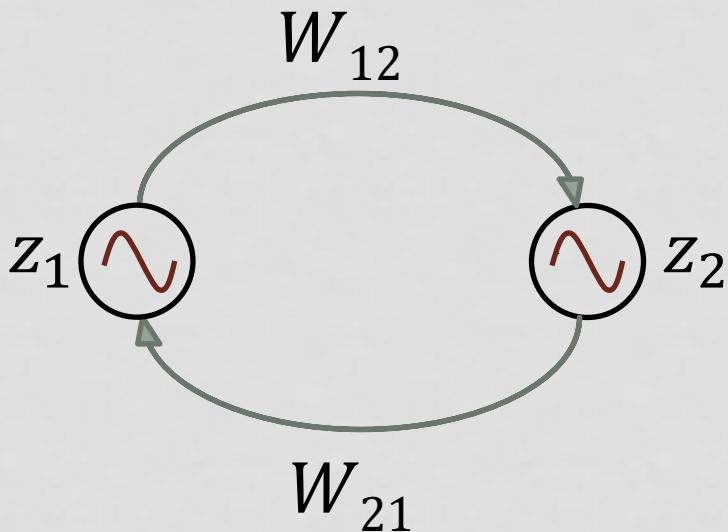
$$\phi = \omega t + \xi$$

$$i = -\epsilon \sin(\omega t + \xi - \omega_0 t - \varphi)$$

PROBLEM#1

A PAIR OF COUPLED HOPF OSCILLATORS

$$\begin{aligned}\dot{z}_1 &= (\mu - |z_1|^2)z_1 + i\omega_1 z_1 + W_{21} \text{real}(z_2) \\ \dot{z}_2 &= (\mu - |z_2|^2)z_2 + i\omega_2 z_2 + W_{12} \text{real}(z_1)\end{aligned}$$



What is the effect of the Coupling Strength (W) on the Phase Difference (ψ)?



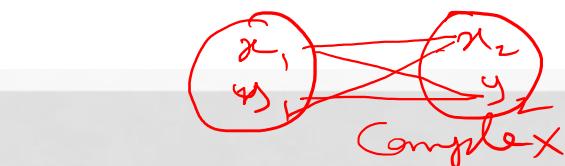
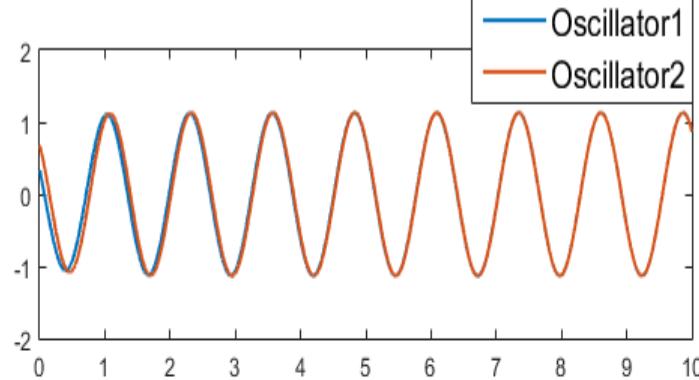
TWO HOPF OSCILLATORS

SAME FREQUENCY ($\omega_1 = \omega_2$), REAL COUPLING

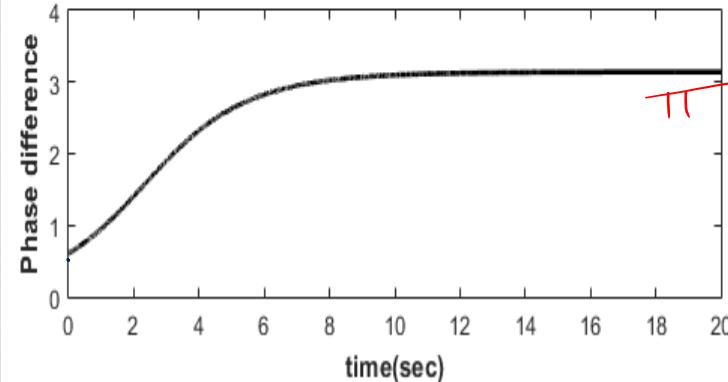
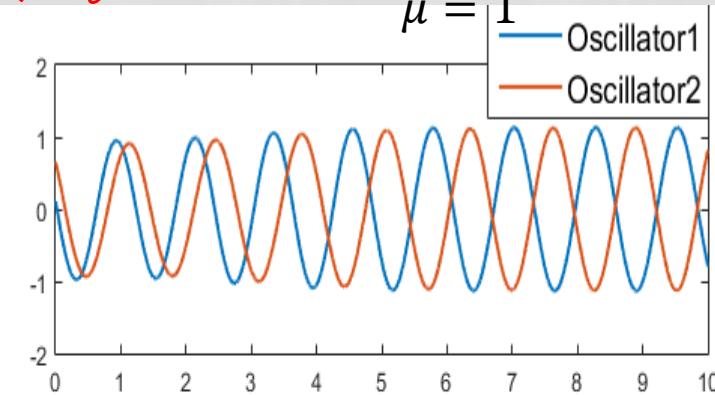
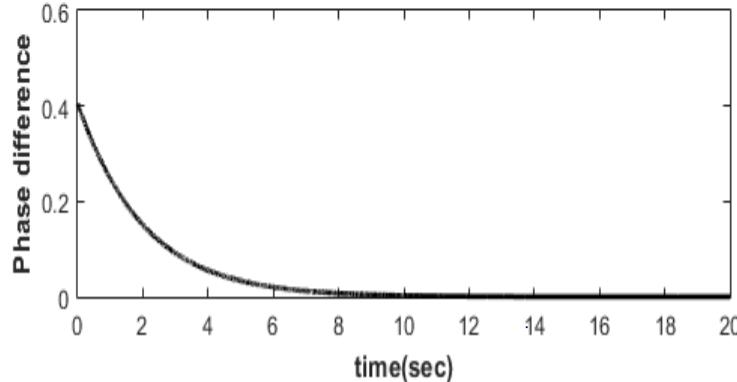


$$\begin{aligned}\omega_1 &= \omega_2 = 5 \\ W_{21} &= W_{12} = 0.5 \\ \mu &= 1\end{aligned}$$

$\omega_1 = \omega_2$
Symmetric coupling



$$\begin{aligned}\omega_1 &= \omega_2 = 5 \\ W_{21} &= W_{12} = -0.5 \\ \mu &= 1\end{aligned}$$



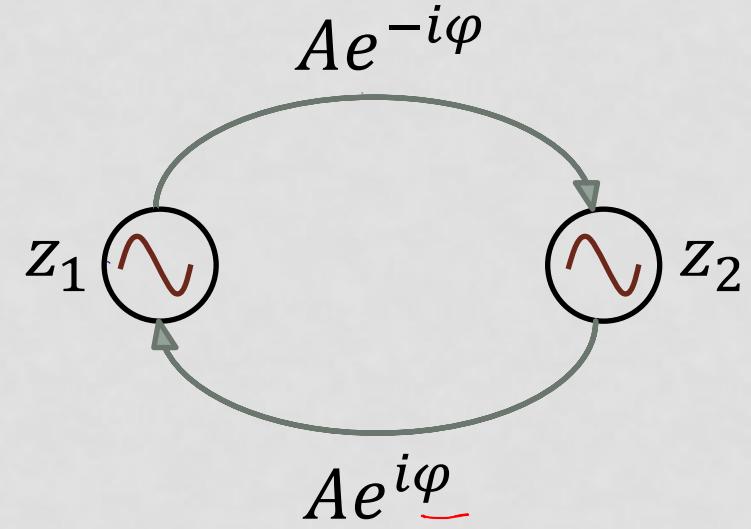
TWO COUPLED HOPF OSCILLATORS: SAME FREQUENCIES ($\omega_1 = \omega_2$), COMPLEX COUPLING

- Complex variable representation of Hopf oscillator.

$$\dot{z} = (\mu - |z|^2)z + i\omega z$$

Where, $z = x + iy = re^{i\theta}$

- Two oscillators coupled through complex lateral weights.



$$\dot{z}_1 = (\mu - |z_1|^2)z_1 + i\omega_1 z_1 + \underline{Ae^{i\varphi}} z_2$$

$$\dot{z}_2 = (\mu - |z_2|^2)z_2 + i\omega_2 z_2 + \underline{Ae^{-i\varphi}} z_1$$

THEORETICAL EXPLANATION

- Polar coordinate representation:

$$\dot{r}_1 = (\mu - r_1^2)r_1 + A \underline{r_2} \cos(\theta_2 - \theta_1 + \varphi)$$

$$\dot{\theta}_1 = \omega_1 + A \frac{\underline{r_2}}{r_1} \sin(\theta_2 - \theta_1 + \varphi)$$

$$\dot{r}_2 = (\mu - r_2^2)r_2 + A \underline{r_1} \cos(\theta_1 - \theta_2 - \varphi)$$

$$\psi = \theta_1 - \theta_2$$

$$\dot{\theta}_2 = \omega_2 + A \frac{\underline{r_1}}{r_2} \sin(\theta_1 - \theta_2 - \varphi)$$

$$\theta_1 - \theta_2 =$$

- From which:

$$\underline{\omega_1 = \omega_2}$$

arbitrary?

$$\dot{\psi} = \dot{\theta}_1 - \dot{\theta}_2 = (\omega_1 - \cancel{\omega_2}) - \frac{A \sin(\psi - \varphi)}{r_1 r_2} (r_1^2 + r_2^2)$$

- Assuming $\omega_1 = \omega_2$, at steady state:

$$\dot{\psi} = 2A \sin(\varphi - \psi) = 0$$

$$r_1 = r_2 = \sqrt{\mu}$$

- Which ensures the phase difference $(\theta_1 - \theta_2)$ between the two oscillators to be φ : the angle of the complex coupling weight.

NUMERICAL RESULTS

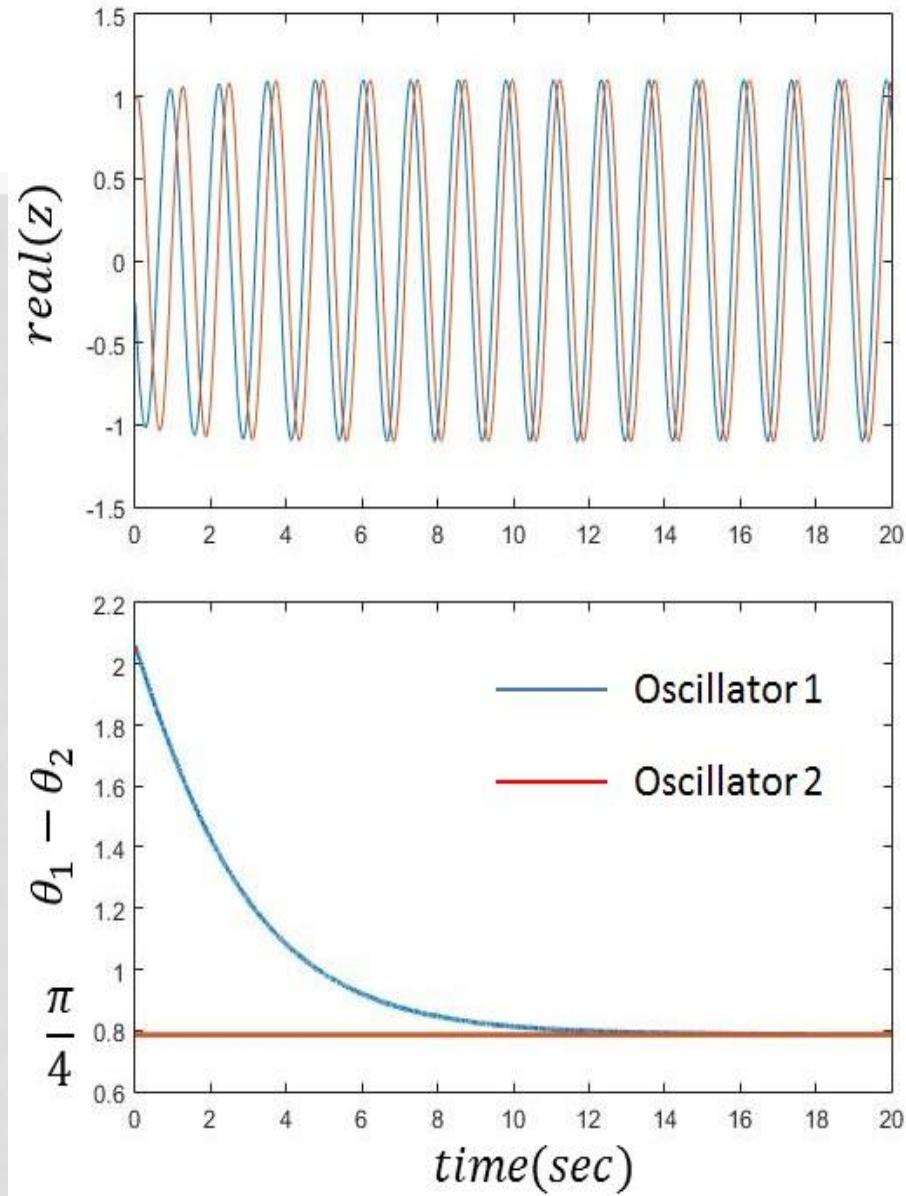
- Parameters:

$$A = 0.2, \varphi = \frac{\pi}{4}$$

$$\omega_1 = \omega_2 = 5$$

Where: $z_1 = r_1 e^{i\theta_1}$ and

$$z_2 = r_2 e^{i\theta_2}$$



COUPLING TWO HOPF OSCILLATORS NEARBY FREQUENCIES, REAL COUPLING

- When two Hopf oscillators, with slightly different frequencies ($\omega_1 \approx \omega_2$) are coupled, for a sufficiently strong coupling strength, the two oscillators come to a common frequency (ω) such that ($\omega_1 < \omega < \omega_2$), with oscillator #2 leading oscillator #1.

$$\omega_1 = \omega_2 + \epsilon \quad \epsilon \ll 1$$

$$\theta_1 > \theta_2$$

$$\begin{matrix} \omega_1 \rightarrow \omega \\ \omega_2 \end{matrix}$$



$$\omega_1 > \omega > \omega_2$$

$$\begin{matrix} \delta \cdot 1 Hz \\ \theta \\ L \\ P \end{matrix}$$

$$\begin{matrix} R \\ 10^0 Hz \end{matrix}$$

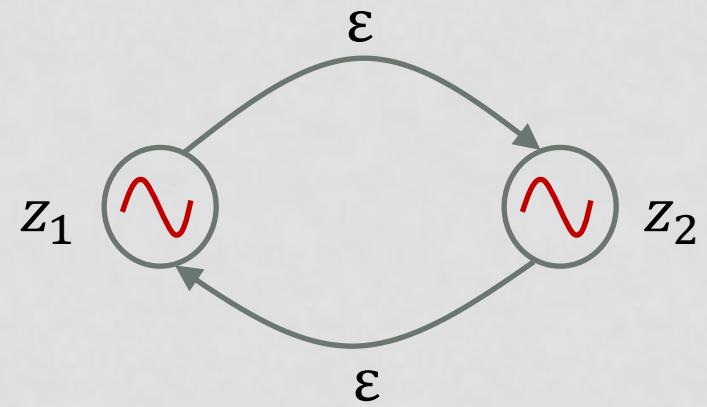
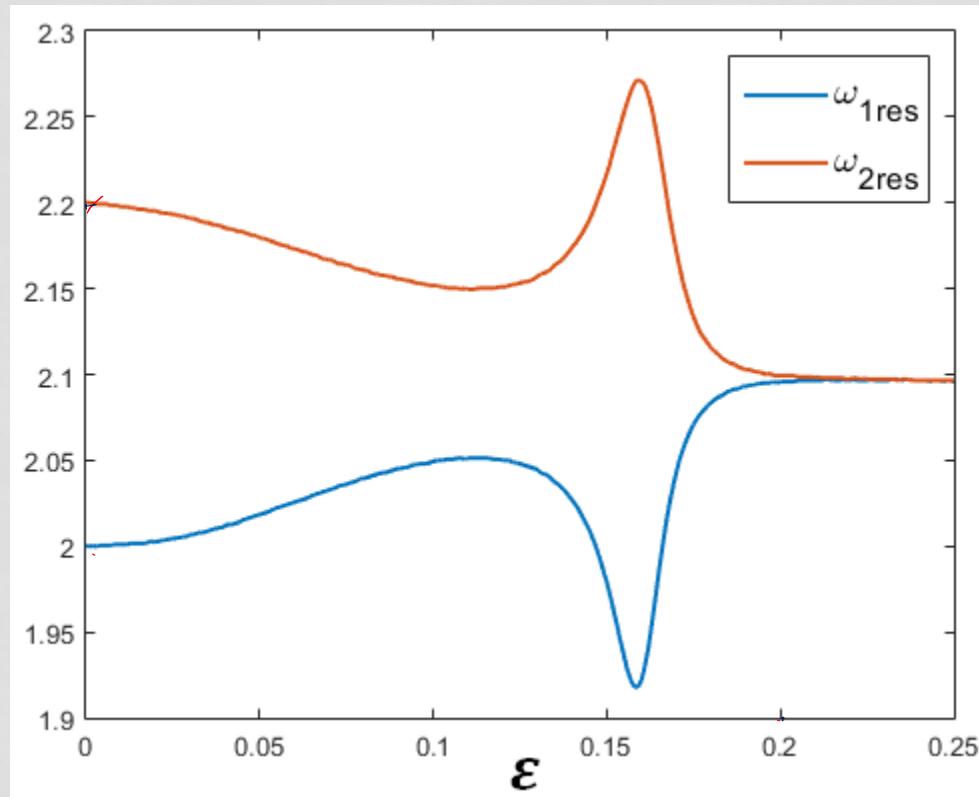


$$\dot{z}_1 = (\mu - |z_1|^2)z_1 + i\omega_1 z_1 + \varepsilon \times \text{real}(z_1)$$

$$\dot{z}_2 = (\mu - |z_2|^2)z_2 + i\omega_2 z_2 + \varepsilon \times \text{real}(z_2)$$

Simulation parameters:

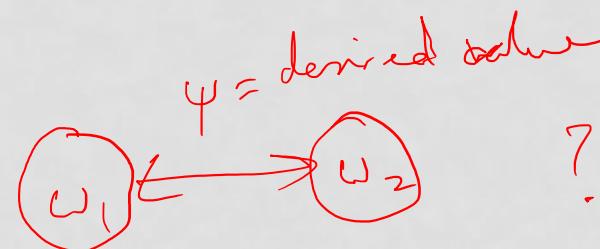
$$\mu = 1, \omega_1 = 2, \omega_2 = 2.2$$



COUPLING TWO HOPF OSCILLATORS WITH ARBITRARILY DIFFERENT FREQUENCIES

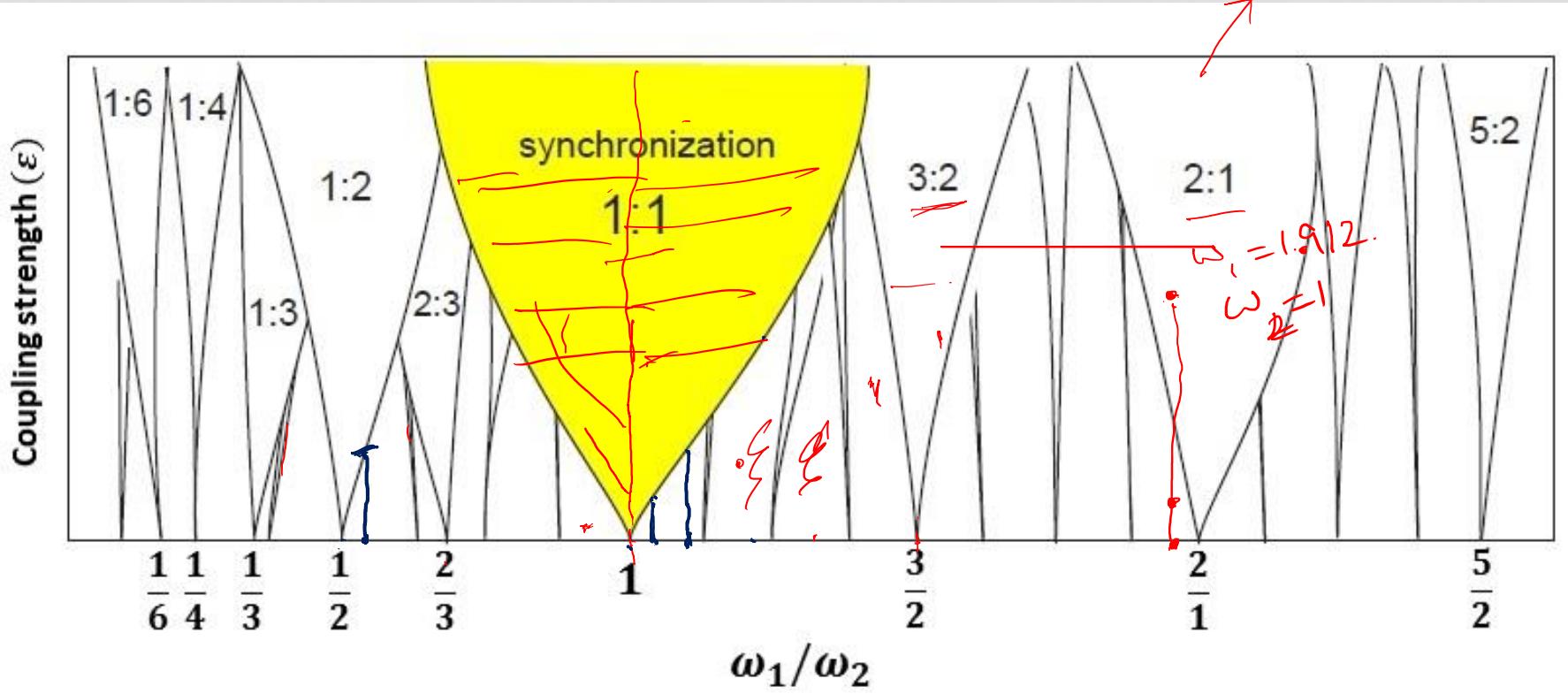
- When two oscillators with different intrinsic frequencies are coupled through real coupling weights a new behavior emerges while analyzing their synchronization behavior, called Arnold Tongues.

$$\begin{aligned}\dot{z}_1 &= (\mu - |z_1|^2)z_1 + i\omega_1 z_1 + \varepsilon \times \text{real}(z_1) \\ \dot{z}_2 &= (\mu - |z_2|^2)z_2 + i\omega_2 z_2 + \varepsilon \times \text{real}(z_2)\end{aligned}$$



COUPLING TWO HOPF OSCILLATORS: ARBITRARILY DIFFERENT FREQUENCIES, REAL COUPLING GENERAL SCENARIO

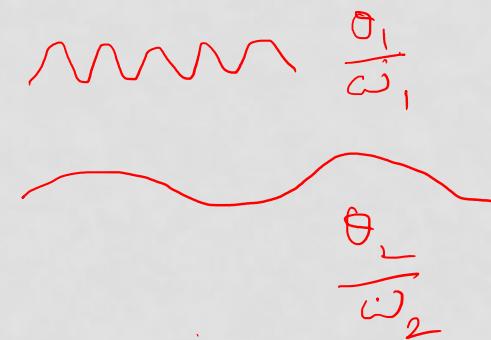
ARNOLD TONGUES



NORMALIZED PHASE DIFFERENCE

- When the frequencies are different, the usual phase difference cannot be constant
- But normalized phase difference can be constant
- normalized phase difference:

$$\psi = \frac{\theta_1}{\omega_1} - \frac{\theta_2}{\omega_2}$$



- This is what is constant when two oscillators are entrained in a simple integral ratio

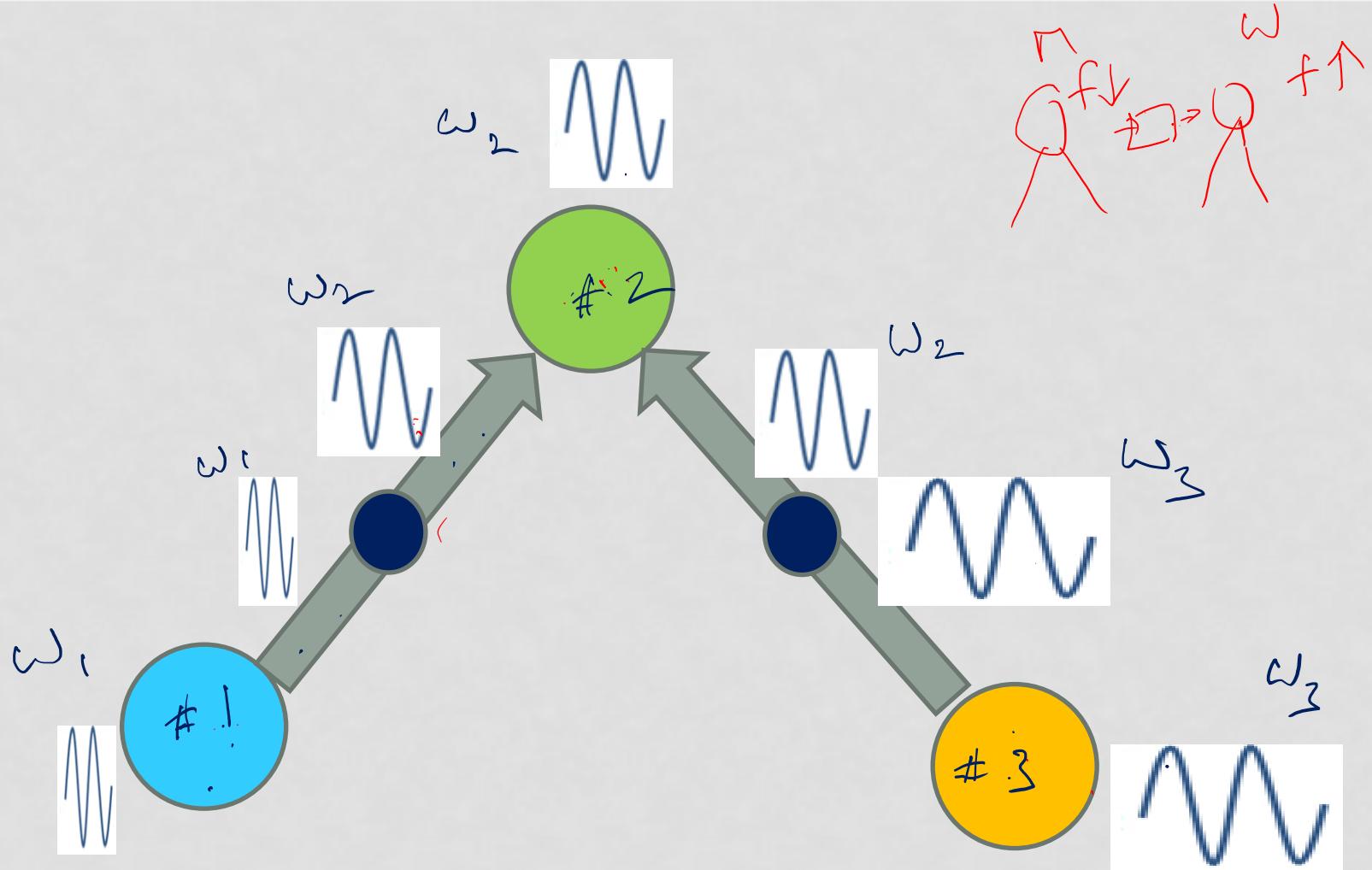
- How to achieve constant normalized phase difference for arbitrary frequencies?

Answer:

It requires a trick called

Power coupling

TRANSFORMING TRANSMITTER FREQUENCIES TO RECEIVER FREQUENCY





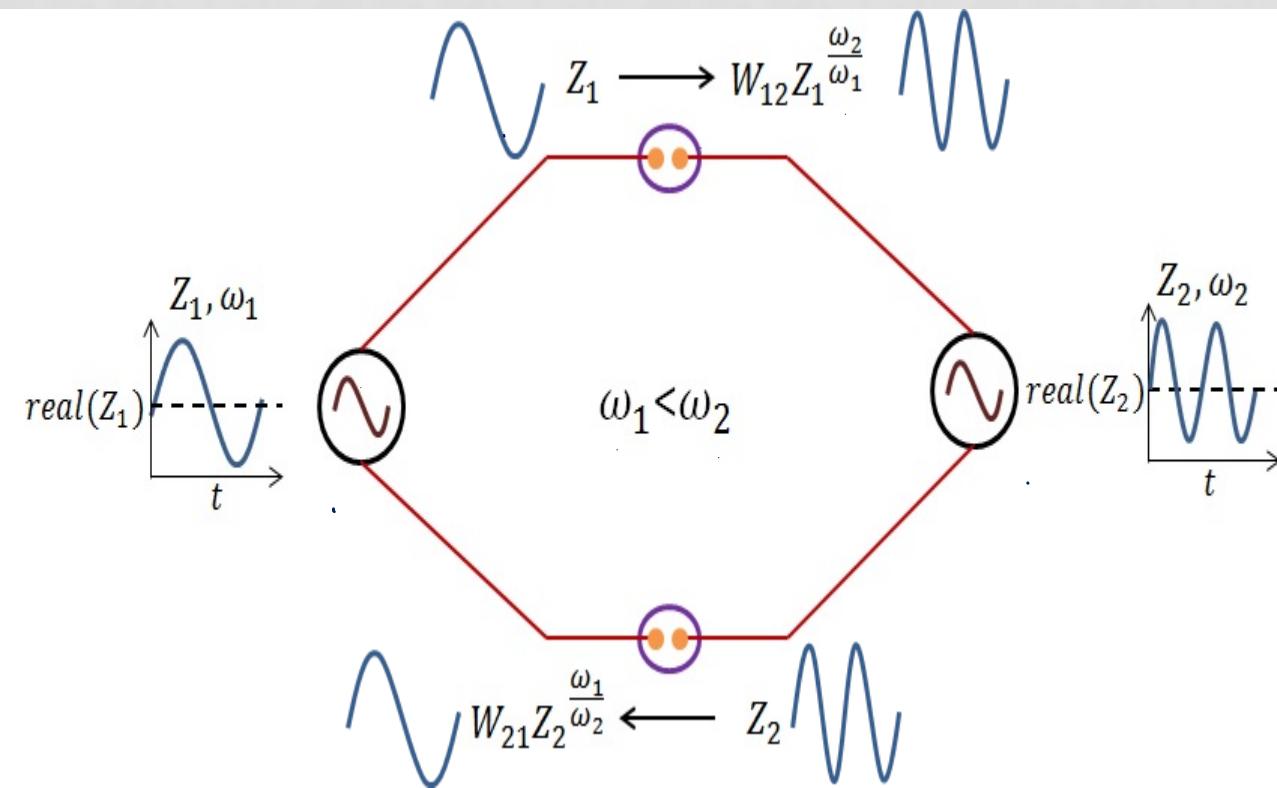
$$z_1 = \underline{A_1 e^{i\omega_1 t}}$$

How to transform ω_1 to ω_2 ?

$$(z_1)^{\circlearrowleft \omega_2 / \omega_1} = (A_1 e^{i\omega_1 t})^{\omega_2 / \omega_1} = (A_1 e^{i\omega_2 t})$$

POWER COUPLING

- Frequency of oscillator 1 is transformed into frequency of oscillator 2, as the signal passes over the connection



THEORETICAL ANALYSIS

- Two oscillators coupled through “power coupling”:

$$\begin{aligned}\dot{z}_1 &= (\mu - |z_1|^2)z_1 + i\omega_1 z_1 + Ae^{t\frac{\varphi}{\omega_2}} z_2 \frac{\omega_1}{\omega_2} \\ \dot{z}_2 &= (\mu - |z_2|^2)z_2 + i\omega_2 z_2 + Ae^{-i\frac{\varphi}{\omega_1}} z_1 \frac{\omega_2}{\omega_1}\end{aligned}$$

- The angular part of the polar form:

$$\begin{aligned}\dot{\theta}_1 &= \omega_1 + A \frac{r_2 \frac{\omega_1}{\omega_2}}{r_1} \sin \omega_1 \left(\frac{\theta_2}{\omega_2} - \frac{\theta_1}{\omega_1} + \frac{\varphi}{\omega_1 \omega_2} \right) \\ \dot{\theta}_2 &= \omega_2 + A \frac{r_1 \frac{\omega_2}{\omega_1}}{r_2} \sin \omega_2 \left(\frac{\theta_1}{\omega_1} - \frac{\theta_2}{\omega_2} - \frac{\varphi}{\omega_1 \omega_2} \right)\end{aligned}$$

- We denote the normalized phase difference:

$$\psi = \frac{\theta_1}{\omega_1} - \frac{\theta_2}{\omega_2}$$

- The derivative of normalized phase difference w.r.t time:

$$\dot{\psi} = \frac{\dot{\theta}_1}{\omega_1} - \frac{\dot{\theta}_2}{\omega_2}$$

$$\dot{\psi} = \frac{Ar_2 \frac{\omega_1}{\omega_2}}{\omega_1 r_1} \sin \omega_1 \left(\frac{\varphi}{\omega_1 \omega_2} - \psi \right) + \frac{Ar_1 \frac{\omega_2}{\omega_1}}{\omega_2 r_2} \sin \omega_2 \left(\frac{\varphi}{\omega_1 \omega_2} - \psi \right)$$

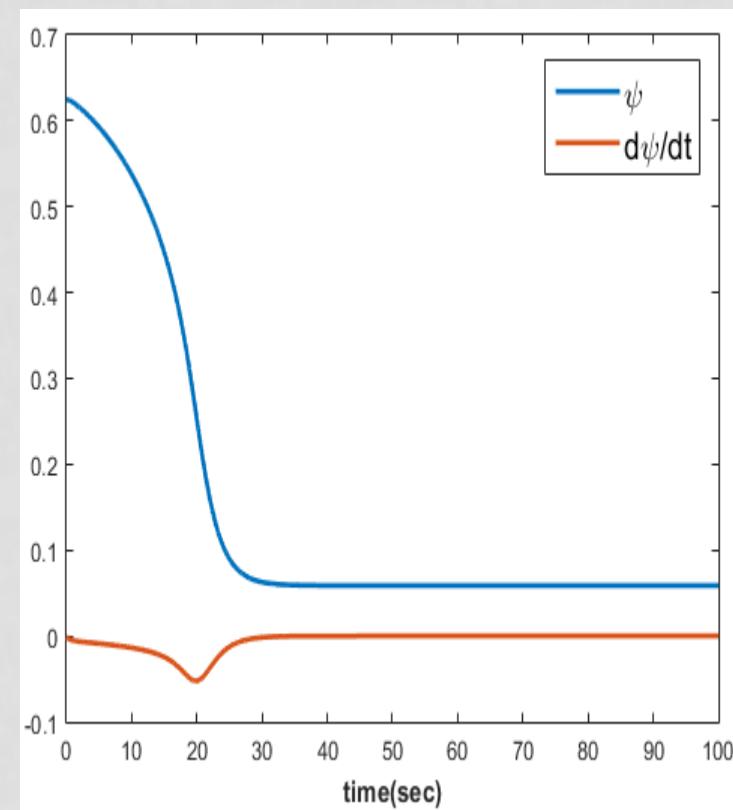
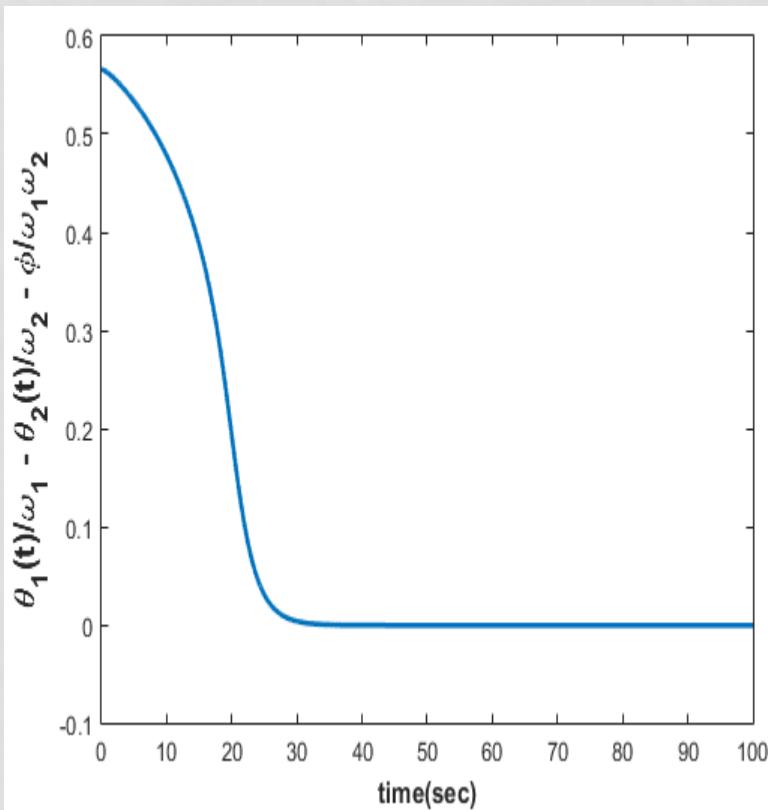
- At steady state; $\psi = \text{constant}$, $\dot{\psi} = 0$ and $r_1 = r_2 = 1$:

$$\dot{\psi} = \frac{A}{\omega_1} \sin \omega_1 \left(\frac{\varphi}{\omega_1 \omega_2} - \psi \right) + \frac{A}{\omega_2} \sin \omega_2 \left(\frac{\varphi}{\omega_1 \omega_2} - \psi \right) = 0$$

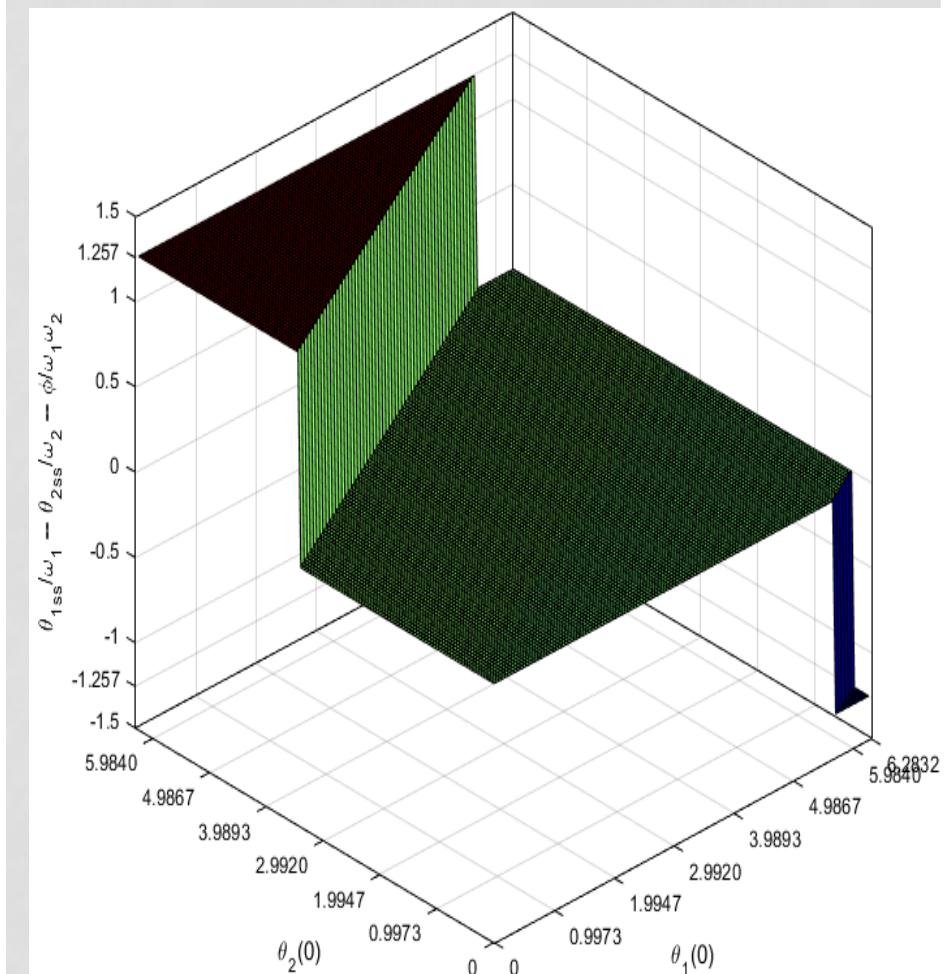
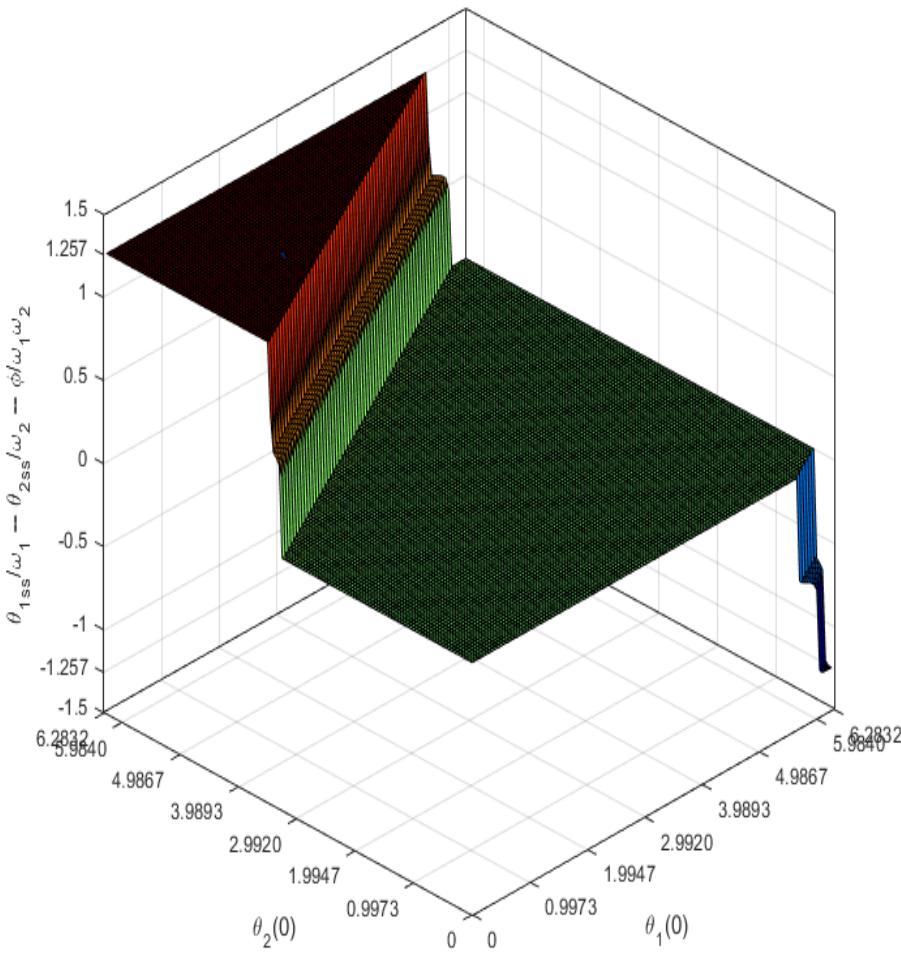
- The steady state normalized phase difference satisfies the equation:

$$\psi = \frac{\theta_1}{\omega_1} - \frac{\theta_2}{\omega_2} = \frac{\varphi}{\omega_1 \omega_2}$$

- The $\sigma = \frac{\theta_1(t)}{\omega_1} - \frac{\theta_2(t)}{\omega_2} - \frac{\varphi}{\omega_1 \omega_2}$, $\psi(t)$ and $\dot{\psi}(t)$ variation w.r.t time for two 'power coupled' Hopf oscillators with the following parameters.
- $\omega_1 = 5$, $\omega_2 = 10$, $A_{12} = A_{21} = 0.2$, $\varphi = 2.9644$. $\varphi/50 = \underline{\psi}$



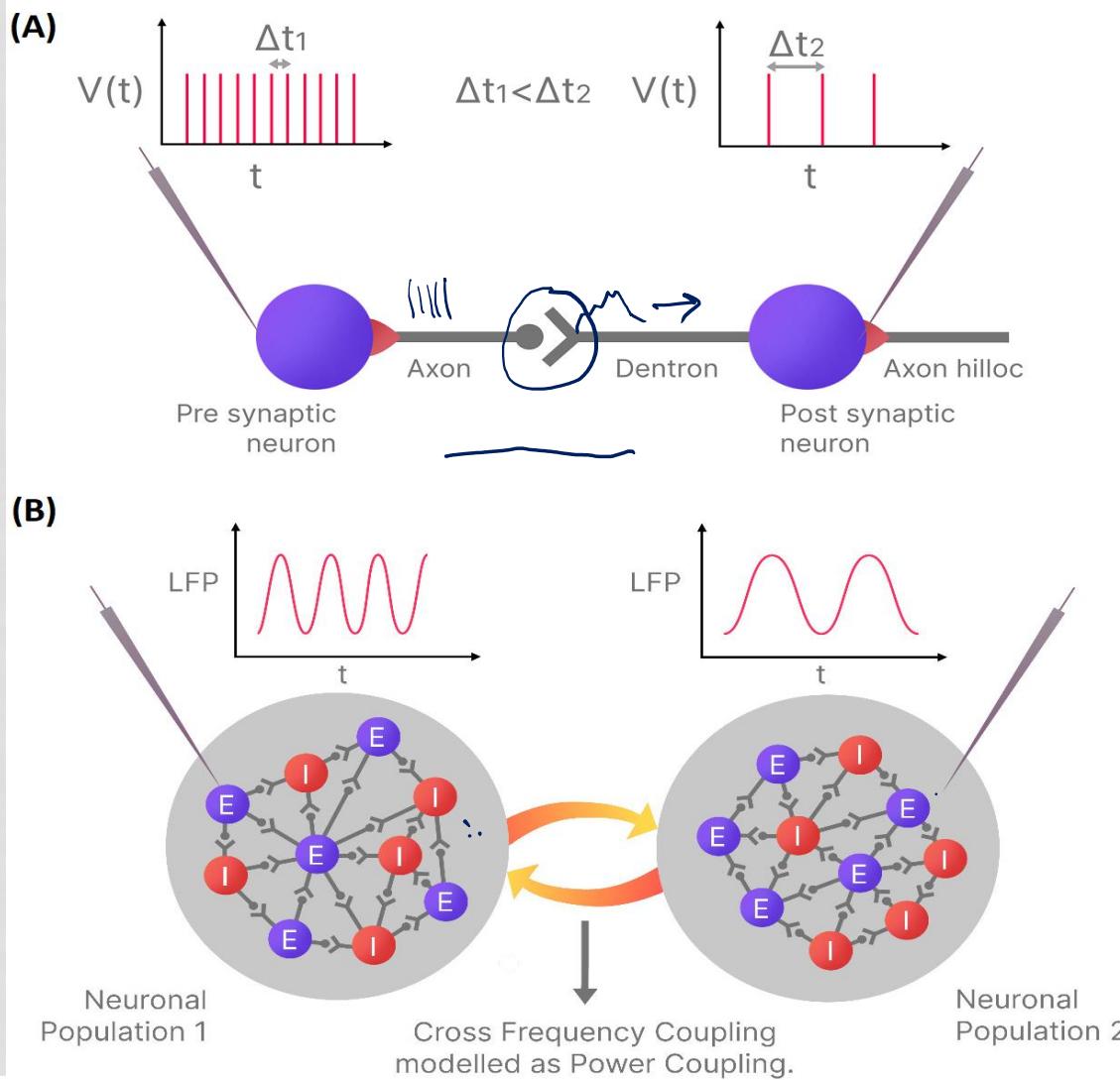
- Depending on initial values of θ_1 and θ_2 , σ_{ss} can attain any of the solutions of $\dot{\psi}_{ss} = 0$ equation.
- The solutions other than $\sigma_{ss} = 0$ are: $\sigma_{ss} = \frac{2\pi n_1}{\omega_1} = \frac{2\pi n_2}{\omega_2}$, where $\frac{n_1}{n_2} = \frac{\omega_1}{\omega_2}$
- In the simulated scenario the stable solutions are: $\sigma_{ss} = 0, \frac{2n_1\pi}{\omega_1} = \frac{2n_2\pi}{\omega_2}$



$\frac{\omega_1}{\omega_2}$ = rational #

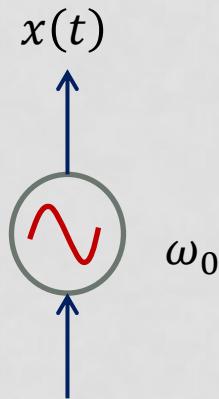
$$\sim \sin(\varphi_1 \left(\frac{2\pi n_1}{\omega_1} \right)) + \sim \sin(\varphi_2 \left(\frac{2\pi n_2}{\omega_2} \right)) = 0$$

BIOLOGICAL INTERPRETATION



FREQUENCY TRACKING

The intrinsic frequency of the oscillator can track the input frequency



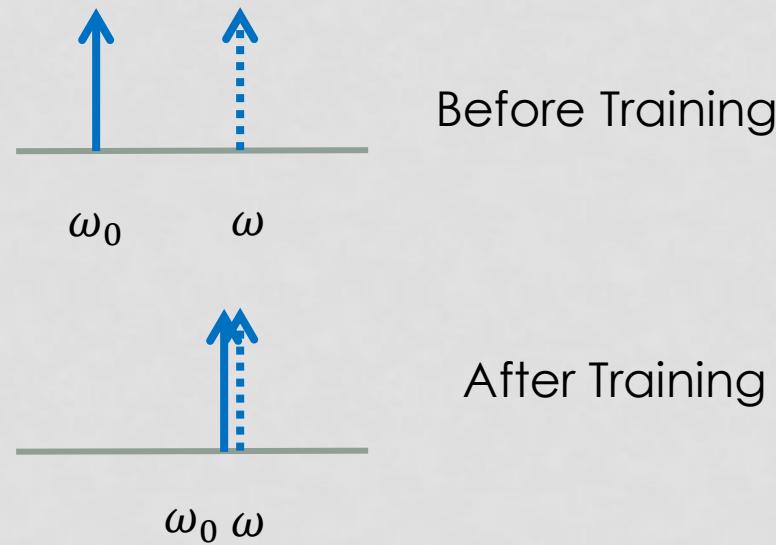
$$F(t) = Ae^{i(\omega t)}$$

$$\dot{x} = (1 - r^2)x - \omega y + \varepsilon F(t)$$

$$\dot{y} = (1 - r^2)y + \omega x$$

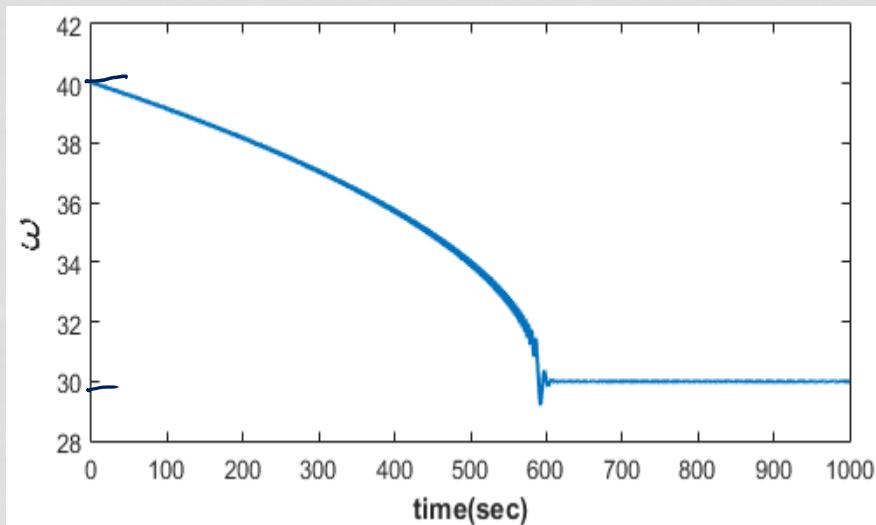
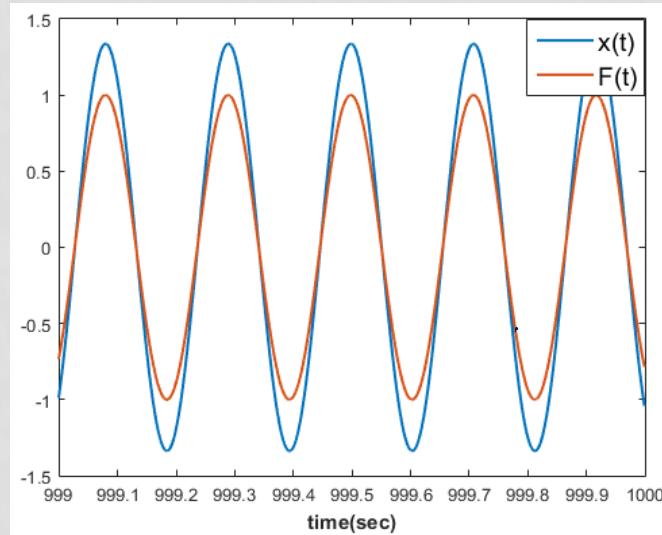
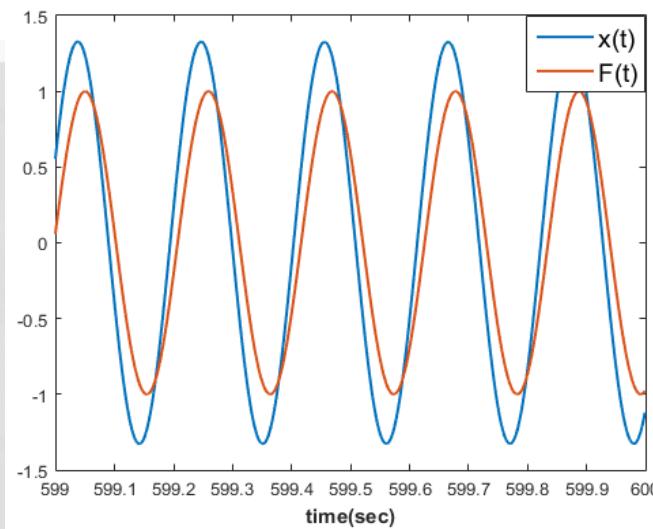
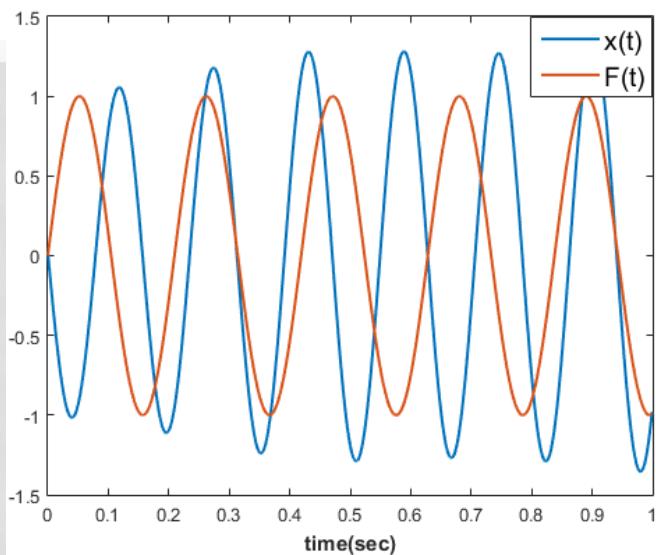
$$\dot{\omega} = -\varepsilon F(t) \frac{y}{r}$$

(Righetti et al 2005)

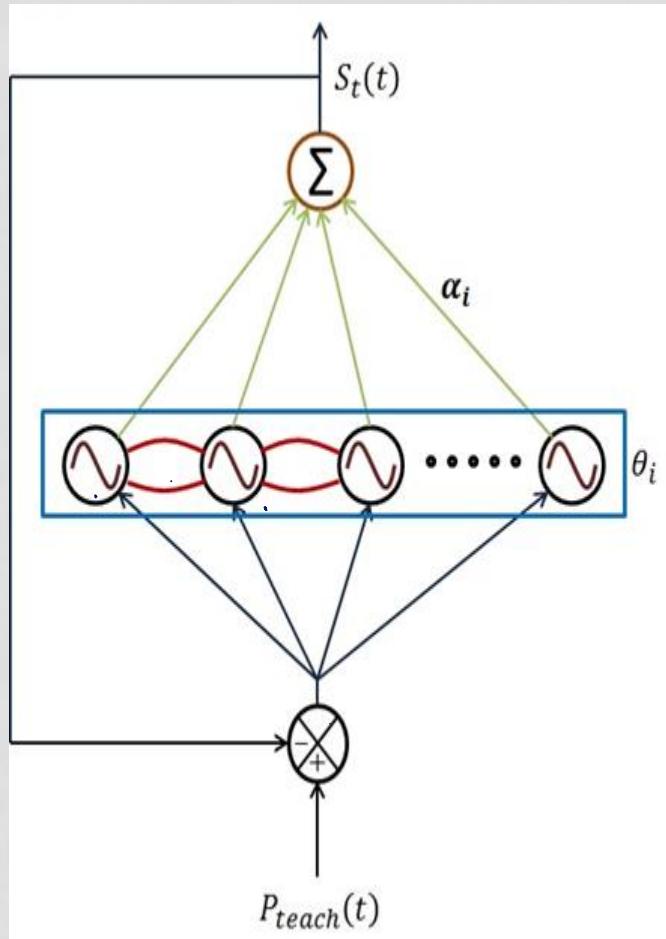


$$\begin{aligned}\dot{x} &= (1 - r^2)x - \omega y + \varepsilon F(t) \\ \dot{y} &= (1 - r^2)y + \omega x \\ \dot{\omega} &= -\varepsilon F(t) \frac{y}{r}\end{aligned}$$

$$\omega(0) = 40, \quad \varepsilon = 0.9, \quad F(t) = \sin 30t$$



COUPLED N-OSCILLATOR NETWORK



Reconstructing a time series
By taking a weighted sum of outputs
of N-oscillator network



RECONSTRUCTING A TIME SERIES

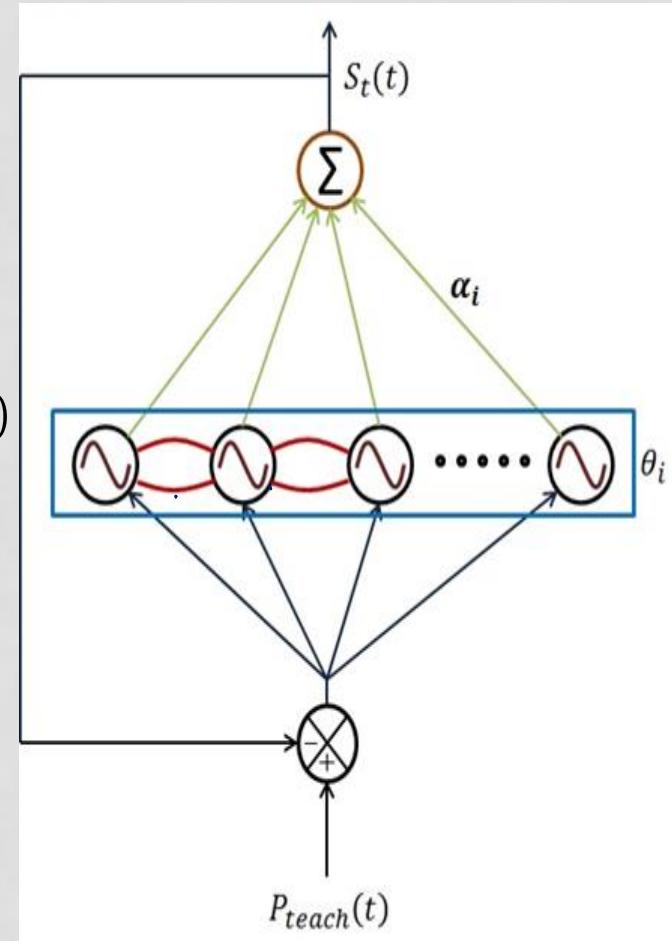
- Network of n no of Kuramoto oscillators can learn Some sort of Fourier decomposition of any $P_{teach}(t)$ signal.

$$\dot{Z}_i = (\mu - |Z_i|^2)Z_i + i\omega Z_i + \sum_j A_{ij} e^{i\frac{\phi_{ij}}{\omega_j} Z_j \frac{\omega_i}{\omega_j}} + \varepsilon(P_{teach}(t) - S(t))$$

$$\begin{aligned} \dot{\theta}_j &= \omega_j + \sum_k A_{jk} \sin \omega_j \left(\frac{\theta_k}{\omega_k} - \frac{\theta_j}{\omega_j} + \frac{\phi_{jk}}{\omega_j \omega_k} \right) \\ &+ \varepsilon(P_{teach}(t) - S(t)) \sin \theta_j \end{aligned}$$

$$\dot{\omega}_j = -\eta_\omega \underbrace{(P_{teach}(t) - S(t))}_{\text{sin } \theta_j} \sin \theta_j \quad \checkmark$$

$$S(t) = \sum_j \alpha_j \cos \theta_j \quad (\text{reconstructed signal})$$



TEST CASE#1

THE INTRINSIC FREQUENCIES OF THE OSCILLATORS LEARNED THE FREQUENCY COMPONENTS OF THE INPUT SIGNAL

$$P_{teach}(t) = A_1 \cos(\omega_1 t + \phi_1) + A_2 \cos(\omega_2 t + \phi_2) + A_3 \cos(\omega_3 t + \phi_3)$$

Where: $A_1 = 1.5909, A_2 = 1.377, A_3 = 1.3938$

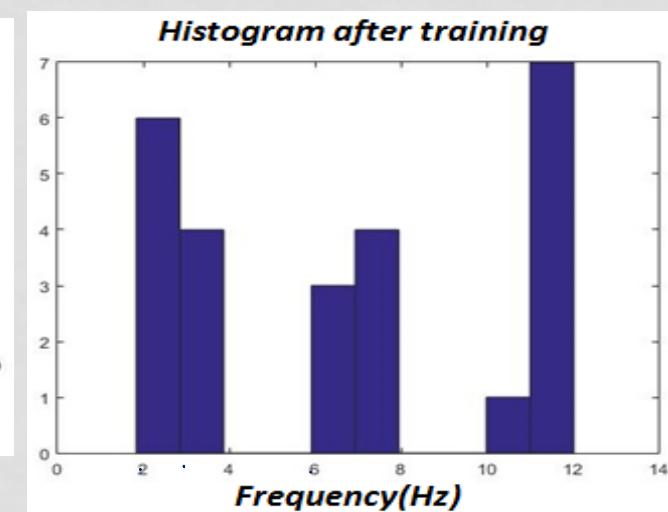
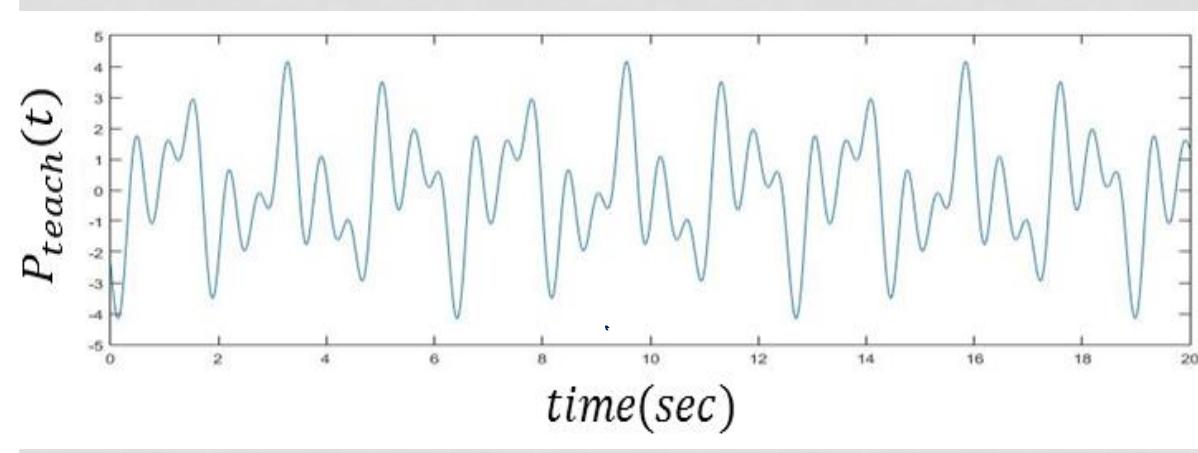
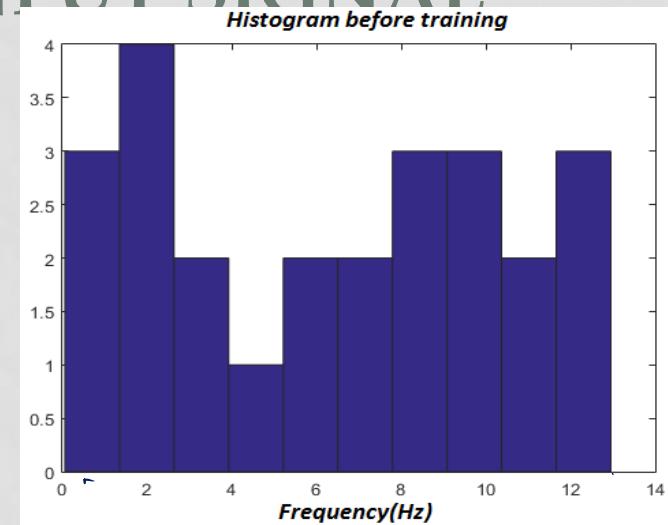
$$\omega_1 = 3, \omega_2 = 7, \omega_3 = 11$$

$$\phi_1 = 2.6911, \phi_2 = 2.6579, \phi_3 = 1.3755$$

$$n = 25$$

$$\eta_\omega = 0.001$$

$$\alpha_j = 0.2$$



GENERALIZED NETWORK#2
 FROM LEARNING FREQUENCY SPECTRUM REPRESENTATION TO
 MODELLING ARBITRARY M NO OF TIME SERIES SIGNALS HAVING SAME
 FREQUENCY COMPONENTS PRESENT IN THE SIGNAL USED TO TRAIN
 NETWORK#1

$$L = \frac{1}{2} \sum_{i=1}^m \sum_t \|Y_{d_i}(t) - Y_{p_i}(t)\|^2$$

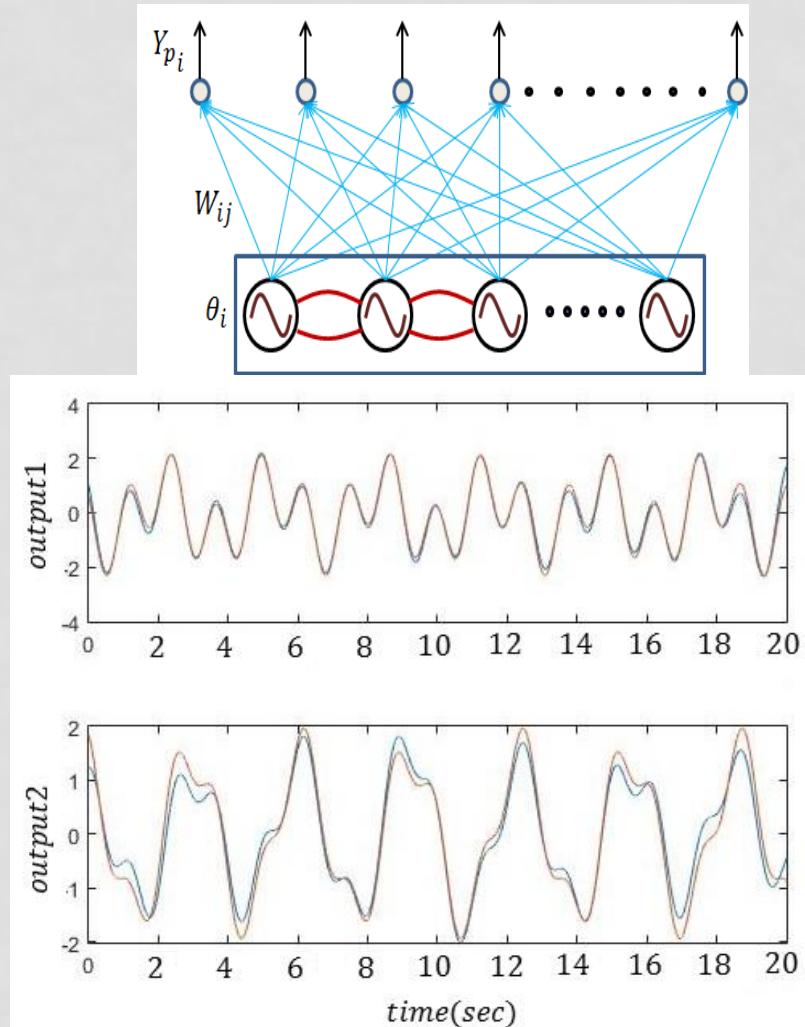
$$Y_{p_i}(t) = \operatorname{real} \left(\sum_{j=1}^n W_{ij} e^{i\theta_j} \right)$$

$$W_{ij} = K_{ij} e^{i\phi_{ij}}$$

$$\Delta K_{ij} = (-1) \eta_{K_{ij}} \sum_t (Y_{d_i}(t) - Y_{p_i}(t)) \cos(\theta_j(t) + \phi_{ij})$$

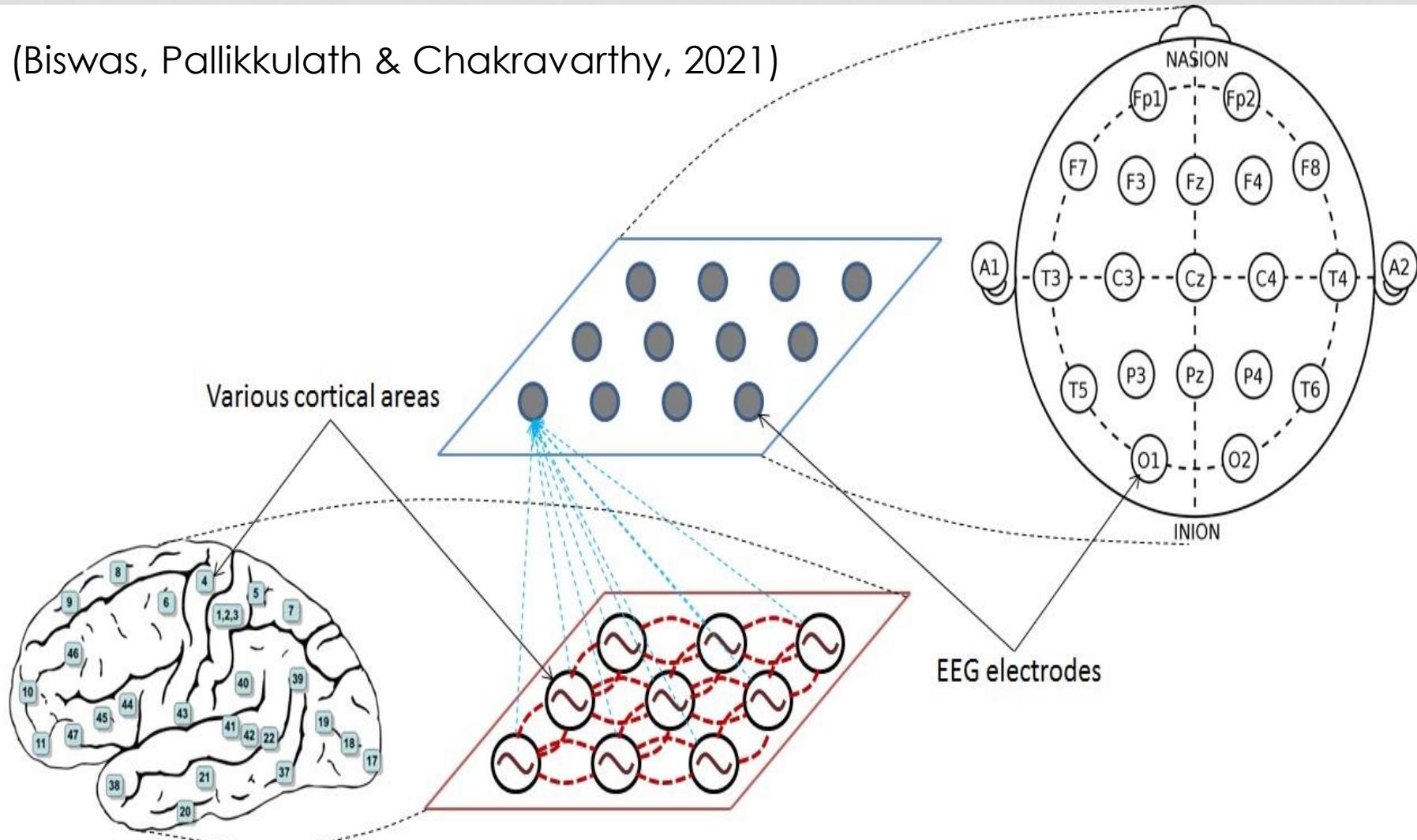
$$\Delta \phi_{ij}$$

$$= (-1) \eta_{\phi_{ij}} \sum_t (Y_{d_i}(t) - Y_{p_i}(t)) (-K_{ij} \sin(\theta_j(t) + \phi_{ij}))$$

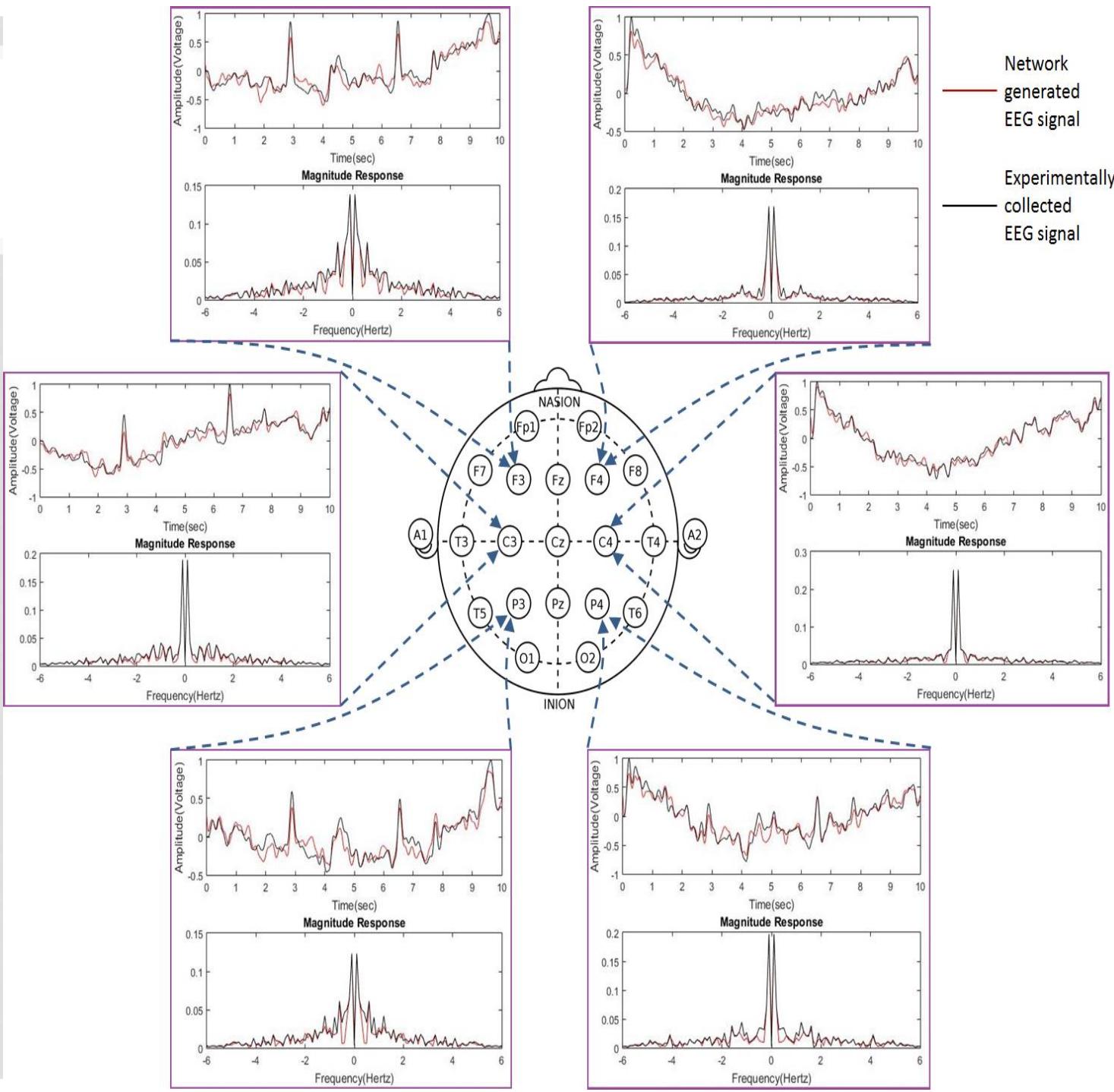


MODELLING ELECTROENCEPHALOGRAPHIC SIGNAL USING NETWORK#2

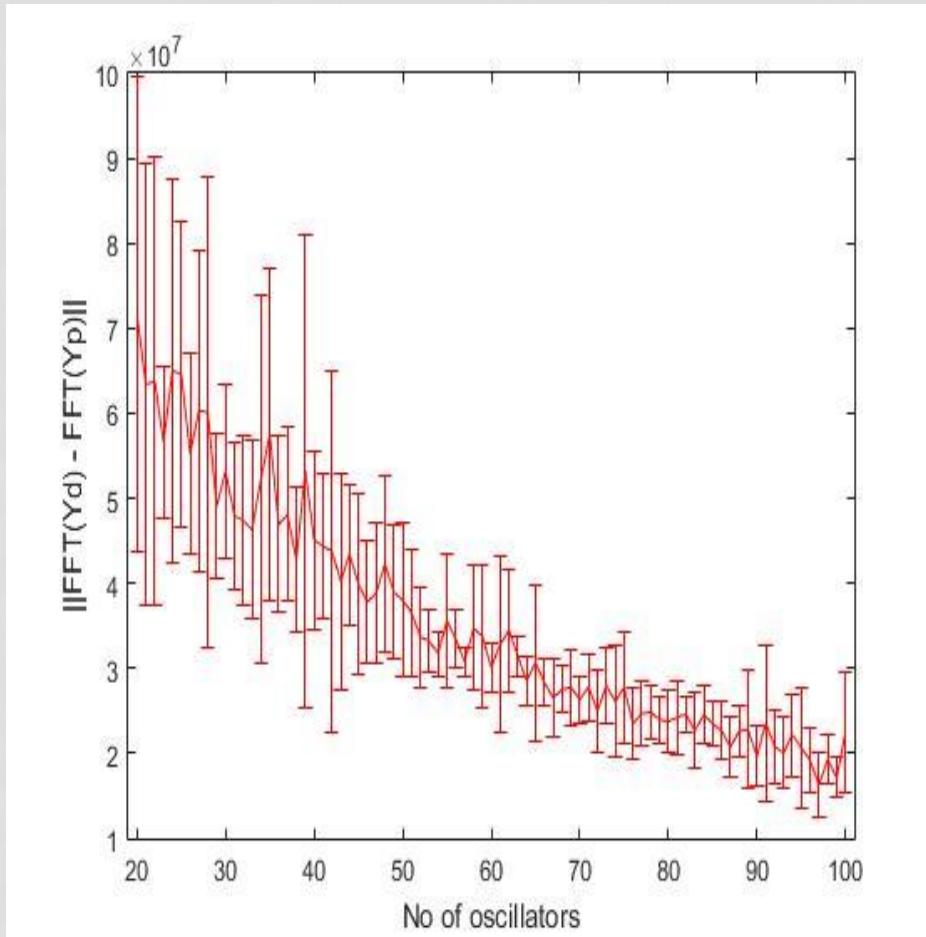
(Biswas, Pallikkulath & Chakravarthy, 2021)



PHASE #1
NETWORK
LEARNS THE
FREQUENCY
SPECTRUM OF A
EEG SIGNAL
COLLECTED
FROM ONE
ELECTRODE AND
IN THE



RECONSTRUCTION ACCURACY INCREASES WITH #OSCILLATORS IN THE NETWORK



FUTURE STEPS

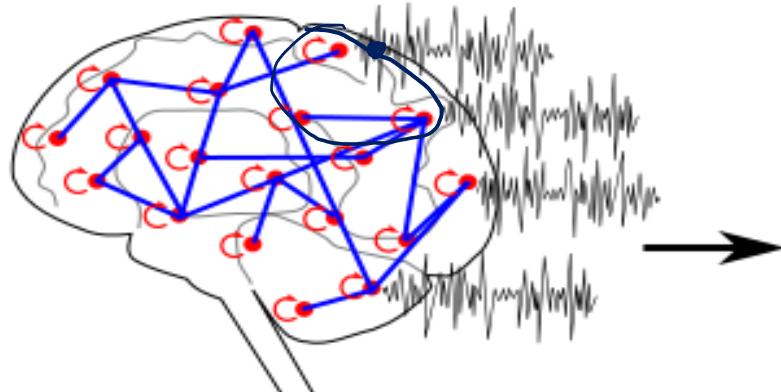
- Large scale Brain models using Oscillatory Deep Neural Networks
- Modeling the Auditory system using Oscillatory Neural Networks
- Generalized Deep Oscillatory Neural Networks

1. LARGE SCALE BRAIN MODELS

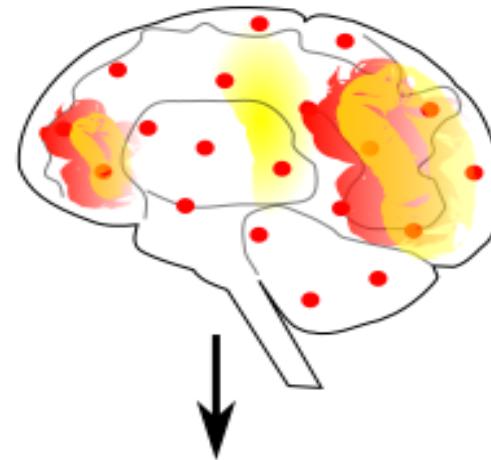
The Virtual Brain (TVB)

- Network of Wilson-Cowan Oscillators
- Connections computed by General optimization

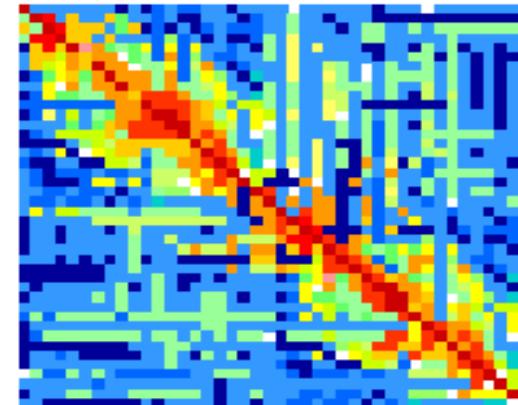
Whole brain simulation
and BOLD signal generation



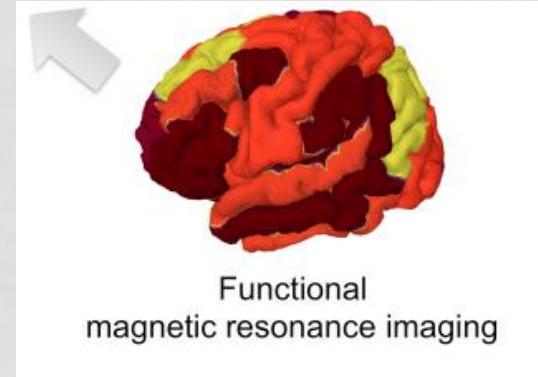
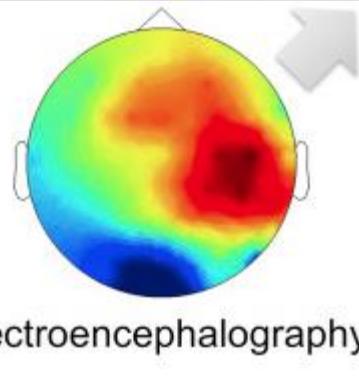
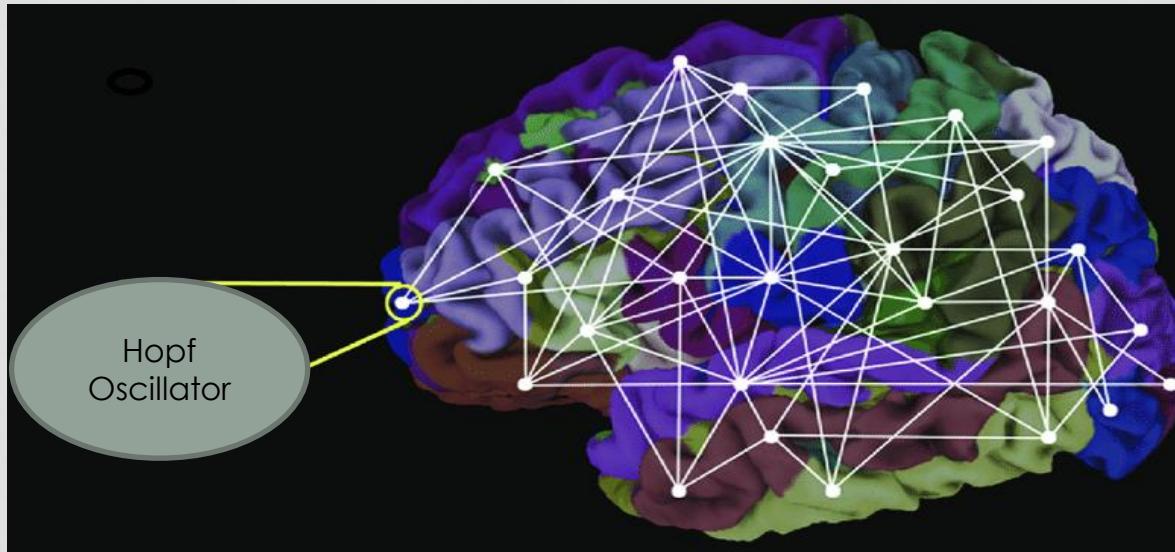
Experimental fMRI data



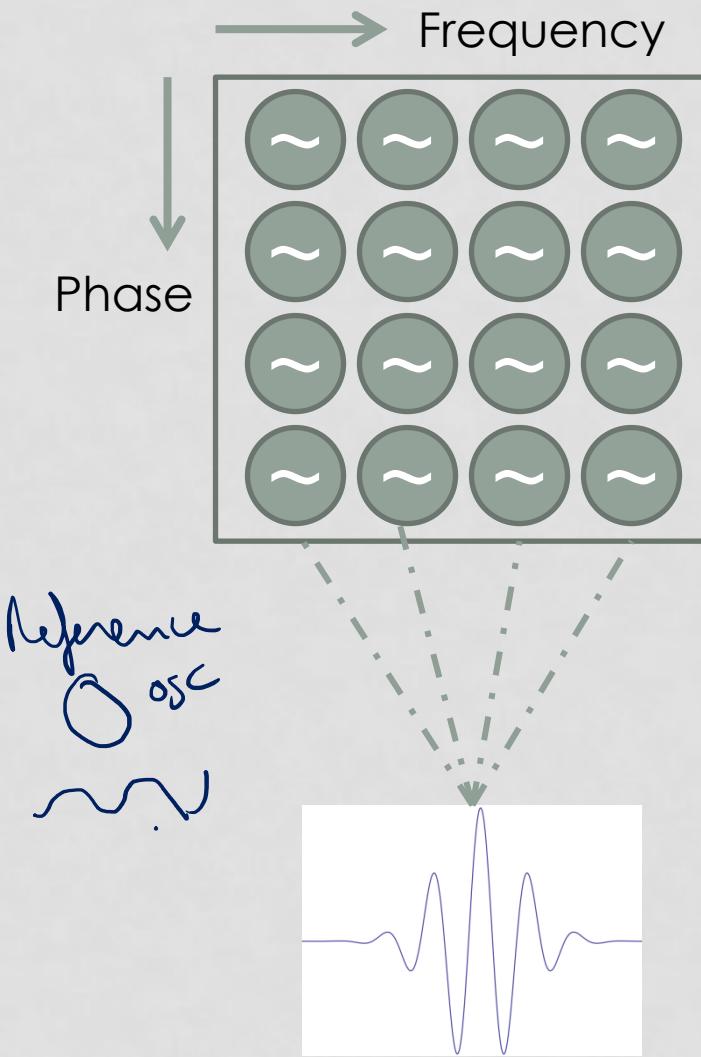
Comparison between simulated
and experimental BOLD signals



LARGE SCALE BRAIN MODELING USING OSCILLATORY NEURAL NETWORK



2. TONOTOPIC MAP MAP OF SOUNDS OR FREQUENCIES

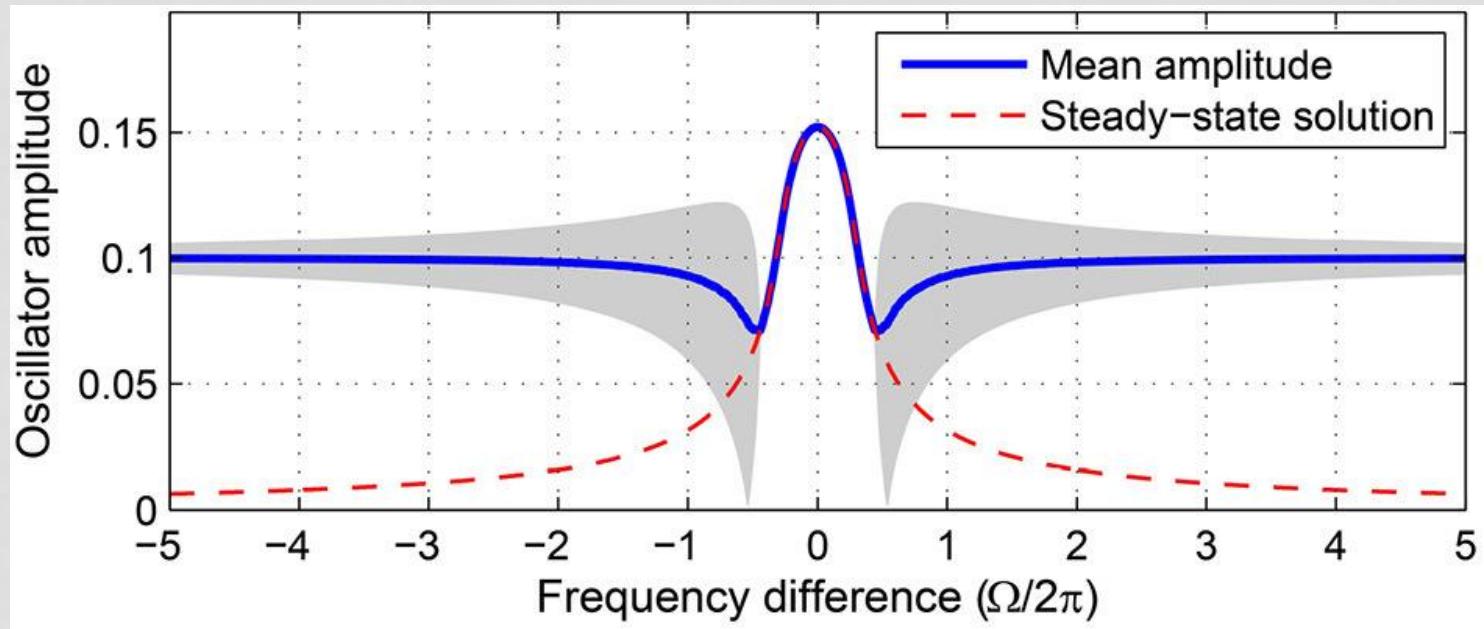


Fourier Series

$$g(t) = a_0 + \sum_{m=1}^{\infty} a_m \cos\left(\frac{2\pi mt}{T}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{2\pi nt}{T}\right)$$

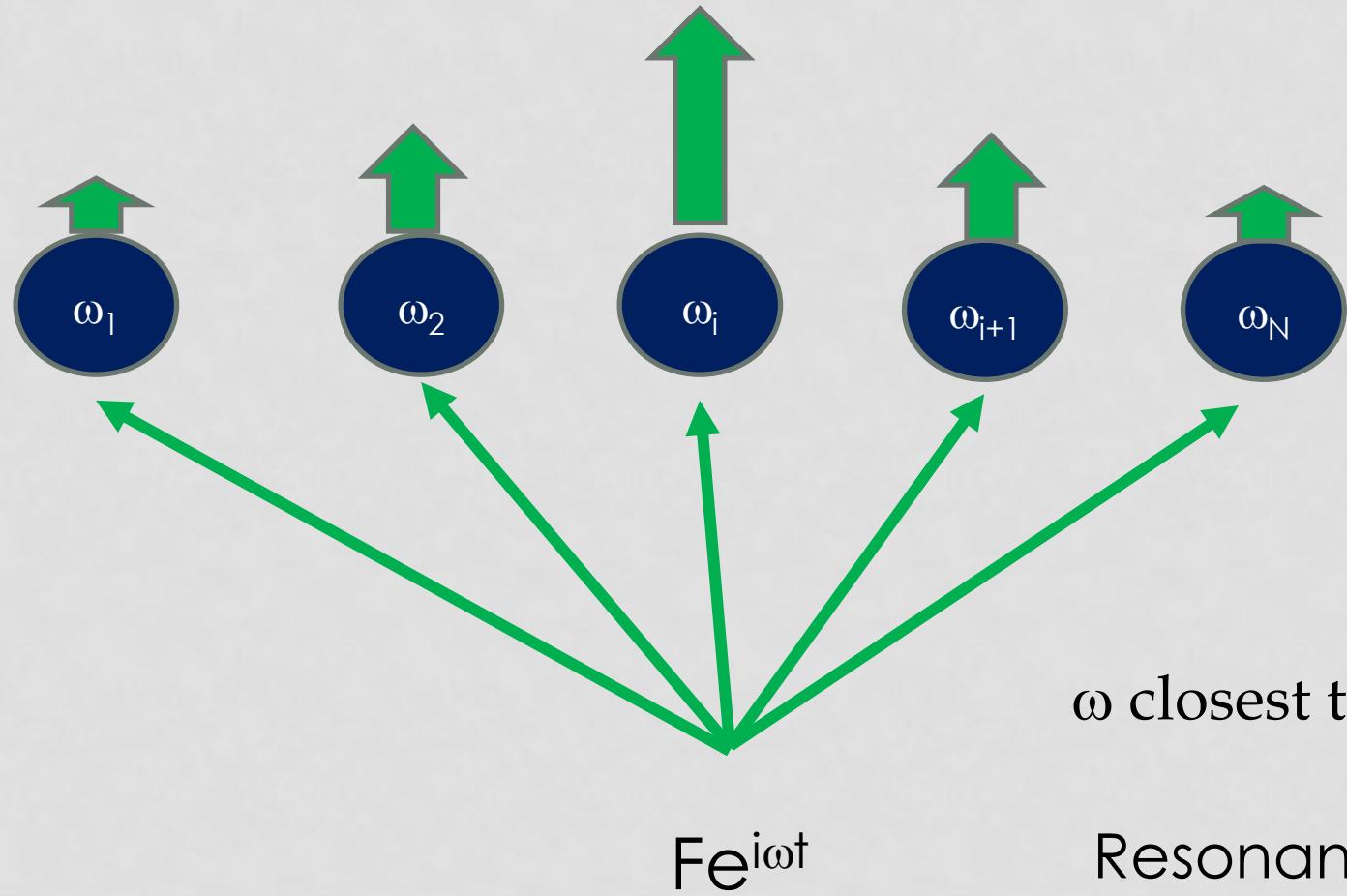
$$= \sum_{m=0}^{\infty} a_m \cos\left(\frac{2\pi mt}{T}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{2\pi nt}{T}\right)$$

HOPF OSCILLATORS EXHIBITS RESONANCE

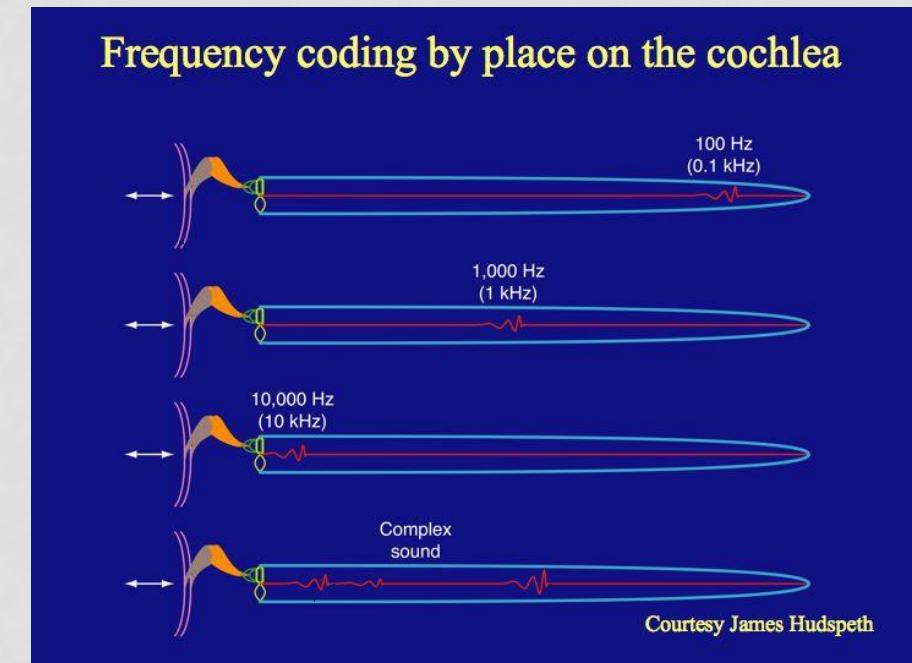
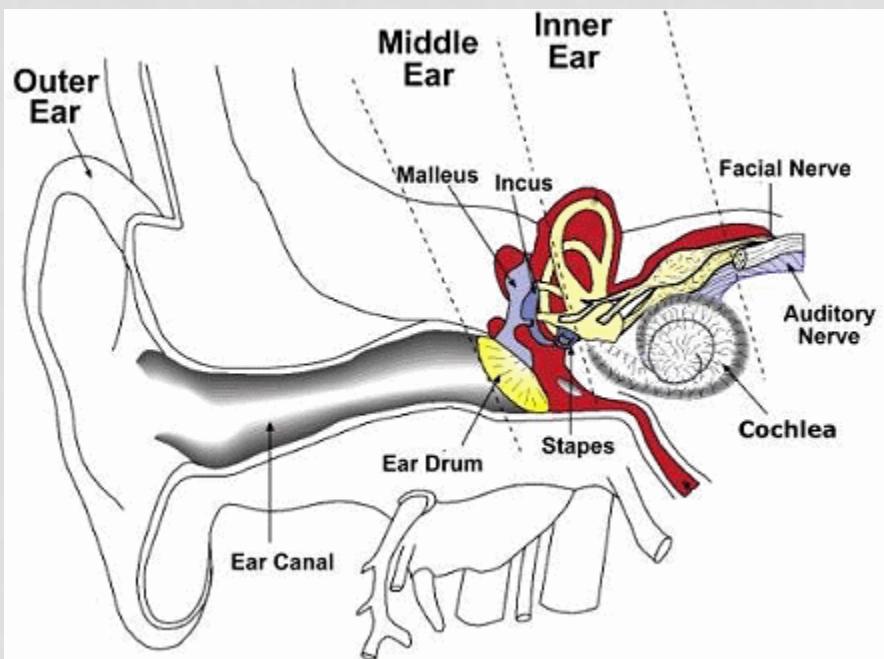


$$\Omega = \omega - \omega_0$$

(Kim & Large,)

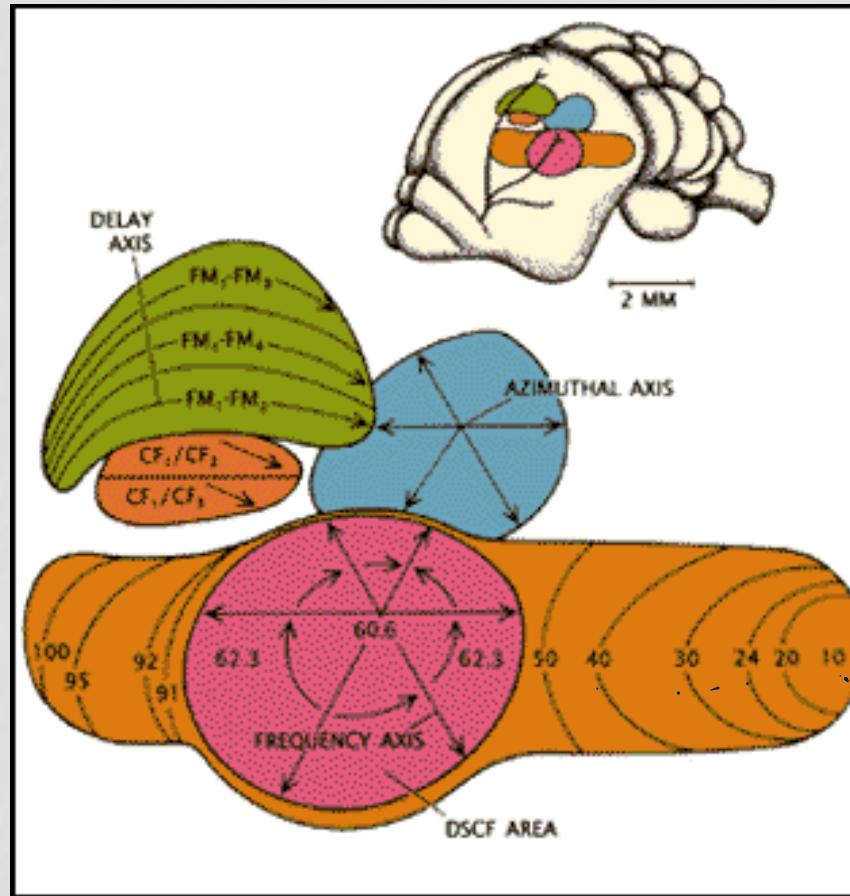
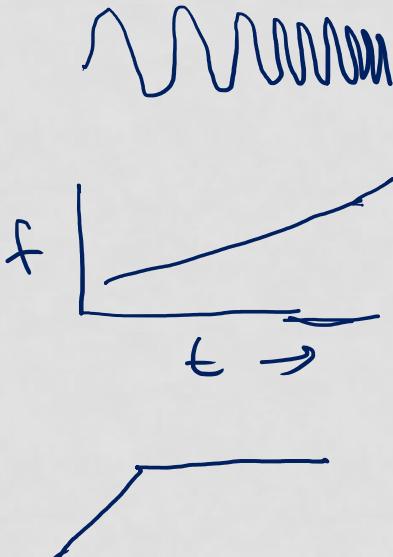


FOURIER DECOMPOSITION IN COCHLEA



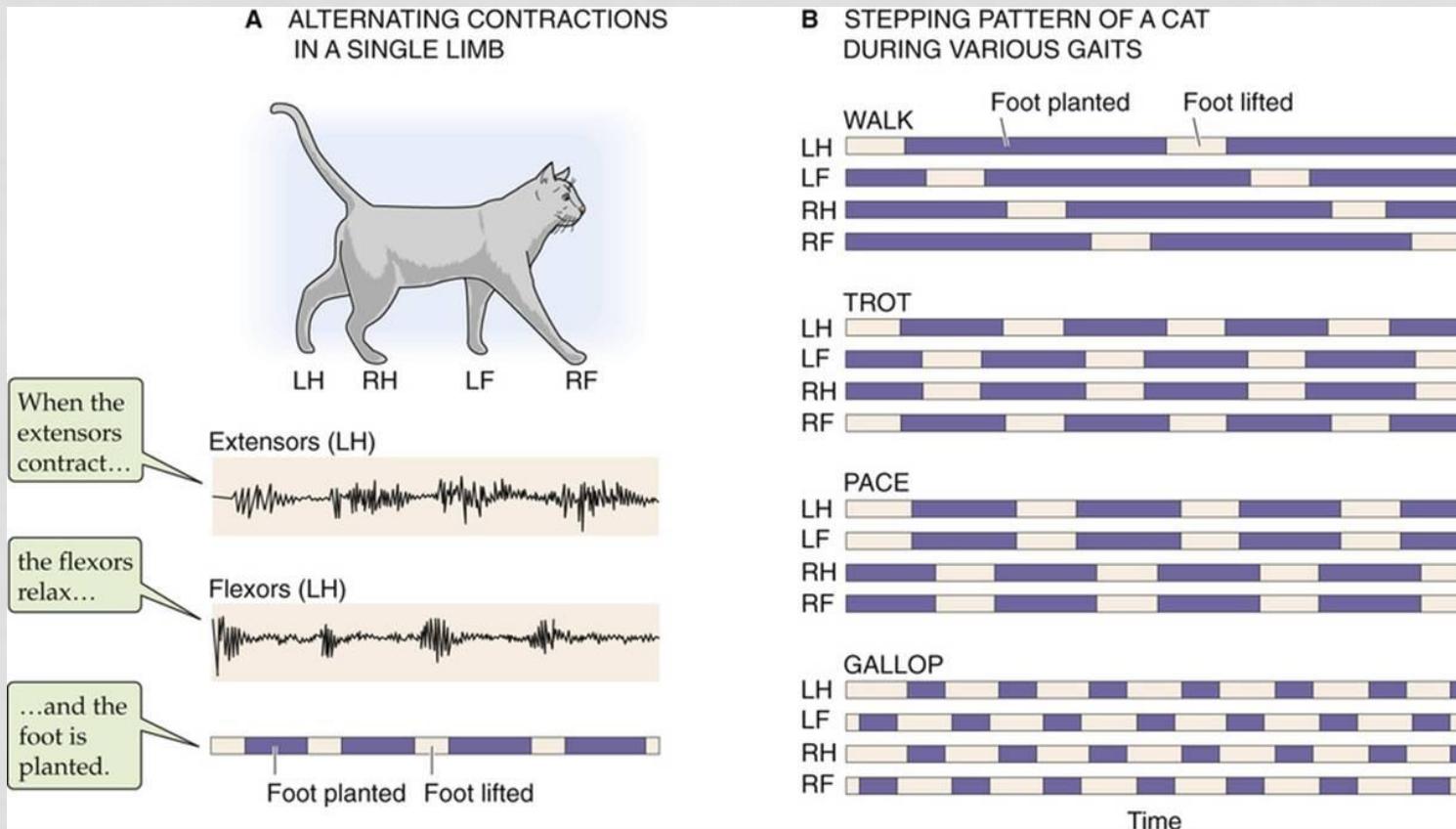
AUDITORY CORTEX OF BATS

chirp signals.

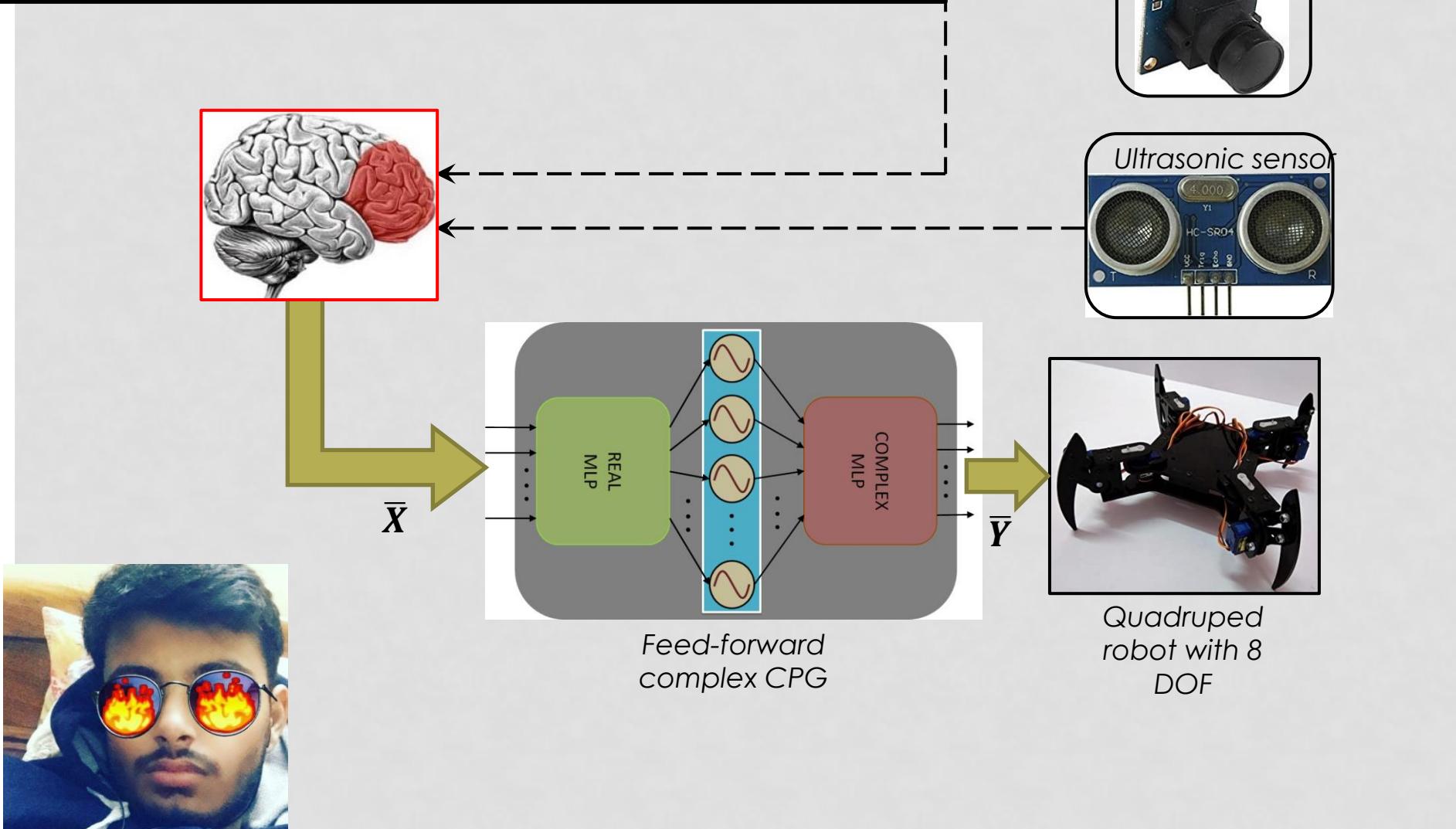


Nobuo
Suga

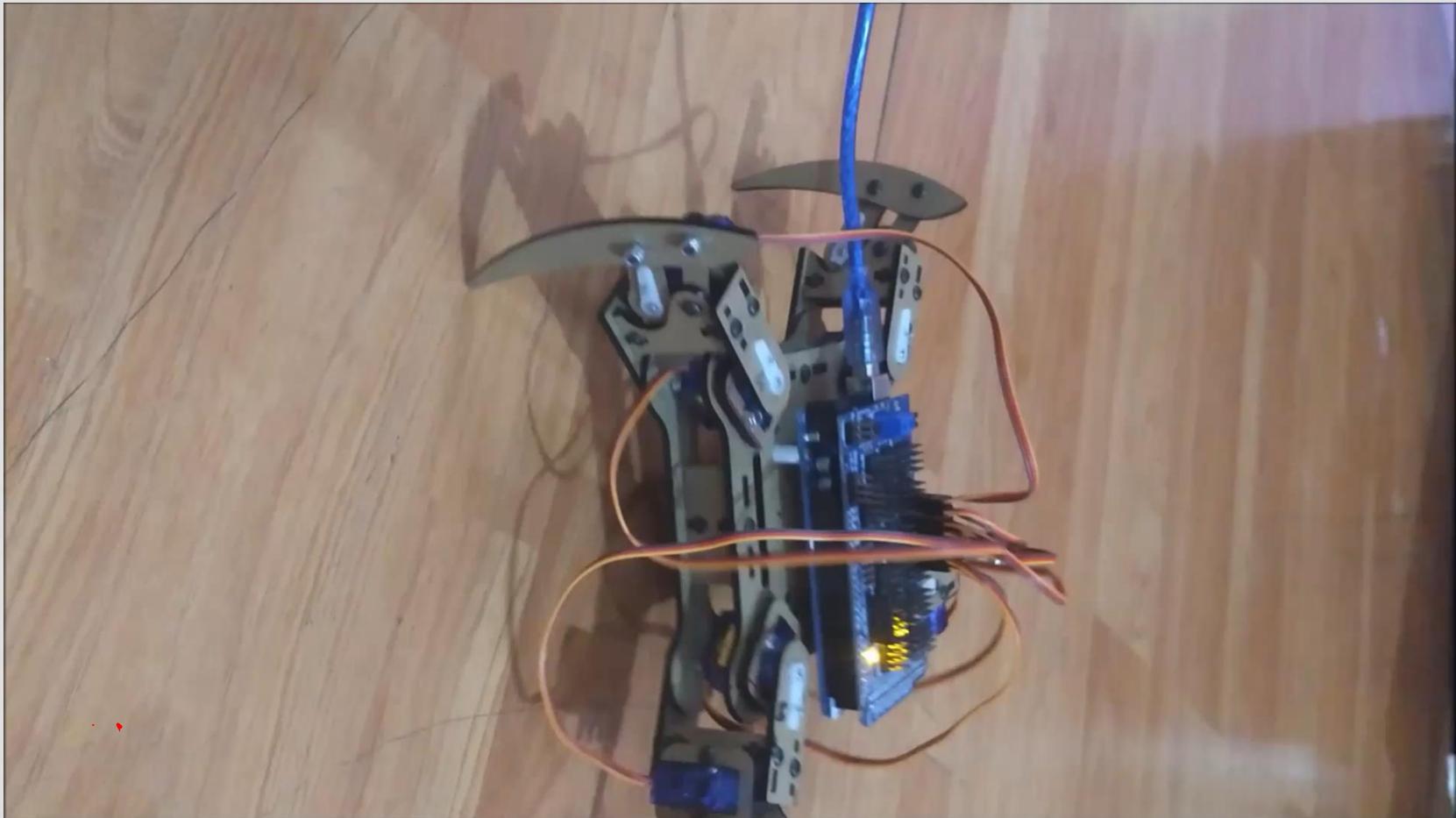
3. MODELING LOCOMOTOR RHYTHMS



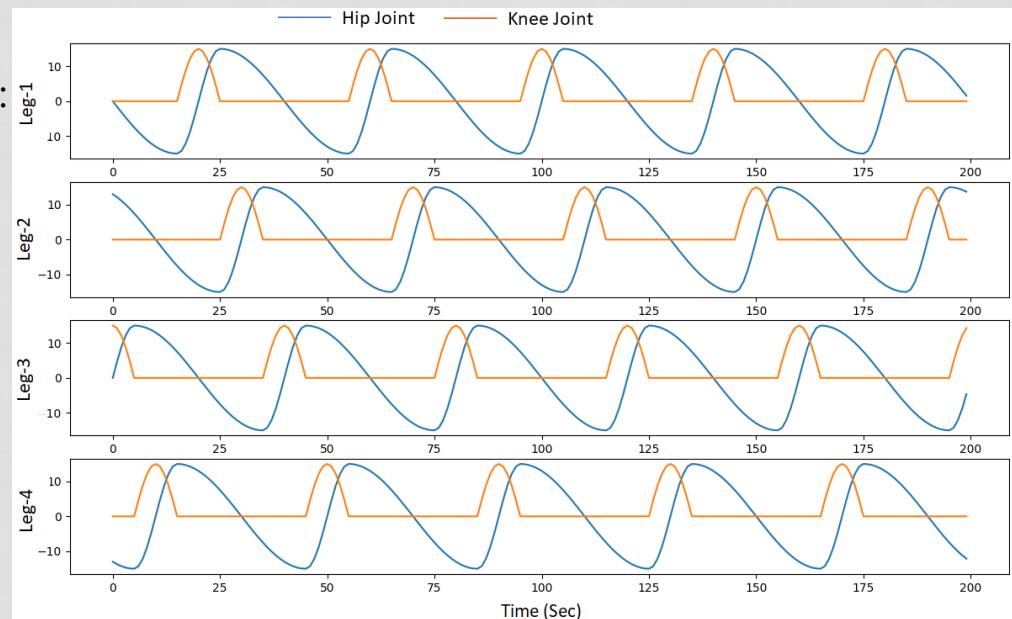
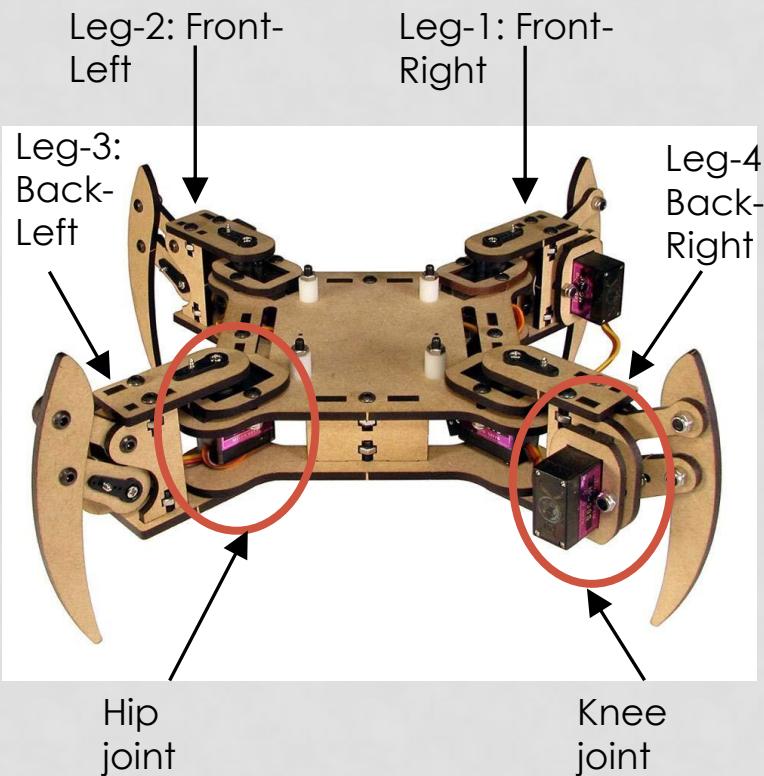
A feed-forward supervised oscillatory CPG model to control robotic locomotion



CURRENT IMPLEMENTATION

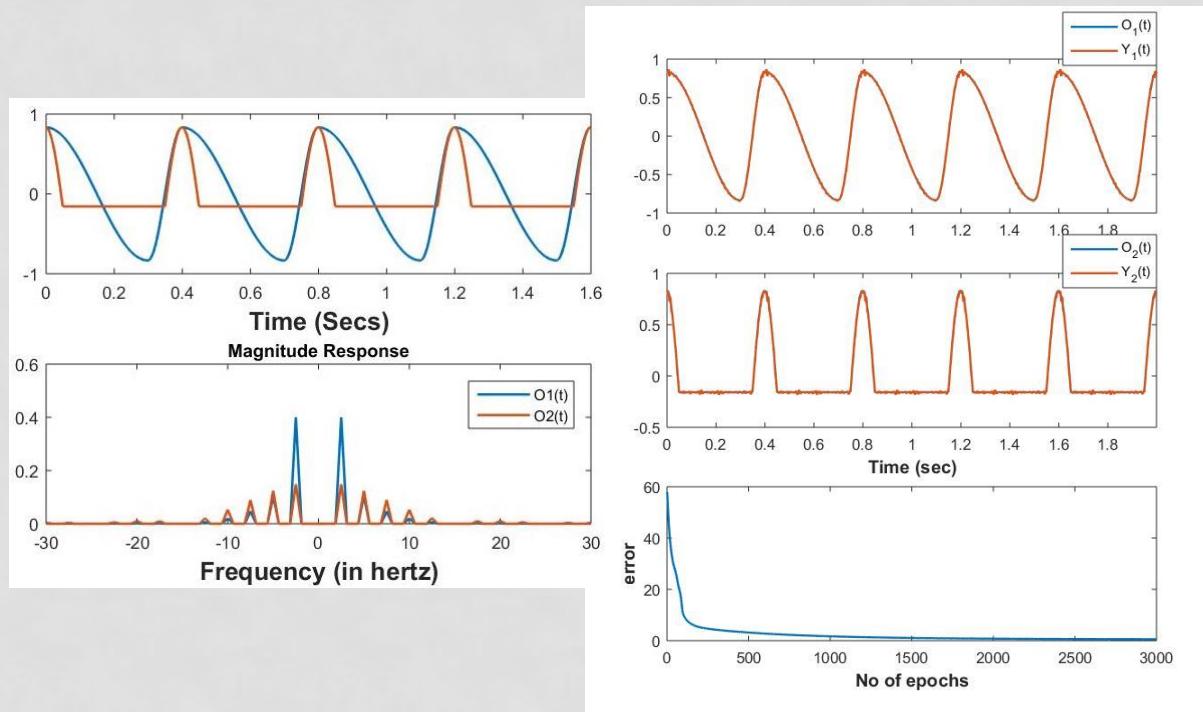


MEPED V2 QUADRUPED ROBOT:

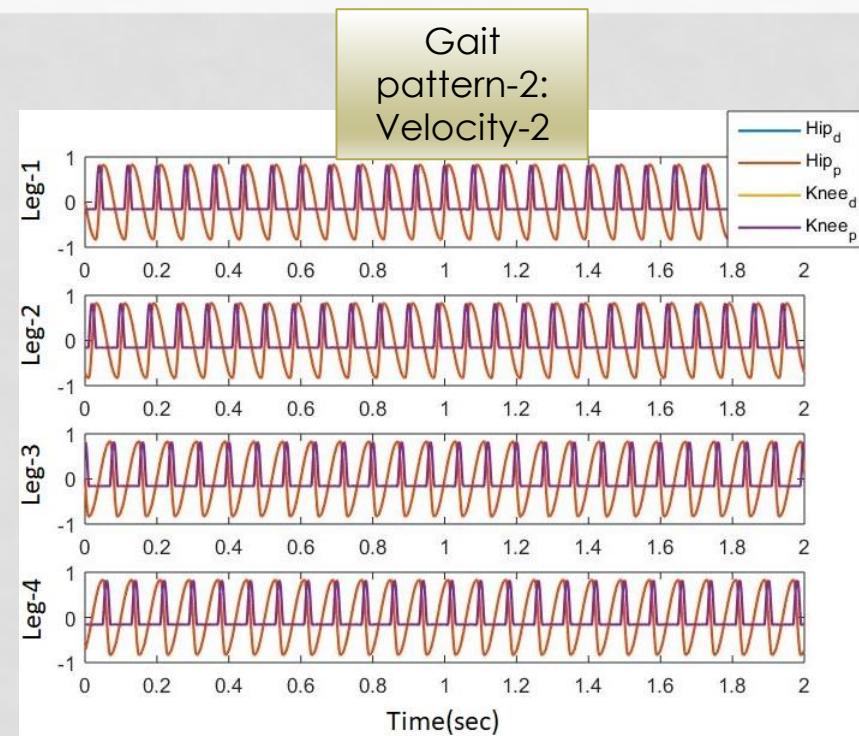
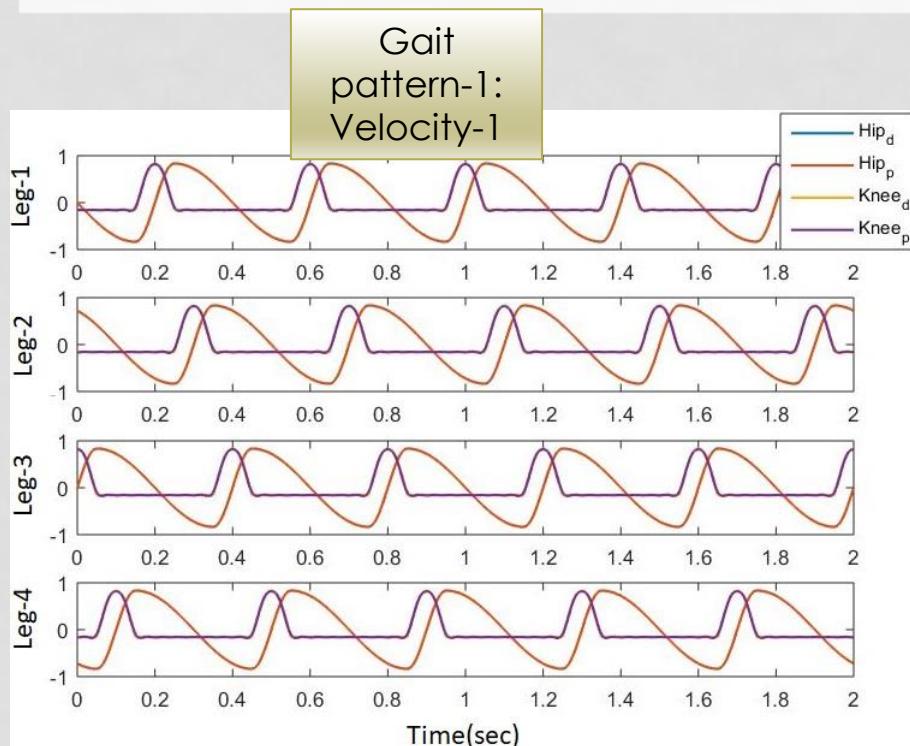


SUPERVISED CPG NETWORK TO PRODUCE GAIT RHYTHMS:

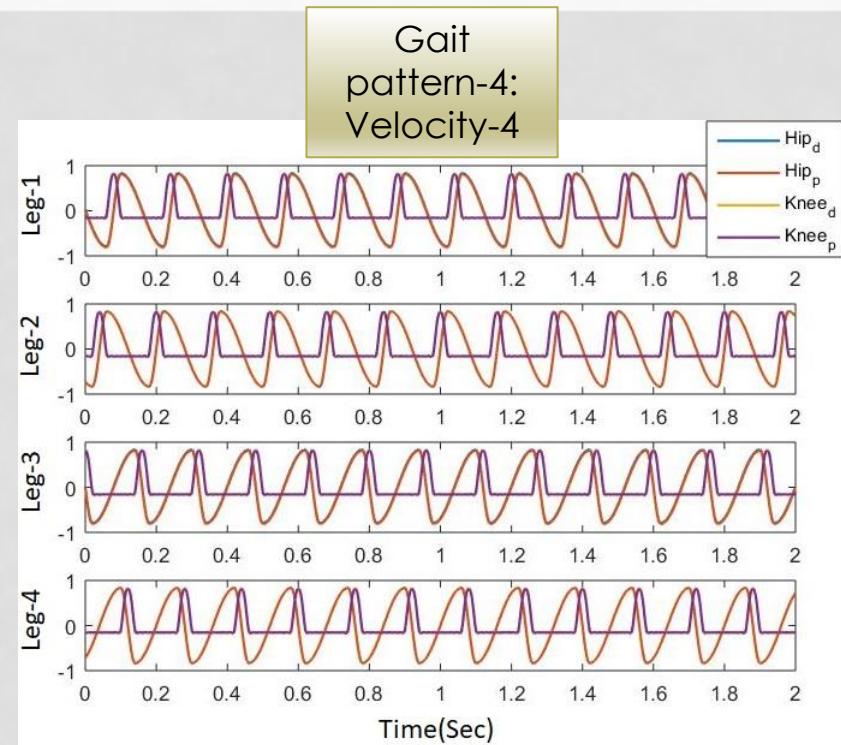
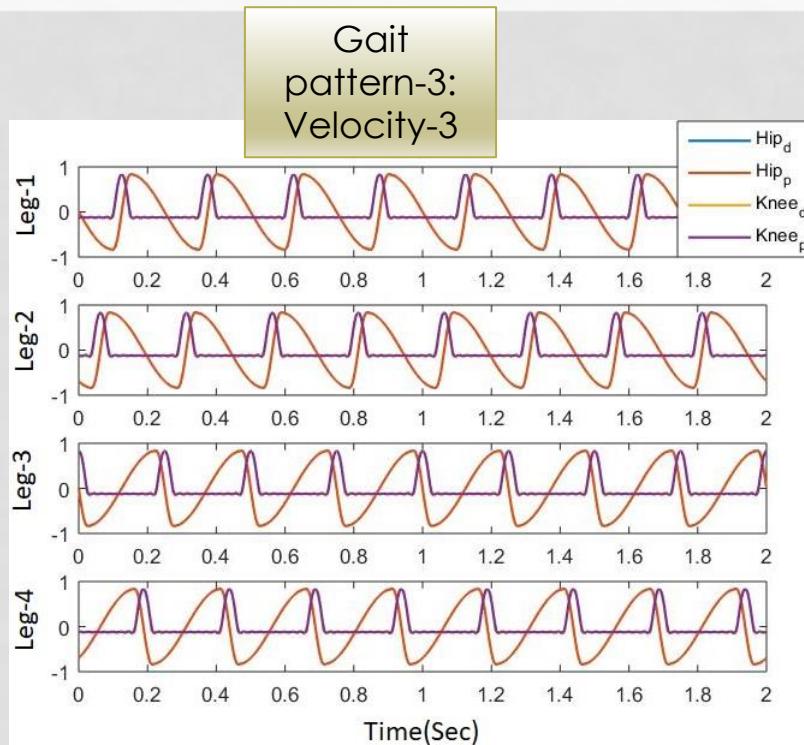
- As the gait rhythms are periodic in nature the constitutive frequency components are harmonics of the fundamental frequency (f_d) of the gait rhythm.
- A perceptron with complex weights projecting the activity of reservoir of oscillators with natural frequencies as harmonics of f_d can learn to produce desired gait pattern with reasonable accuracy.



THE FOUR DIFFERENT GAIT PATTERNS LEARNT BY THE NETWORK:



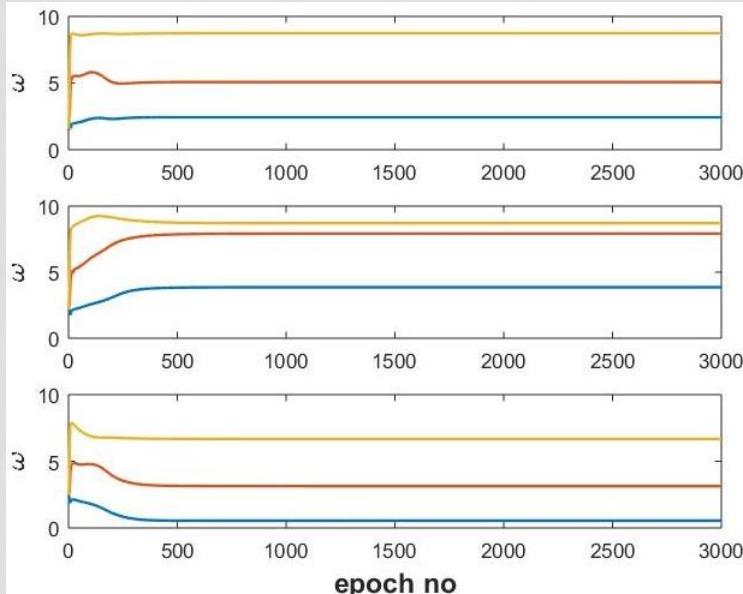
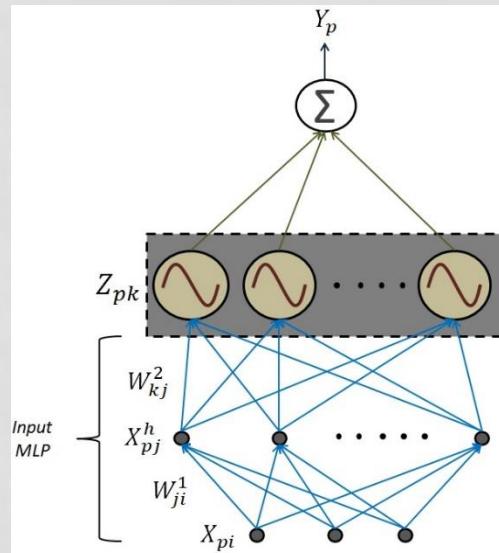
DESIRED AND THE NETWORK LEARNT GAIT PATTERNS:



A FEED-FORWARD HYBRID NETWORK WITH HOPF OSCILLATORS TO LEARN FOURIER DECOMPOSITION OF MULTIPLE SIGNALS

Network-1:

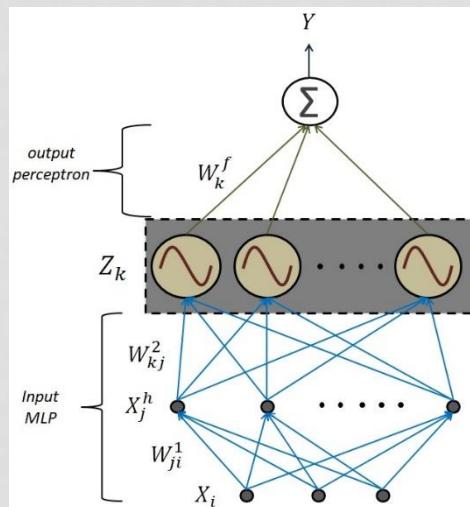
- The output of the “input MLP” is the natural frequencies of the oscillators.
- The parameters of the MLP are supposed to learn the frequency components in the desired output signal.



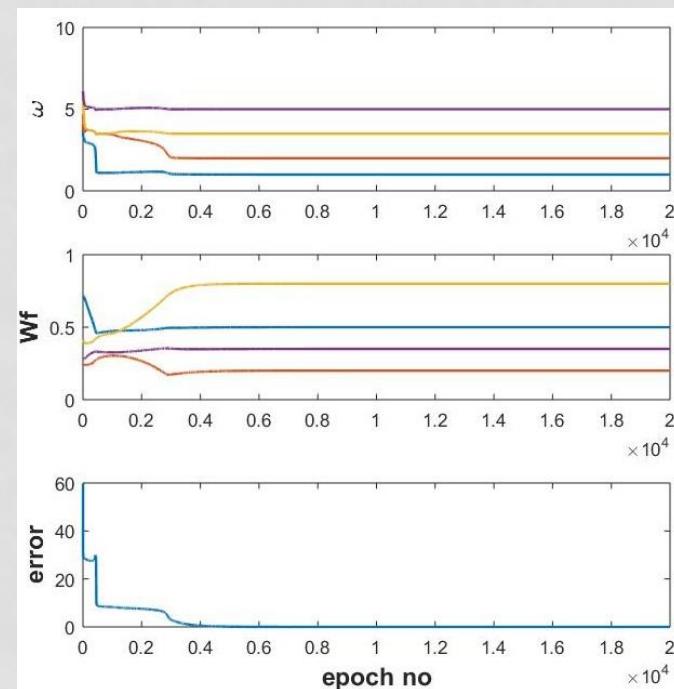
	Desired input			Desired output
1	0.7544	0.8892	0.7104	$\cos(8.7403t) + \cos(5.0725t) + \cos(2.4314t)$
2	0.2465	0.4514	0.9312	$\cos(3.8588t) + \cos(8.7079t) + \cos(7.9213t)$
3	0.6157	0.9312	0.6311	$\cos(6.6784t) + \cos(0.5457t) + \cos(3.1394t)$

NETWORK-2: CAN LEARN FREQUENCY AS WELL AS THE MAGNITUDE OF A OUTPUT SIGNAL

- The real weights of the “output” perceptron learn the magnitude of the various frequency components learnt by the oscillators.

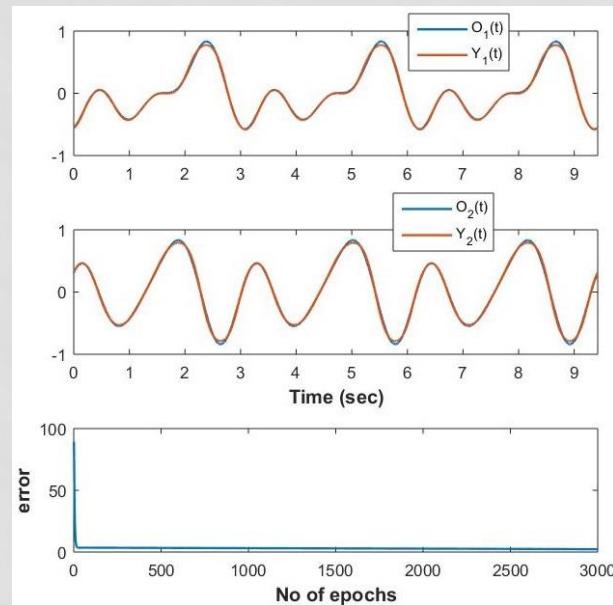
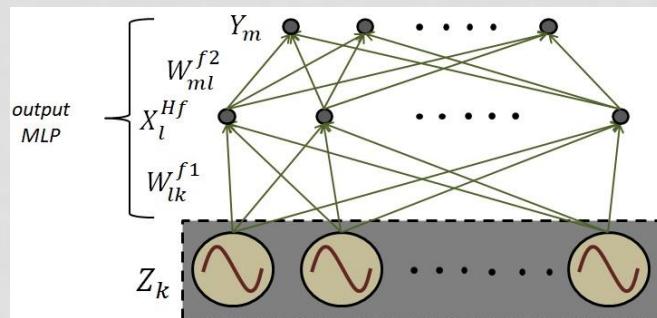


	Desired input			Desired output
1	0.4108	0.7703	0.3400	$0.2\cos(2t) + 0.8\cos(3.5t) + 0.5\cos(t) + 0.5\cos(5t)$



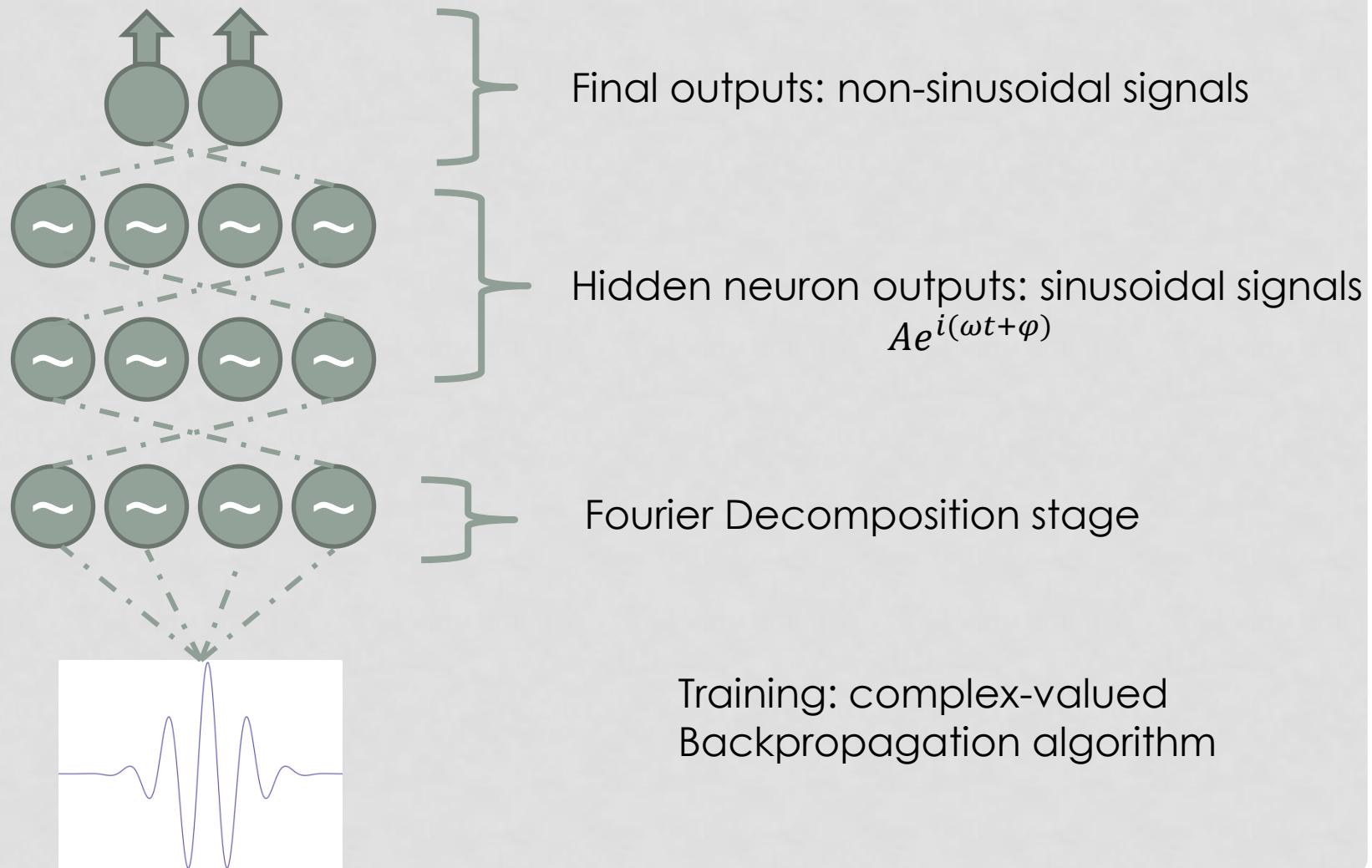
NETWORK-3: CAN LEARN THE MAGNITUDE AND THE PHASE OFFSET OF THE DESIRED OUTPUT SIGNAL

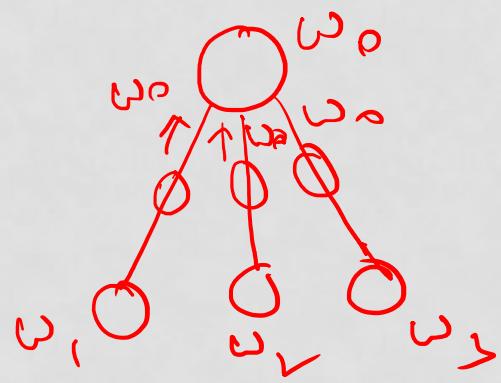
- Given that the constitutive frequency components of the desired output signals at the output nodes are same as the natural frequencies of the Hopf oscillators the network can learn to predict the output by tuning the parameters of the feed-forward complex MLP.



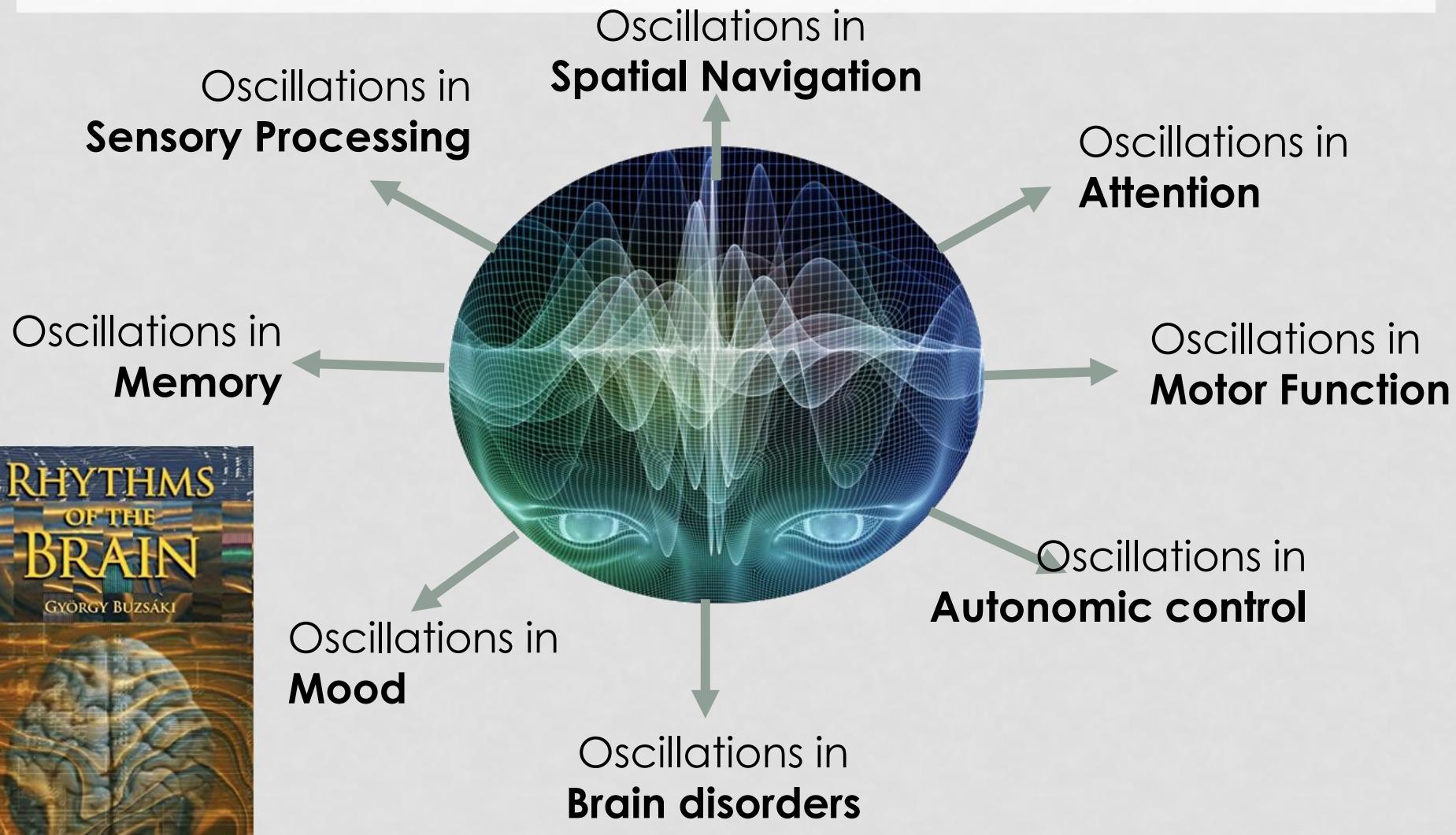
Desired input	Desired output 1	Desired output 2
$Z(t)$ $(\omega = [2; 4; 6])$	$0.9963 \cos(2t + 4.3585)$ + $0.7657 \cos(4t + 5.9176)$ + $0.6521 \cos(6t + 3.9679)$	$0.5754 \cos(2t + 3.8775)$ + $1.2634 \cos(4t + 4.0434)$ + $0.3763 \cos(6t + 0.9521)$

4. GENERALIZED OSCILLATORY DEEP NEURAL NETWORKS





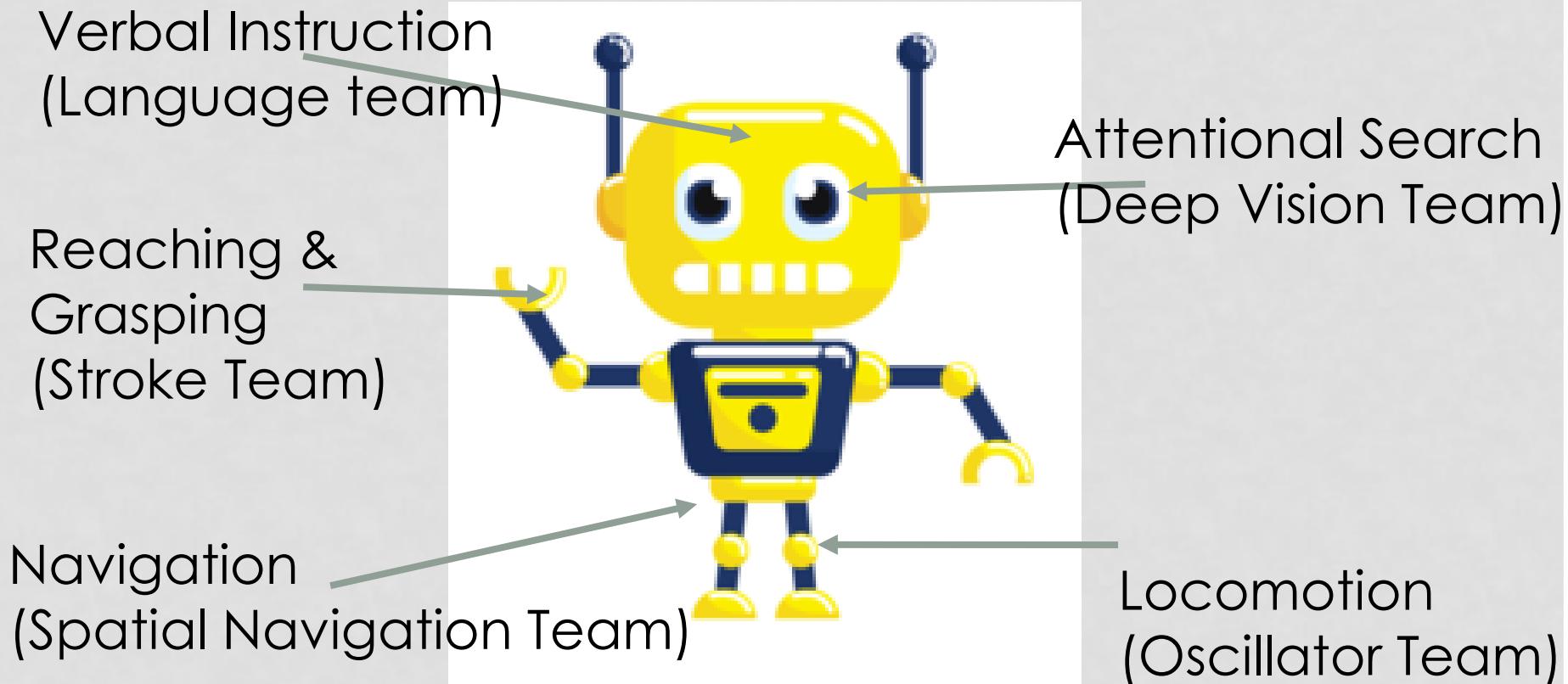
AN OSCILLATOR THEORY OF BRAIN FUNCTION



APPLICATION

HANU: HIERARCHICAL ADAPTIVE NAVIGATING UNIT

A SEARCH-AND-LOCATE ROBOT WITH
A BRAIN-INSPIRED NAVIGATIONAL SYSTEM



ACKNOWLEDGEMENTS

- Dipayan Biswas
- Aasit
- Surya Kiran
- Shreyas
- Ankit

thank
you

A Task-Optimized Neural Network Replicates Human Auditory Behavior, Predicts Brain Responses, and Reveals a Cortical Processing Hierarchy

Highlights

- A deep neural network optimized for speech and music tasks performed as well as human listeners
- The optimization produced separate music and speech pathways after a shared front end
- The network made human-like error patterns and predicted auditory cortical responses
- Network predictions suggest hierarchical organization in human auditory cortex

Authors

Alexander J.E. Kell, Daniel L.K. Yamins,
Erica N. Shook,
Sam V. Norman-Haignere,
Josh H. McDermott

Correspondence

alexkell@mit.edu (A.J.E.K.),
jhm@mit.edu (J.H.M.)

In Brief

Kell et al. show that a deep neural network optimized to recognize speech and music replicated human auditory behavior and predicted cortical fMRI responses. Different network layers best predict primary and non-primary voxels, revealing hierarchical organization in human auditory cortex.



A Task-Optimized Neural Network Replicates Human Auditory Behavior, Predicts Brain Responses, and Reveals a Cortical Processing Hierarchy

Alexander J.E. Kell,^{1,2,6,7,*} Daniel L.K. Yamins,^{3,4,6} Erica N. Shook,^{1,2} Sam V. Norman-Haignere,¹ and Josh H. McDermott^{1,2,5,*}

¹Department of Brain and Cognitive Science, MIT, Cambridge, MA, USA

²Center for Brains, Minds, and Machines, MIT, Cambridge, MA, USA

³Departments of Psychology and Computer Science, Stanford University, Stanford, CA, USA

⁴Stanford Neurosciences Institute, Stanford, CA, USA

⁵Program in Speech and Hearing Biosciences and Technology, Harvard University, Cambridge, MA, USA

⁶These authors contributed equally

⁷Lead Contact

*Correspondence: alexkell@mit.edu (A.J.E.K.), jhm@mit.edu (J.H.M.)

<https://doi.org/10.1016/j.neuron.2018.03.044>

SUMMARY

A core goal of auditory neuroscience is to build quantitative models that predict cortical responses to natural sounds. Reasoning that a complete model of auditory cortex must solve ecologically relevant tasks, we optimized hierarchical neural networks for speech and music recognition. The best-performing network contained separate music and speech pathways following early shared processing, potentially replicating human cortical organization. The network performed both tasks as well as humans and exhibited human-like errors despite not being optimized to do so, suggesting common constraints on network and human performance. The network predicted fMRI voxel responses substantially better than traditional spectrotemporal filter models throughout auditory cortex. It also provided a quantitative signature of cortical representational hierarchy—primary and non-primary responses were best predicted by intermediate and late network layers, respectively. The results suggest that task optimization provides a powerful set of tools for modeling sensory systems.

INTRODUCTION

Human listeners extract a remarkable array of information about the world from sound. These abilities are enabled by neuronal processing that transforms the sound waveform entering the ear into cortical representations thought to render behaviorally important sound properties explicit. Although much is known about the peripheral processing of sound, auditory cortex is less understood, particularly in computational terms. There is growing consensus that frequency and modulation tuning explain aspects of primary auditory cortical responses (Depireux

et al., 2001; Humphries et al., 2010; Miller et al., 2002; Santoro et al., 2014), but the organization of the rest of auditory cortex into regions and pathways remains unresolved, particularly in humans (Norman-Haignere et al., 2015; Rauschecker and Scott, 2009; Recanzone and Cohen, 2010).

Our understanding of auditory cortex is limited in part by the lack of quantitative models of how neural circuitry transforms sound waveforms into representations that enable behavior. Existing models of auditory processing are mostly limited to one or two stages, typically based on linear filtering of spectrogram-like input (Carlson et al., 2012; Chi et al., 2005; Dau et al., 1997; McDermott and Simoncelli, 2011; Mlynarski and McDermott, 2018). Such models explain aspects of auditory perception (McDermott et al., 2013; Patil et al., 2012) and cortical responses (Norman-Haignere et al., 2015; Santoro et al., 2014; Schönwiesner and Zatorre, 2009), but they are clearly incomplete. Neural responses are known to be nonlinear functions of the spectrogram (Christianson et al., 2008; David et al., 2009), and state-of-the-art machine hearing systems are highly nonlinear (Hershey et al., 2017), suggesting that auditory recognition requires invariances that cannot be obtained from the linear operations typically employed in auditory models.

In this paper, we develop a multi-stage computational model that performs real-world auditory tasks. The underlying hypothesis was that everyday recognition tasks may impose strong constraints on the auditory system, such that a model optimized to perform such tasks might converge to brain-like representational transformations. We optimized a deep neural network to map sound waveforms to behaviorally meaningful categories (words or musical genres), leveraging recent advances in what has become known as deep learning (LeCun et al., 2015). Although some aspects of such networks deviate substantially from biological systems, they are currently the only known model class that attains human-level performance on many real-world classification tasks. Following early hopes that such models would yield biological insights (Lehky and Sejnowski, 1988; Zipser and Andersen, 1988), contemporary deep neural networks have been shown to replicate key aspects of visual system



organization (Eickenberg et al., 2017; Güçlü and van Gerven, 2015; Yamins and DiCarlo, 2016). However, their utility for other brain systems remains unclear.

To evaluate the network, we compared its task performance with that of human listeners across a variety of conditions. The network recognized word and musical genres as well as human listeners did, and its error patterns resembled those of humans despite not being optimized to do so. We then used the network's features to predict fMRI voxel responses throughout auditory cortex, finding it to be substantially more predictive than the commonly used spectrotemporal filter model (Chi et al., 2005).

Motivated by these results, we used the network to address an unresolved question in auditory neuroscience: the extent to which auditory cortical computation is hierarchical—consisting of a sequence of stages, potentially corresponding to cortical regions (Okada et al., 2010; Rauschecker and Scott, 2009; Wessinger et al., 2001). In non-human animals, cytoarchitectonic and tracer studies are consistent with a tripartite hierarchical organization (Kaas and Hackett, 2000), and various sources of physiological evidence have been interpreted as supporting hierarchical organization (Atencio et al., 2012; Camalier et al., 2012; Chechik et al., 2006; Rauschecker et al., 1995; Recanzone and Cohen, 2010). However, the extent to which such findings generalize to humans is unclear, in part because of the unique importance of speech and music to human hearing. In humans, hierarchy is most commonly proposed for speech processing, where speech-specific responses only emerge outside of primary areas, suggestive of multiple processing stages (Chang et al., 2010; de Heer et al., 2017; Evans and Davis, 2015; Liebenthal et al., 2005; Norman-Haignere et al., 2015; Obleser et al., 2010; Overath et al., 2015; Peelle et al., 2010; Uppenkamp et al., 2006). Yet it remains unclear whether such regional differences reflect sequential stages of processing. Indeed, some have argued against hierarchy, instead proposing an anatomically distributed organization (Formisano et al., 2008; Staeren et al., 2009).

Our neural network model is intrinsically hierarchical, with the output of one stage forming the input to the next, and thus it provided a means of operationalizing and evaluating the complexity of responses in different parts of auditory cortex. This approach has proven fruitful in the visual system, where the presence of hierarchy is well established—different network layers best predict responses at different stages of the visual cortical hierarchy (Cichy et al., 2016; Güçlü and van Gerven, 2015; Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al., 2014). We used a similar approach to probe the relative complexity of responses in different parts of auditory cortex, where the large-scale organization is less settled. We find that intermediate model layers best explain primary auditory cortical responses, while deeper layers best explain voxels in non-primary areas. These results provide quantitative evidence of a computational hierarchy in human auditory cortex.

RESULTS

Network Tasks

To build our neural network model, we used two tasks that were behaviorally relevant and for which we could obtain large sets of

labeled data: word recognition and musical genre identification (Figure 1A). The word task required identifying which of 587 words was positioned at the midpoint of a 2 s excerpt of speech; the genre task required identifying which of 41 musical genres a 2 s music clip belonged to (see Tables S1 and S2 for all words and genres). Speech and music training examples were drawn from large, labeled corpora (Bertin-Mahieux et al., 2011; Garofolo and Consortium, 1993; Paul and Baker, 1992) and were superimposed on different types of real-world background “noise” to make the task more challenging and realistic (see STAR Methods for details). Whereas tasks similar to our word recognition task are arguably ecologically important to humans, the genre task was selected primarily because contemporary methods for training deep neural networks require large, labeled datasets, and genre tags, unlike other musical descriptors, are presently available for millions of music clips. The input to the network was a “cochleagram,” a time-frequency decomposition of the sound signal that mimics aspects of cochlear signal processing. The network parameters were optimized to map the cochleagram to class labels for each of the two tasks.

Network Architecture Optimization

The network consisted of a series of layers instantiating several standard operations: convolution with linear filters, pointwise nonlinearities, normalization, and pooling. Neural network training is most often associated with the optimization of network filter weights, but networks are also defined by architectural hyperparameters that can substantially affect performance (Pinto et al., 2009; Yamins et al., 2014; Zoph et al., 2017). These include the number of layers, number of units per layer, operations within each layer, filter sizes, and type of pooling operations. Good task performance can often be achieved using architectures that performed well on a related task (Razavian et al., 2014; Zoph et al., 2017). However, because the two tasks we used were relatively novel for convolutional networks, and because we wanted a single network to perform both tasks, we optimized across architectural hyperparameters in addition to the network filter weights. We selected the model architecture via a two-stage procedure, first searching for architectures that performed well on either task in isolation and then searching over ways of combining architectures into a single network that performed both tasks.

In the first stage, we generated nearly two hundred candidate architectures (Figure 1B; STAR Methods). For each architecture, filter weights were optimized via stochastic gradient descent for either the word or genre task alone. Millions of labeled training examples were generated by superimposing exemplars of each word or genre with background noise excerpts at various signal-to-noise ratios (SNRs). After training, performance for each architecture was assessed with left-out stimuli. The same architecture performed best on both tasks. This architecture had twelve layers of processing: five convolutional, three pooling, two normalization, and two fully connected layers (see STAR Methods for details).

In the second stage, we sought a single model that achieved good performance on both the word and genre tasks. A priori, it seemed plausible that speech and music (and potentially other) tasks could be performed using shared initial stages of acoustic

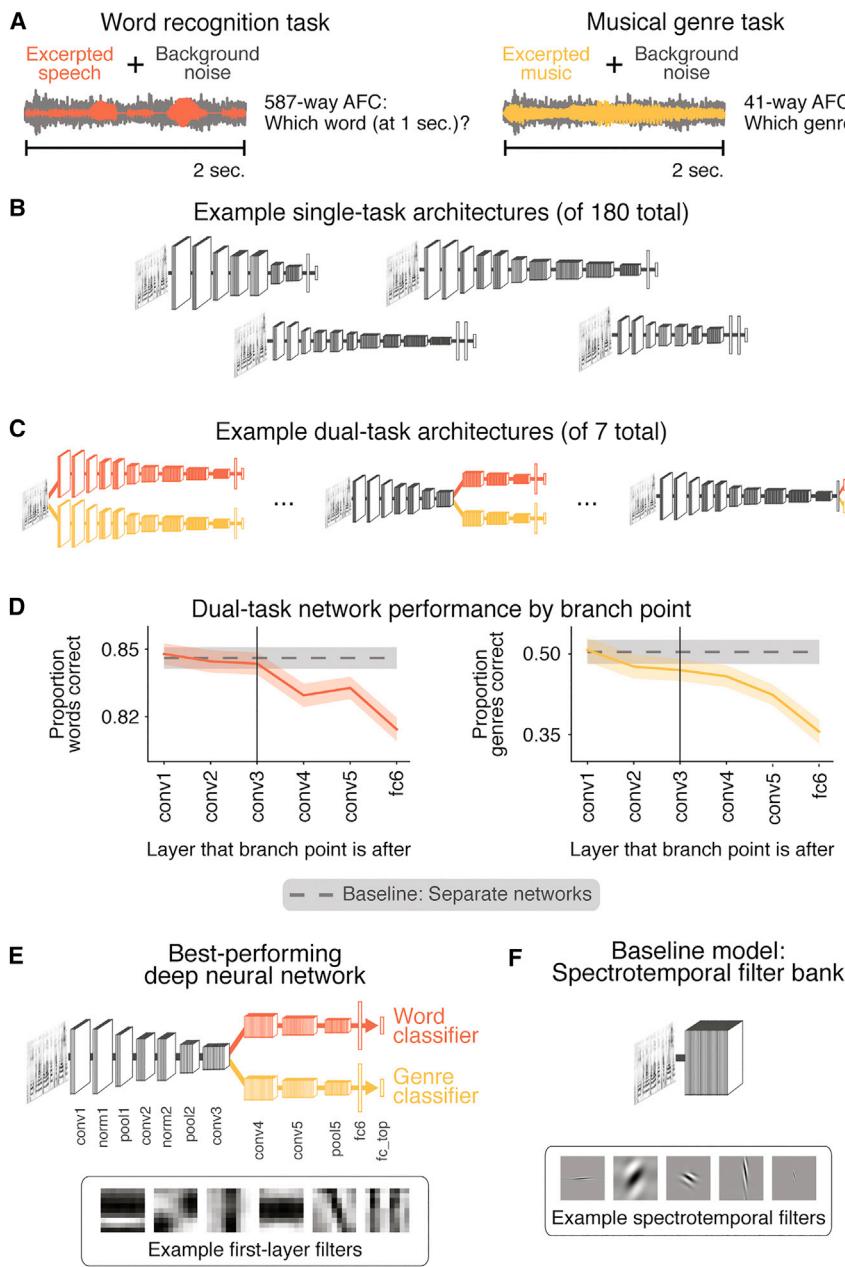


Figure 1. Deep Neural Network Training: Tasks and Architecture Search

(A) Tasks used for model optimization. The network received a 2 s clip of excerpted speech or music mixed with background noise (e.g., a recording of a city street). The network classified either which of 587 words occurred in the middle of the clip or from which of 41 genres the musical excerpt was drawn. (B) Example candidate single-task architectures. Architectures varied in the number of layers, size of kernels, etc.

(C) Example candidate dual-task architectures, generated by merging the best performing single-task architectures into a single branched network that performed both tasks. Left: architecture with no shared layers (i.e., two separate networks). Middle: an architecture with a few shared layers. Right: an architecture with nearly all shared layers.

(D) Task performance as a function of the branch point position within the network. We considered branch points at each convolutional or fully connected layer, because these are the only layers whose parameters are altered during task training (because they contain the filter weights optimized by backpropagation). Error bars plot SEM, bootstrapped over stimuli and classes. We sought to share as many layers of processing as possible without producing a performance decrement and thus selected the architecture that branched after the third convolutional layer (conv3; vertical black line).

(E) The model architecture that resulted from the task optimization procedure. “conv” denotes convolutional layers (always followed by rectification); “norm” denotes normalization layers; “pool” denotes pooling layers; “fc” denotes fully connected layers. Bottom: example first-layer filters.

(F) Schematic of a commonly used model of auditory cortex, consisting of a single stage of linear spectrotemporal filters on top of a model of the cochlea (a “cochleogram”). Bottom: example spectrotemporal filters.

analysis, after which they might require segregated domain-specific processing. We therefore created “branched” versions of the architecture found in the first stage of architectural optimization (Figure 1C), sharing some number of initial layers of processing before branching into two task-specific processing streams. Because task training did not alter the operations in pooling and normalization layers, the candidate branch points only preceded each of the seven convolutional or fully connected layers. We optimized the filter weights in each of these seven networks for both tasks jointly, using stochastic gradient descent. We then evaluated task performance.

The architecture with fully separate pathways performed better than the architecture with shared processing up until the

classification layers. This result is perhaps unsurprising given that the fully separate architecture has nearly twice as many parameters. However, architectures sharing a few early layers performed nearly identically to the fully separate architecture (Figure 1D). On grounds of parsimony, we selected the architecture that shared as much early processing as possible without significantly impairing task performance relative to the fully separate model (determined by bootstrap; STAR Methods). The selected architecture (Figure 1E) shared a total of seven layers, after which the network branched into two sets of five task-specific layers, with each branch culminating in output layers whose responses could be interpreted as probability distributions over classes for each task (i.e., words or genres). The results of this optimization suggest that some degree of speech- and music-specific processing is useful to achieve good task performance, but that shared early processing could be beneficial given a resource

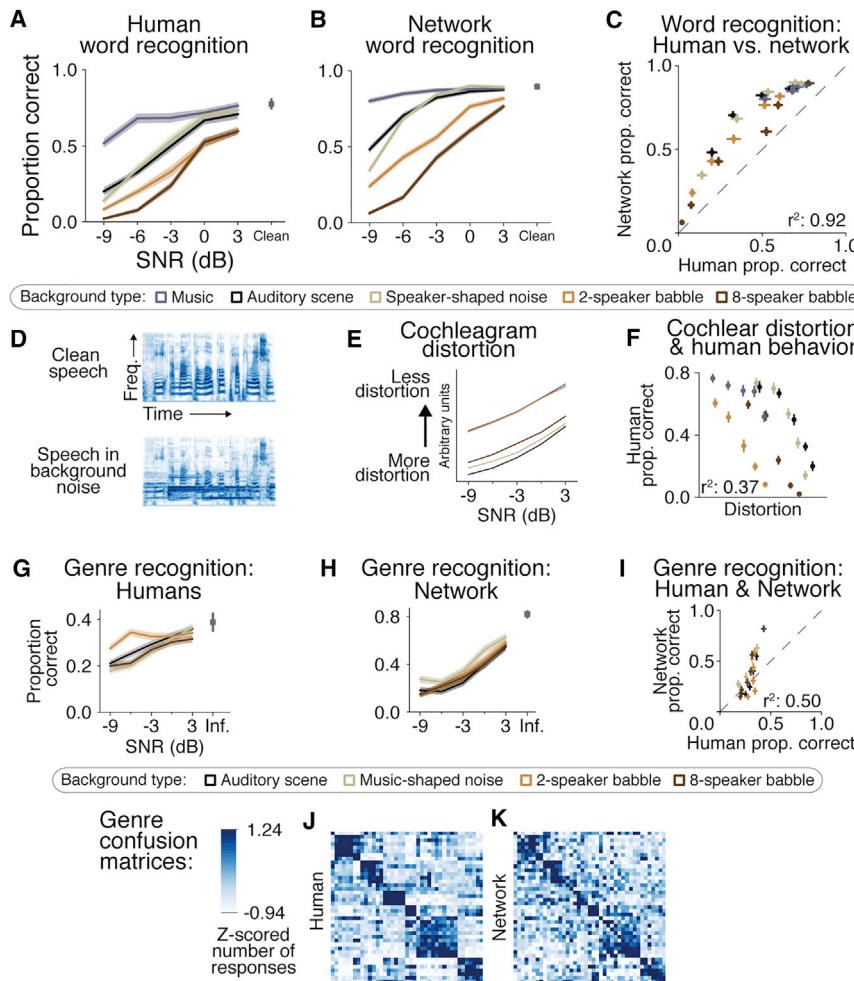


Figure 2. Comparison of Human and Network Behavior: Word and Genre Recognition Tasks

(A) Human performance on word recognition task ($n = 18$). y axis plots proportion of words correctly identified; x axis plots signal-to-noise ratio (SNR) of the speech signal. Each line plots performance for a particular type of background noise. Gray point on right plots performance without background noise (i.e., infinite SNR). Error bars plot within-subject SEM.

(B) Network performance on the word recognition task (using the same stimuli and task as for human listeners). Same plotting conventions as (A) were used. Error bars plot SEM, bootstrapped over stimuli and words.

(C) Scatterplot comparing word recognition performance of human listeners and the network for each background type at each SNR (i.e., identical data as A and B). Dashed line indicates unity.

(D) Example cochleograms of a speech signal with and without background noise. Distortion was computed as the mean absolute difference between these two cochleograms.

(E) Cochleogram distortion by condition. y axis is oriented to facilitate comparison with (A) and (B).

(F) Scatterplot of human performance and cochleogram distortion (i.e., data from A and E). The rank correlation between distortion and human performance is substantially lower than that between human performance and network performance (C). See Figure S1A for results when restricting distortion measurements to time-frequency bins with substantial speech signal power.

(G) Human performance on the genre classification task ($n = 111$). Same plotting conventions as (A) were used.

(H) Network performance on the same stimuli and task. Same plotting conventions as (B) were used.

(I) Scatterplot comparing genre classification performance of human listeners and the network for each background type at each SNR. Identical data as (G) and (H) are shown. Dashed line indicates unity.

(J) Genre confusion matrix for human listeners. Each column indicates the correct genre, and each row indicates the selected genre. Saturation denotes frequency with which human listeners selected a genre label for exemplars of a particular genre (Z scored within columns).

(K) Network confusion matrix. Same plotting conventions as (J) were used.

limitation (e.g., the number of neurons). The resulting network architecture is consistent with recent evidence for segregated speech and music pathways in non-primary auditory cortex (Angulo-Perkins et al., 2014; Leaver and Rauschecker, 2010; Norman-Haignere et al., 2015; Tierney et al., 2013).

Comparison of Model and Human Behavior

A complete model of the auditory system should replicate human auditory behavior. We thus began by measuring human performance for the word and genre tasks, comparing both absolute performance and the pattern of errors to that of the network. For the word classification task, listeners typed what they heard using an interface that auto-completed the 587 words. For the genre classification task, listeners selected the five most likely genres for the 2 s excerpt they heard. A “top 5” task was used to ensure that the task was reasonably well defined given overlap between different genres (e.g., “New Age” versus “Ambient”; see Figure S6B for overlap matrix).

In both cases the speech and music excerpts were presented in different types of background noise at a range of SNRs. We measured network performance on the same stimuli and tasks.

Word Recognition Behavioral Comparison

Human listeners’ word recognition performance improved with SNR, as expected, but some types of background noise impaired performance more than others (Figure 2A). The network exhibited similar absolute performance levels and similar dependence of performance on background noise ($r^2 = 0.92$, $p < 10^{-13}$; Figures 2B and 2C). The pattern of performance was not explained by simple measures of distortion in the cochlear representation of speech ($r^2 = 0.37$, lower than that with network performance, $p < 10^{-4}$; Figures 2D–2F). Figure S1A shows similar results when restricting the distortion measure to only those cochleogram bins with substantial speech energy; Figure S1B shows the results of distortion measured in network layers.

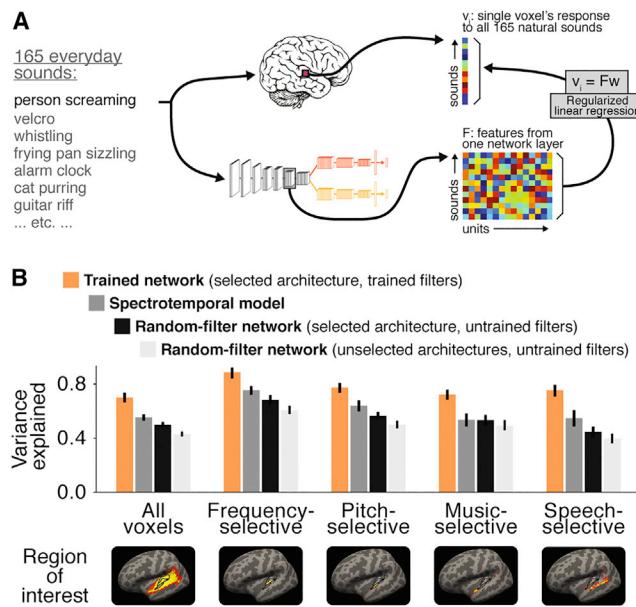


Figure 3. Predicting fMRI Responses to Natural Sounds: Comparison of Network with Baseline Models

(A) Schematic of method for predicting fMRI responses from neural network. We measured responses to 165 natural sounds in an fMRI experiment (Norman-Haignere et al., 2015), estimating each voxel’s average response to each sound (top). We then presented the same 165 natural sounds to the trained network and extracted each model unit’s time-averaged response to each of these sounds (schematized for one layer in bottom row). We modeled each voxel as a linear combination of model units from a given layer, estimating the linear transform with half the sounds and measuring the prediction quality by correlating the empirical and predicted response to the left-out sounds. We performed this procedure for each network layer and many random splits of the sounds.

(B) Variance explained in functionally localized regions of interest (ROIs). ROIs are shown below graph, with heatmaps of voxel counts across subjects. Black outlines show three anatomically defined sub-divisions of primary auditory cortex (TE 1.1, 1.0, and 1.2), taken from probabilistic maps (Morosan et al., 2001). Orange bar denotes the trained network, dark gray denotes the spectrotemporal filter model, black denotes a network with the identical architecture but with random, untrained filters, and light gray denotes the results from many random-filter networks with different architectures (bar plots median across architectures). Error bars are within-subject SEM.

Genre Recognition Behavioral Comparison

The network also approximately replicated human performance levels on the musical genre task (Figures 2G–2I). Because the number of genres was modest, we compared confusion matrices (this was impractical for the word task because the number of words made the confusion matrix prohibitively large). Confusion matrices for humans and the network were significantly correlated (Figures 2J and 2K; Spearman $r^2 = 0.25$, $p < 10^{-104}$).

Taken together, the behavioral analyses suggest that the performance-optimized neural network replicates non-trivial patterns in human speech and music perceptual behavior, despite not being explicitly optimized to do so.

Predicting Cortical Responses from Network Features

We next examined potential correspondence between representations in the network and auditory cortex, measuring how well

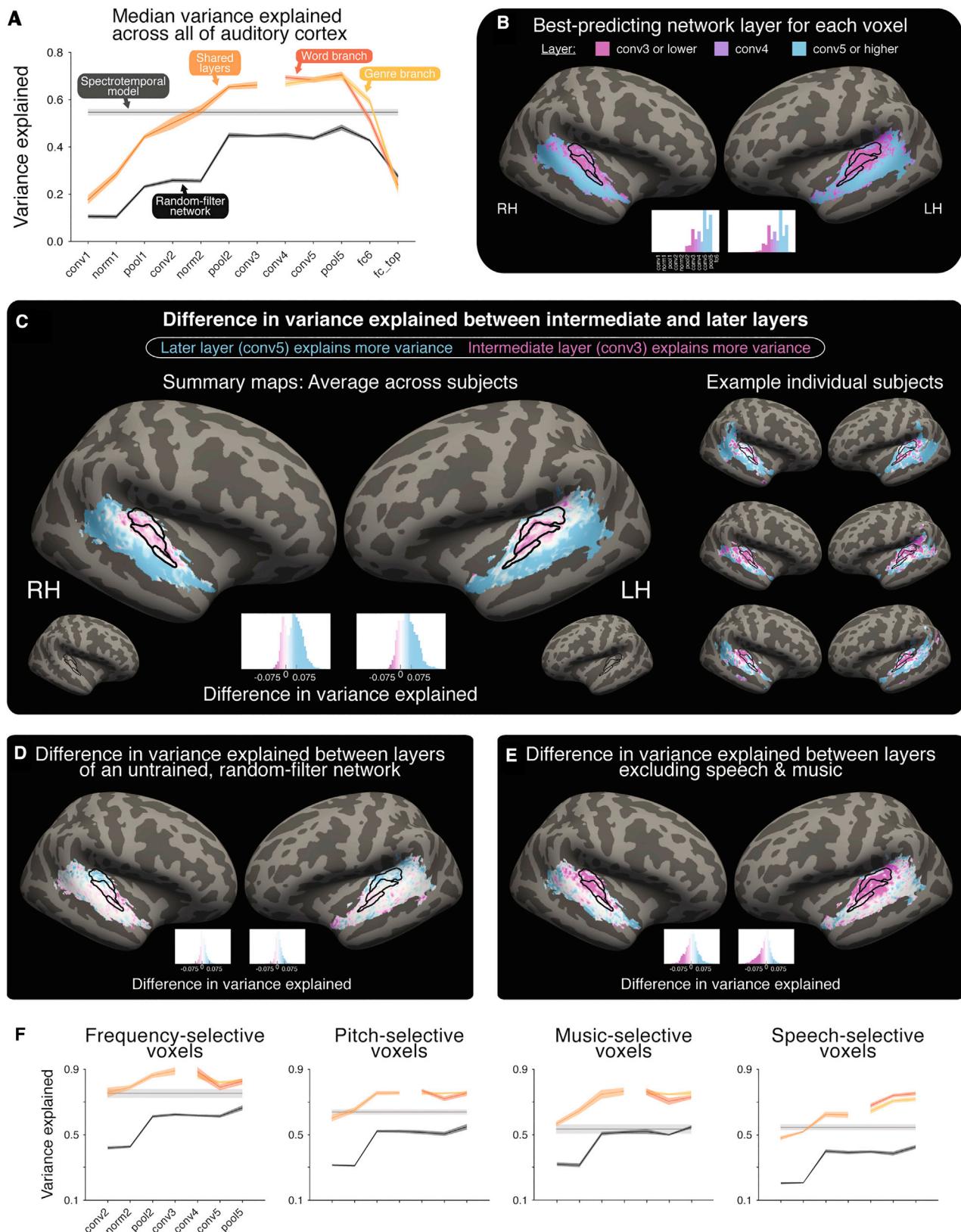
different network layers could predict fMRI voxel responses to a broad sample of 165 natural sounds (Table S3). Some of these sounds were speech and music, but most (113 of 165) were not. We measured the response of model units to each sound and predicted each voxel’s response from the time-averaged model unit responses from each layer using regularized linear regression (Figure 3A). Time averaging (over the duration of the sound) was used because the blood-oxygen-level-dependent (BOLD) signal is sluggish relative to the 2 s stimulus duration.

Responses from model units were linearly combined to produce a “synthetic voxel” that best approximated the measured voxel response. This general procedure has become standard for evaluating encoding models of brain responses (Güçlü and van Gerven, 2015; Klindt et al., 2017; Naselaris et al., 2011; Santoro et al., 2014; Yamins et al., 2014), the rationale being that the linear transformation discovered by regression aligns brain and model response spaces as best possible without (nonlinearly) distorting the representations. We measured the amount of BOLD variance predicted in left-out sounds, correcting for both the reliability of the measured voxel response and the reliability of the predicted voxel response (Schoppe et al., 2016; Spearman, 1904). We compared the variance explained by the network features, the spectrotemporal filters in the standard cortical model, and the features from a neural network with untrained (i.e., random) filter weights.

Voxelwise Variance Explained across Auditory Cortex

To get an overall sense of the quality of the model predictions, we first computed the median variance explained across all auditory cortical voxels. The network model explained substantially more variance than the spectrotemporal model (69.9% versus 55.1%; $p < 0.001$, paired t test; Figure 3B). We used a relatively large parameterization of the spectrotemporal model that yielded at least as many features as any neural network layer. To ensure a fair comparison, we verified that our parameterization of the spectrotemporal model saturated the amount of variance explainable by such a model for these data (Figure S2C).

A priori, it seemed possible that the improved voxel predictions could arise merely from the intrinsic hierarchical organization of a deep neural network, irrespective of the particular architecture and filters produced by task optimization. For instance, a feedforward network with convolution and pooling produces a range of receptive field sizes across its layers, some of which might match cortical receptive field sizes. To control for this possibility, we measured predictions from a network with the selected model architecture but with random, untrained weights. As shown in Figure 3B, the random network features explained a substantial fraction of voxel response variance, consistent with previous findings that random nonlinear functions are useful for regression (Lukoševičius and Jaeger, 2009). However, the random network predicted voxel responses substantially worse than the trained network (paired t test on variance explained across all voxels, $p < 0.001$) and the spectrotemporal model (paired t test, $p < 0.05$). Alternative network architectures (also with random weights) yielded even worse response predictions (summarized by the median across architectures, paired t test $p < 0.05$). Overall, these results indicate that task optimization across both architectures and filters was important for achieving good cortical response predictions.



(legend on next page)

Voxelwise Variance Explained in Regions of Interest

To evaluate the quality of voxel response predictions in different parts of auditory cortex, we examined the variance explained within four functionally defined regions of interest (ROIs), localized in each participant with independent data: frequency-, pitch-, music-, and speech-selective voxels (Figure 3B; see STAR Methods for voxel selection details). In all cases, we took the top 5% of all reliable voxels when ranked according to the ROI criterion, excluding any voxels that were included in multiple ROI definitions (Figure S3A). The network model's predictions were better than the standard spectrotemporal filter model and the random filter network in each ROI (paired t tests, $p < 0.001$). Figure S3B shows that results were robust to varying the ROI criterion threshold. Taken together, these results show that our network explains responses to natural sounds substantially better than the spectrotemporal filter model throughout auditory cortex.

Assessing Hierarchical Organization

Because the network transforms its acoustic input via a feedforward cascade, responses of later network layers result from more nonlinear operations than those of earlier layers. We sought to leverage this property to assess the potential hierarchical position of different portions of auditory cortex, comparing voxel responses with responses from different network layers.

Before examining potential differences between different parts of auditory cortex, we first examined the ability of each network layer to predict voxels across all of auditory cortex. The median variance explained increased across layers up until the final two layers, after which it decreased (Figure 4A). Moreover, all but the earliest and latest layers of the trained network surpassed the predictions of the spectrotemporal model. See Figure S2B for significance of individual voxel predictions and Figures S4A and S4B for a map of variance explained across the auditory cortex.

In addition, nearly every layer of the trained network explained more variance than the corresponding layer of the random filter network (paired t tests, all $p < 0.01$, except for final layer), though the dependence on layer for the random filter network was nonetheless coarsely similar to that of the trained network (Pearson's $r = 0.88$, $p < 0.001$). Here, we show the results for one random filter network, but results were similar with different samples of random weights, shown in Figure S2D. These find-

ings are consistent with the idea that the “receptive field” sizes of particular layers (determined by the network architecture) may be well matched to that found in auditory cortical regions but also suggest that task optimization is critical to produce model features that replicate cortical tuning properties. The poor predictions by the final layers of the network could be due to a mismatch in the scale of the resulting features, which pool information across the full duration of the input. The poor predictions could also reflect the fact that final network layers lead up to perceptual decisions. The neurons underlying such decisions might be present in auditory cortex but spatially organized so as to be inaccessible with conventional fMRI, or could reside outside of auditory cortex altogether (e.g., in frontal or parietal areas).

Given that the very early and very late layers were poor predictors of auditory cortical responses, in subsequent analyses we restricted attention to the layers in between.

Hierarchical Organization: Maps

To examine the relationship between the network stages and potential stages of auditory cortex, we determined the best-predicting layer for each reliably sound-responsive voxel in auditory cortex. As shown in Figure 4B, intermediate layers best predicted the voxels in what is classically considered primary (“core”) auditory cortex (denoted by the black outlines). Beyond the core, in either anterior, lateral, or posterior directions, the deeper layers provided the best predictions. The results are consistent with the idea that primary and non-primary cortex are situated at distinct hierarchical stages of processing.

Although intuitive, selecting the best-predicting layer for each voxel is categorical and thus does not convey the magnitude of the difference in prediction quality between layers. For a continuous measure of hierarchical position, we additionally measured the difference in variance explained by intermediate and deep layers of the trained network for each voxel. Here we show layers conv5 and conv3, but the general pattern is robust to the particular choice of pairs of layers (Figure S5A). The summary map in Figure 4C averages across individual-subject results, but a similar pattern was evident in individual subjects (Figure 4C, right; see Figure S5B for all eight individuals). The results are again consistent with hierarchical organization. Notably, the analogous map generated from the random-weights network (Figure 4D) does not exhibit signs of hierarchical structure. This result suggests that the differences evident across cortical

Figure 4. Using the Network to Probe Hierarchical Organization in Human Auditory Cortex

- (A) Median variance explained across all voxels in auditory cortex for each network layer. Orange denotes results for trained network (break in curve corresponds to branch point). Black denotes results for identical architecture but with random, untrained filter weights. Gray line plots the variance explained by the spectrotemporal filter model for comparison. Error bars are within-subject SEM.
- (B) Map of best-predicting layer for each voxel (median across subjects). Inset on right shows color scale and histograms for each hemisphere.
- (C) Map of the difference in voxel response variance explained by later and intermediate network layers for the trained network (conv5 versus conv3). Maps on left show mean across subjects; right is three example individual subjects. Below is histogram/color bar of all values for each hemisphere. Black outlines show three anatomically defined sub-divisions of primary auditory cortex (TE 1.1, 1.0, and 1.2), taken from probabilistic maps (Morosan et al., 2001).
- (D) Map of the difference in voxel variance explained by higher and lower network layers for the untrained, random-filter network (conv5 versus conv3). Conventions, including color scale, are same as (C).
- (E) Map of the difference in variance explained by later and intermediate layers (conv5 versus conv3), excluding speech and music stimuli from the regressions. Analogous to (C), with same conventions and color scale as (C) and (D).
- (F) Layerwise variance explained in functionally localized regions of interest for the trained network and random-filter network. Same plotting conventions as (A) were used, except that, for clarity, only intermediate layers (where predictions exceeded those of the spectrotemporal model) are plotted. Error bars are within-subject SEM.

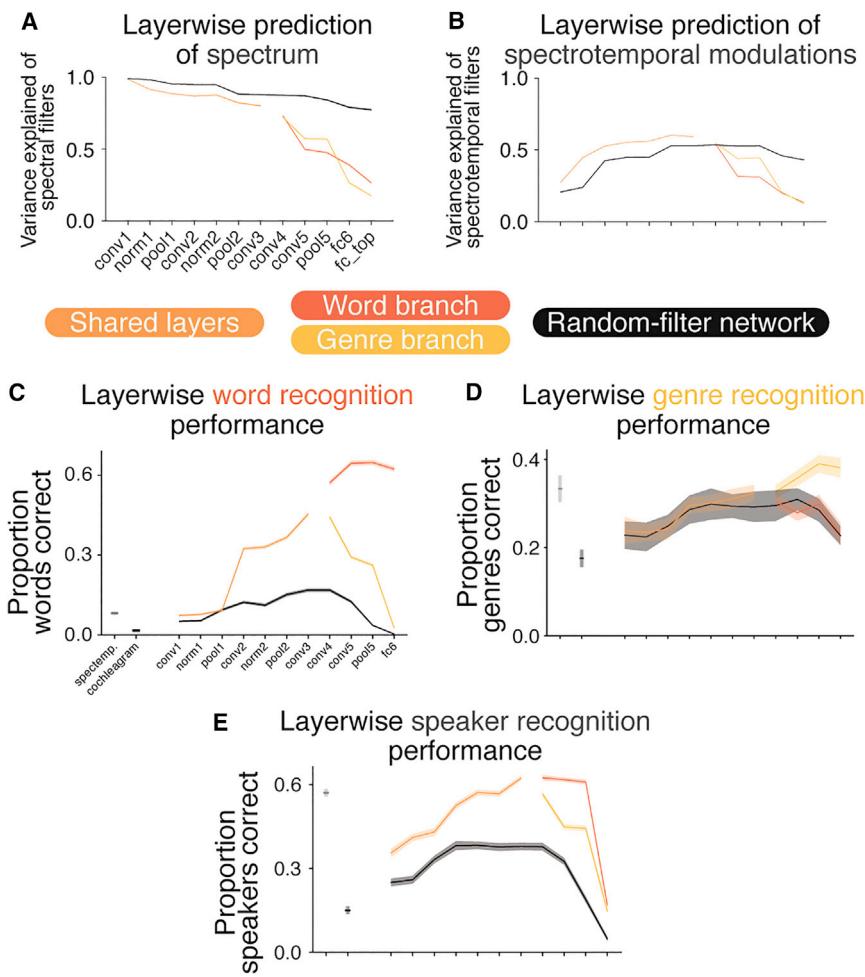


Figure 5. Analysis of Network Representations

(A) Layerwise predictions of the frequency spectrum (as measured by the time-averaged output of the cochlear model that provides the input to the network) from network features. Here and elsewhere, error bars plot one SEM, obtained by bootstrap, and in many cases are so small as to not be clearly visible.

(B) Layerwise predictions of spectrotemporal modulation power (measured from the baseline model shown in Figure 1F) from network features.

(C) Layerwise classification of spoken words using network features.

(D) Layerwise classification of musical genre using network features.

(E) Layerwise classification of speaker identity using network features.

regions reflect tuning properties learned by the network model in the service of auditory task performance.

Given that our network was trained on speech and music tasks, it is natural to wonder whether our evidence for hierarchical cortical organization is simply a reflection of the music and speech selectivity previously reported in non-primary auditory cortex (Angulo-Perkins et al., 2014; Leaver and Rauschecker, 2010; Norman-Haignere et al., 2015; Overath et al., 2015; Tierney et al., 2013). For instance, later layer responses might better serve to detect the presence or absence of speech or music, which could help predict the response of speech- or music-selective voxels. However, even when music and speech stimuli were omitted from the set of 165 sounds (leaving 113 stimuli), the general pattern of results persisted—earlier layers best predicted voxels in the “core,” while non-primary voxels were best predicted by later layers (Figure 4E). The hierarchical structure revealed by our network model thus appears to reflect the complexity of cortical responses to everyday sounds more generally.

Hierarchical Organization: Regions of interest

To further assess hierarchical structure, we examined network layer predictions for voxels within the four functionally defined cortical ROIs from Figure 3B. For each network layer, we pre-

dicted the responses of individual voxels to all 165 natural sounds and summarized the predictions with the median explained variance across voxels in each ROI. For comparison, we again measured the variance explained by the spectrotemporal filter model and by each layer of the same architecture with random weights. The four ROIs produced distinct layerwise predictivity curves (Figure 4F). Frequency-selective (tonotopic) voxels were best explained by intermediate layers of the network. By contrast, pitch- and music-selective voxels were roughly equally well predicted by intermediate and deep layers. However, the increase in variance

explained by the network compared to the spectrotemporal model was significantly larger for music voxels compared to pitch voxels, producing a model-by-region interaction for later network layers ($p < 0.05$ for each layer after pool2). Speech-selective voxels were best explained by deep layers of the network, with a large advantage for the trained network over the spectrotemporal model. Finally, the word and genre branches best predicted speech- and music-selective voxels, respectively, as expected (see Figure S3E for similar results with networks trained on either task individually). Results were again robust to the threshold used to define ROIs (Figure S3C).

The results differed substantially for the random-weight network, whose predictions were consistently lower than those of the trained network (black lines in Figure 4F; paired t test all $p < 0.005$) and were never better than those of the spectrotemporal filter model. Consistent with the map results of Figure 4D, the dependence on network layer did not differ across ROIs (no interaction, $p = 0.99$, unlike the trained network, $p < 0.05$). Indeed, there was little variation in the variance explained by layers beyond pool2 (the slight increase from conv5 to pool5 for the network shown in the figure is inconsistent across multiple random filter initializations; Figure S2D; see also Figure 5 for analogous results with predictions of acoustic features).

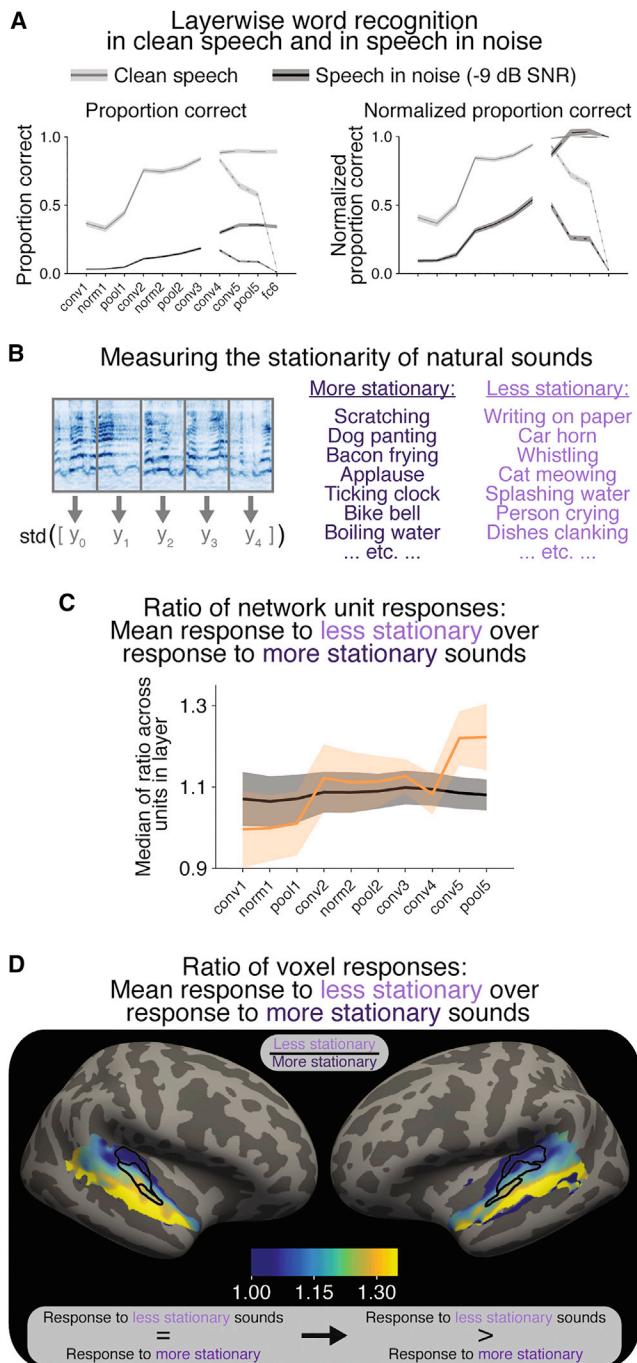


Figure 6. Analysis of Network and Brain Responses to Sound Stationarity

(A) Layerwise classification of spoken words with and without noise. Left graph shows raw performance; to aid comparison of the shape of these curves across layers, right graph shows performance normalized by classification performance of the network's output layer. Here and elsewhere, error bars plot one SEM, obtained by bootstrap.

(B) Schematic of stationarity measure, based on the variability of texture statistics measured in short time windows.

(C) Ratio of mean network unit responses to most and least stationary sounds selected from a set of natural sounds. Error bars are larger than in other plots due to the modest size of the sound sets.

These results suggest that task optimization was critical to instantiating differentiated features across layers that were well matched to auditory cortical tuning.

Overall, the ROI analyses provide further support for hierarchical organization. The results are consistent with the idea that tonotopic voxels (best explained by intermediate layers of the network) are situated early in a cortical hierarchy. By contrast, pitch-, music-, and speech-selective voxels appear to be situated later, best explained by later layers that instantiate more complex functions of the acoustic input.

Analysis of Network Representations

The network and cortex both appear to be organized hierarchically, but what representational transformations do they instantiate? To explore this question, we took advantage of the ability to interrogate the model representations at will.

Predicting Acoustic Features

We first examined the network's representation of standard acoustic features: those captured by a model of the cochlea (which provides the network's input) and the baseline spectrotemporal modulation model (Figure 1F). We examined whether these two types of information were explicit in each layer's representation (i.e., linearly decodable), using a similar procedure to that which we employed to predict voxel responses. We fit a linear mapping from the network features to the acoustic features using one subset of natural sounds and measured the quality of the resulting predictions with another subset. The ability to extract spectral information decreased from early to late layers of the network (Figure 5A), whereas the ability to extract spectrotemporal modulations peaked in intermediate layers (Figure 5B). Later layers predicted both sets of features worse than earlier layers, and this decrease was accentuated in the trained network compared to an untrained, random-filter network, indicating that it is not simply due to the network architecture.

Real-World Task Performance

We next tested the extent to which the network features in each layer could be used to perform tasks: the word and genre tasks that the network was trained on and, to examine generalization, a speaker identification task that played no role in training. We examined performance for each layer by fixing all the weights in the network and optimizing a linear (softmax) classifier that took the output of a given layer as its input. Unlike predictions of standard acoustic features, word and genre classification improved from early to late network layers, differing substantially between the two task-specific branches (Figures 5C and 5D). With the exception of the very last layer of the network, this pattern also held for the speaker task (Figure 5E). The speaker classification performance suggests that the learned network representations generalize at least somewhat to other tasks. Taken together, these analyses indicate that task-related information implicit in the cochlear representation is gradually transformed to become more explicit (see Figure S6 for complementary results using representational similarity analysis).

(D) Ratio of voxel responses to the same sets of most and least stationary sounds.

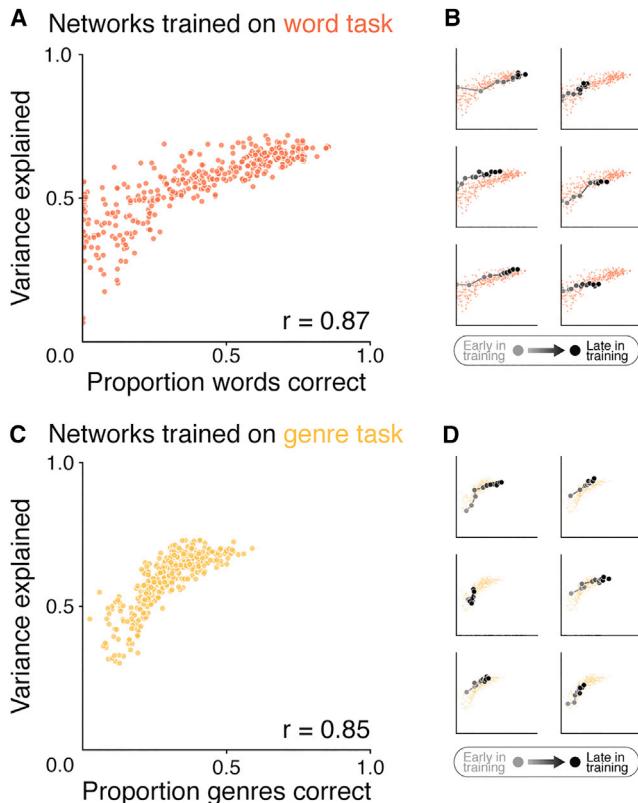


Figure 7. Network Task Performance Correlates with Cortical Predictivity

(A) Scatterplot of the median variance explained by a network across all voxels in auditory cortex versus the proportion of words correctly identified by that network, for networks trained on the spoken word recognition task. Each point is a network (a certain architecture with a certain amount of training). The networks were only optimized for task performance (i.e., the x axis).

(B) Same data as in (A), but with a single network architecture's performance and predictivity evaluated over training epochs plotted in gray/black. Earlier training points are gray; later training points are black. Each panel highlights a different architecture.

(C) Analogous to (A), but for networks trained on the musical genre recognition task. Conventions are as in (A).

(D) Analogous to (B), but for networks trained on the genre task. Conventions are same as (B).

Sensitivity to Noise-like Sounds: Testing a Model Prediction

To further characterize the network's representational transformations, we examined how background noise affected task performance in each layer. We took the results from the word classifiers used in Figure 5C and split them up according to the amount of background noise. Figure 6A shows that in the absence of noise, spoken words could be classified relatively well from intermediate network layers (e.g., conv2 or norm2). By contrast, classification of speech in noise did not approach asymptotic performance levels until later network layers (e.g., conv4 or conv5), indicating that representations in later layers are more noise-robust (see Figure S7A for effect of intermediate SNRs).

The noise robustness of the later layers motivated us to examine their sensitivity to noise-like stimuli. We measured the

response in each network layer to two sets of natural sounds: one with relatively stable statistical properties (i.e., stationary and thus noise-like), and one with less stable statistics. We quantified stationarity by dividing a sound's cochleagram into temporal bins, extracting perceptually relevant sound statistics in each bin, and taking the standard deviation over time (Figure 6B; see STAR Methods for details). The stimuli were subsets of those from the fMRI experiment, but with speech and music excluded to ensure that any observed effect was not simply due to selectivity for these sound classes. We then measured each network layer's mean response to the two sets of natural sounds. As shown in Figure 6C, deeper layers of the trained network exhibited a larger response to non-stationary compared to stationary sounds. This effect was not observed for the untrained network, suggesting it was not simply due to the extent of spectral and temporal integration in later layers.

To examine whether a similar effect differentiated primary from non-primary cortex, we compared voxel responses to the same sets of sounds used in the network analysis (taken from the fMRI dataset used throughout this paper). As shown in Figure 6D, responses to stationary and non-stationary sounds were comparable in and around primary auditory cortex but diverged in non-primary areas, with higher responses to non-stationary sounds (see Figure S7C for maps in individuals). Because speech and music were excluded from the analysis, this result is not simply a reflection of speech and music selectivity in non-primary auditory cortex. The result is suggestive of a suppression of noise-like sounds in later stages of the auditory hierarchy and may in part explain the general ability of the network model to predict cortical responses to natural sounds.

Relationship between Cortical Predictions and Task Performance

Given that untrained networks predicted voxel responses substantially worse than a task-optimized network (Figure 3B), it is natural to wonder how a network's task performance relates to its ability to predict cortical responses. Previous work found that networks with better performance on a real-world visual object recognition task better predict cortical responses in the ventral visual stream (Yamins et al., 2014). We explored whether a similar relationship might hold in the auditory system.

We examined task performance and cortical prediction quality for 57 different architectures at 14 different time points during task training (yielding a total of 798 different neural networks). Each network was trained for either word recognition or genre classification (with a single, unbranched processing stream), and we measured how well each network performed the task for which it was trained (with stimuli not used for training). Additionally, we measured the median variance explained by each network across all voxels in auditory cortex. For each voxel, we selected each network's best-predicting layer and evaluated that layer's variance explained in left-out data. The performance of a network on a task strongly correlated with the variance it explained in auditory cortical responses (Figure 7A, word task, Spearman $r = 0.87$; Figure 7C for genre task, Spearman $r = 0.85$; both $p < 10^{-100}$; example trajectories of individual architectures over the course of training are shown in Figures 7B and

7D). These findings suggest that task-based optimization of deep neural networks can help yield more predictive models of sensory systems.

DISCUSSION

We developed a quantitative model of auditory computation by optimizing a deep neural network on real-world speech and music tasks. The optimization yielded a model architecture with separate music and speech pathways following a shared front end, potentially replicating one aspect of human cortical organization. The model performed both tasks as well as human listeners, exhibited patterns of behavioral errors like those of humans, and predicted fMRI responses throughout auditory cortex substantially better than a standard cortical model in common use. Task optimization was both necessary and sufficient to achieve the best existing predictions of auditory cortex: necessary in that hierarchical model structure without task optimization yielded poor predictions, and sufficient in that the model was only optimized to perform the tasks and was not otherwise constrained to match human behavior or brain responses. The resulting model provided evidence for hierarchical organization within auditory cortex—intermediate model layers best predicted primary auditory cortex, while deeper layers best predicted non-primary responses. The differentiation between primary and non-primary cortex appears to not simply be a function of pooling information over larger regions of time and/or frequency, in that a network with the same architecture but random weights did not produce the same result. Analyses of the network suggested a hypothesis about cortical processing, which we then tested, finding that non-primary auditory cortex is less responsive to noise-like natural sounds. Despite being trained on speech and music tasks, the key scientific findings from the model were robust to the exclusion of cortical responses to speech and music from the analysis, indicating that the apparent hierarchical structure is somewhat general and is not simply a reflection of neural selectivity for speech and music. Taken together, these results suggest that real-world tasks may substantially constrain both neural processing and behavior, and that task optimization may provide a powerful approach to developing models of neural systems.

A New Model of Auditory Cortex

Our modeling methodology is distinct from traditional approaches rooted in physiological observations or signal processing principles, and the resulting model differs from its predecessors in many respects. Our model is substantially deeper than previous models, with twelve layers of computation following peripheral auditory processing—others have had one or two stages at most (Carlson et al., 2012; Chi et al., 2005; Dau et al., 1997; McDermott and Simoncelli, 2011; Mlynarski and McDermott, 2018). This increase in depth aids the compact expression of successively richer sets of functions of the audio input (Montufar et al., 2014). These rich functions are useful for good real-world task performance, and they also enable improved prediction of cortical responses.

We also introduced a method for learning multiple task-specialized pathways, which produced speech- and music-specific processing streams following several shared stages. This architecture presumably reflects partially overlapping demands of speech and music processing and is consistent with recent evidence for functional segregation in non-primary auditory cortex (Angulo-Perkins et al., 2014; Leaver and Rauschecker, 2010; Norman-Haignere et al., 2015; Tierney et al., 2013). Applying this approach with additional tasks (environmental sound recognition, sound localization, etc.) could help reveal the computational relationship between tasks and generate additional hypotheses about functional segregation and pathway organization in the brain.

Compared to more traditional hand-engineered models (Chi et al., 2005; Dau et al., 1997; McDermott and Simoncelli, 2011), one disadvantage of our model is that individual units are less readily understood. However, we emphasize that the deep neural networks evaluated here have approximately the same number of free parameters for predicting voxel data as do more traditional hand-engineered models, such as the spectrotemporal filter model employed in our work. All the nonlinear neural network parameters were determined by task optimization alone; the only parameters fitted to voxel data were those of the linear map from model units to voxels. Furthermore, given that we evaluated prediction accuracy with left-out stimuli, a larger number of parameters in and of itself will not, in general, lead to better predictions. The deep neural network's training task can be considered a normative constraint—i.e., a hypothesis about the phylogenetic and/or ontogenetic pressures that shape human listeners' behavior and auditory cortical representations.

Deep Neural Networks as a Model of Human Perceptual Judgments

Perhaps the most significant departure from previous auditory models is that our model performs real-world tasks on par with human listeners. Achieving human performance on everyday perceptual tasks was unheard of until just a few years ago but is now attainable in an increasing number of domains due to the efficacy of deep learning. Real-world task performance increases the plausibility of a model, as any complete model of the auditory system should perform the tasks that humans perform. It also enables model evaluation via human model comparisons of behaviors that matter for everyday listening. To our knowledge, our model is the first to provide a detailed match to patterns of human performance across conditions on real-world auditory tasks. We note that the comparison of network performance and human behavior involved zero free parameters—we simply measured the performance of the model and of human listeners in different conditions.

How should we interpret the similarity of performance? One possibility is that both the human and the network are approaching performance limits inherent to the tasks, and thus any model reaching human-level task performance would also exhibit human-like error patterns. Alternatively, the human network similarity could reflect algorithmic similarity, such that there could exist models with human-level performance but with different error patterns. Additional model classes (currently unavailable) will be necessary to disambiguate these alternatives.

Independent of the interpretation, the behavioral similarity between humans and the network lends strength to the goal-driven modeling enterprise.

Improved Cortical Response Predictions

The network predicted cortical responses better than the standard spectrotemporal filter model in both primary and non-primary auditory cortex (Figure 3B). The improved predictions in primary regions suggest that even primary sensory cortex may be better understood by considering how task demands shape representations. Given prior evidence of primary cortical neurons' tuning to spectrotemporal modulations (Santoro et al., 2014; Schönwiesner and Zatorre, 2009), the improved predictions from the network could reflect relatively simple nonlinear functions of spectrotemporal filter responses, such as normalization or pooling. These simple operations are absent from the standard spectrotemporal auditory model but present in our network model.

Evidence for Hierarchical Organization of Auditory Cortex

Investigators have long been intrigued by the idea of hierarchical structure in auditory cortex (Boemio et al., 2005; Okada et al., 2010; Rauschecker and Scott, 2009; Rauschecker et al., 1995; Recanzone and Cohen, 2010). Anatomical data in non-human primates suggest a division into three stages—core, belt, and parabelt (Kaas and Hackett, 2000), which differ in tuning properties (Rauschecker et al., 1995; Recanzone and Cohen, 2010) and response latencies (Camalier et al., 2012). But even in non-human animals the divisions and functions of processing stages remain debated. Moreover, it is not obvious how auditory cortical organization in non-human animals may apply to humans, in part because speech and music are both uniquely human and central to human hearing.

In human auditory cortex, hierarchy is most often considered for speech processing. Speech-selective responses emerge only in non-primary areas (de Heer et al., 2017; Evans and Davis, 2015; Mesgarani et al., 2014; Okada et al., 2010; Overath et al., 2015) and cannot be accounted for by modulation features standardly used to model primary areas (Norman-Haignere et al., 2015; Overath et al., 2015). However, large swaths of non-primary human auditory cortex are not speech selective (Norman-Haignere et al., 2015; Overath et al., 2015), leaving the generality of any potential multi-stage organization an open question. The extent and nature of hierarchical organization in humans has thus remained unsettled (Formisano et al., 2008; Leaver and Rauschecker, 2010; Staeren et al., 2009; Wessinger et al., 2001). Part of the difficulty is that it is not obvious how to assess hierarchy without knowledge of fine-grained connectivity between regions, which is not presently available for human auditory cortex (Cammoun et al., 2015).

We propose an alternative method for evaluating hierarchy, using a hierarchical model to operationalize the “complexity” of neuronal tuning. We compare brain and model responses to the same natural sounds, using the similarity between the two at each stage of the model as a measure of neuronal response complexity. This model-based approach provides an advance over previous, more subjective notions of response complexity (Rauschecker et al., 1995), though it requires a rich experimental

dataset to distinguish response properties at different model stages. Our analyses thus far do not yield compelling evidence that human auditory cortex exhibits the tripartite hierarchical organization commonly proposed in other animals (i.e., core, belt, and parabelt), but such organization may become evident when our methodology is applied to additional neural datasets, or with a more refined model.

Similarities and Differences with the Visual System

Task-optimized artificial neural networks have recently proven to be powerful models of visual cortex, in part because they recapitulate aspects of the hierarchical structure of the ventral visual stream. Because the visual cortical regions and pathways are well established, this similarity was less a novel scientific finding than an effective way of validating the task-driven modeling approach (Cichy et al., 2016; Güçlü and van Gerven, 2015; Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al., 2014). By contrast, less is known about the organization of the auditory cortex, and our model provides novel evidence for hierarchical cortical processing.

One difference between our results and analogous work in the visual system is that primary visual cortical responses have been found to be best predicted by early layers of a deep network (Cadena et al., 2017; Güçlü and van Gerven, 2015; Khaligh-Razavi and Kriegeskorte, 2014), whereas we found primary auditory cortical responses to be best predicted by intermediate network layers. This difference is consistent with the idea that primary auditory cortex may be situated later in a computational hierarchy than primary visual cortex, potentially due to the existence of more subcortical nuclei in the auditory system (King and Nelken, 2009). Further investigating this question, for instance, by predicting subcortical responses with early network layers, is a promising future direction.

Limitations and Future Directions

Although the network better accounts for human behavior and cortical responses compared to previous models, there are many limitations that motivate future work. First, the match to human behavior is imperfect. The genre task exhibited the largest discrepancies, perhaps because it is not critically important for humans and/or may be significantly influenced by cultural experience (it was chosen primarily because it is the only music-related task for which a suitably large database of labeled samples is readily available). Training networks on additional music-related tasks, or tasks not specific to speech or music, could yield a more complete model of human behavior.

Second, the trained network does not explain all of the reliable response variance in the BOLD signal. Some of the remaining variance may reflect the importance of additional tasks, limitations of the regression procedure for mapping between the network model and the brain (Klindt et al., 2017), representations not easily learned in discriminative (e.g., classification-trained) models, or the presence of computations not easily implemented in a feedforward architecture. Finer-grained brain data (e.g., higher-field MRI, electrocorticography, or single-unit physiology) could reveal additional limitations, likely including the absence of realistic temporal dynamics and the lack of a role for behavioral goals or cortical state (e.g., arousal). Such phenomena may

require extending the network architecture to include recurrent dynamics, feedback connections, and/or attention.

Finally, although we propose a model of auditory cortex shaped by task demands, the learning procedure we employ to satisfy those demands likely deviates substantially from biological learning. Humans almost surely do not require millions of labeled examples to learn to recognize words, and instead presumably use some mixture of supervised, unsupervised, and reinforcement learning. The similarity we observe between our model and the human auditory system therefore suggests that systems can arrive at similar final states via different learning algorithms. Future developments in semi- and self-supervised learning will hopefully enable models of human behavior and cortical responses to be learned via more biologically and ecologically realistic procedures.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- **KEY RESOURCES TABLE**
- **CONTACT FOR REAGENT AND RESOURCE SHARING**
- **EXPERIMENTAL MODEL AND SUBJECT DETAILS**
 - Word & genre recognition psychophysics
 - fMRI cortical responses to natural sounds
- **METHOD DETAILS**
 - Tasks for network training
 - Definition of constituent operations of a convolutional neural network (CNN)
 - CNN architectural optimization and filter weight training
 - Psychophysics comparing human listeners and the network
 - fMRI data analysis: Preprocessing and voxel selection
 - Using neural network layers as voxelwise encoding models to predict cortical responses
 - Examining representations in the network
 - Analyzing network responses to more and less stationary sounds
 - Examining relationship between network task performance and auditory cortical variance explained
- **QUANTIFICATION AND STATISTICAL ANALYSIS**
 - Branch point selection
 - Psychophysics statistics
 - ROI analysis statistics
 - Network analysis statistics
 - Significance of individual voxel predictions
- **DATA AND SOFTWARE AVAILABILITY**

SUPPLEMENTAL INFORMATION

Supplemental Information includes seven figures and three tables and can be found with this article online at <https://doi.org/10.1016/j.neuron.2018.03.044>.

ACKNOWLEDGMENTS

The authors thank Nancy Kanwisher for sharing fMRI data, Steve Shannon for MR support, Kevin Woods for analysis of speakers for the representational

similarity analysis, Ariel Herbert-Voss for help collecting behavioral data, Atsushi Takahashi for assistance with MR acquisition protocol design, and Satra Ghosh and the OpenMind team for computing resources. The authors also thank Michael Cohen, Elias Issa, Rebecca Saxe, Pedro Tsividis, and members of the McDermott lab for comments on the manuscript. This work was supported by an NVIDIA Corporation hardware donation, NIH grant NEI-R01 EY014970 to J. DiCarlo (supporting D.L.K.Y.), a DOE Computational Science Graduate Fellowship (DE-FG02-97ER25308) to A.J.E.K., a McDonnell Scholar Award to J.H.M., and NSF grant BCS-1634050 to J.H.M.

AUTHOR CONTRIBUTIONS

Conceptualization, A.J.E.K., D.L.K.Y., and J.H.M.; Methodology, A.J.E.K. and D.L.K.Y.; Investigation, A.J.E.K., E.N.S., and S.V.N.-H.; Software, A.J.E.K. and D.L.K.Y.; Visualization, A.J.E.K. and E.N.S.; Writing – Original Draft, A.J.E.K. and J.H.M.; Writing – Review & Editing, A.J.E.K., D.L.K.Y., E.N.S., S.V.N.-H., and J.H.M.; Supervision and Funding Acquisition, J.H.M.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: May 18, 2017

Revised: December 22, 2017

Accepted: March 23, 2018

Published: April 19, 2018

REFERENCES

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., and Kudlur, M. 2016. TensorFlow: a system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 265–283.
- Angulo-Perkins, A., Aubé, W., Peretz, I., Barrios, F.A., Armony, J.L., and Concha, L. (2014). Music listening engages specific cortical regions within the temporal lobes: Differences between musicians and non-musicians. *Cortex* **59**, 126–137.
- Atencio, C.A., Sharpee, T.O., and Schreiner, C.E. (2012). Receptive field dimensionality increases from the auditory midbrain to cortex. *J. Neurophysiol.* **107**, 2594–2603.
- Bertin-Mahieux, T., Ellis, D.P., Whitman, B., and Lamere, P. (2011). The Million Song Dataset. In International Society for Music Information Retrieval, pp. 591–596.
- Boemo, A., Fromm, S., Braun, A., and Poeppel, D. (2005). Hierarchical and asymmetric temporal sensitivity in human auditory cortices. *Nat. Neurosci.* **8**, 389–395.
- Brainard, D.H. (1997). The psychophysics toolbox. *Spat. Vis.* **10**, 433–436.
- Cadena, S.A., Denfield, G.H., Walker, E.Y., Gatys, L.A., Tolias, A.S., Bethge, M., and Ecker, A.S. (2017). Deep convolutional models improve predictions of macaque V1 responses to natural images. *bioRxiv*.
- Camalier, C.R., D'Angelo, W.R., Sterbing-D'Angelo, S.J., de la Mothe, L.A., and Hackett, T.A. (2012). Neural latencies across auditory cortex of macaque support a dorsal stream supramodal timing advantage in primates. *Proc. Natl. Acad. Sci. USA* **109**, 18168–18173.
- Cammoun, L., Thiran, J.P., Griffa, A., Meuli, R., Hagmann, P., and Clarke, S. (2015). Intrahemispheric cortico-cortical connections of the human auditory cortex. *Brain Struct. Funct.* **220**, 3537–3553.
- Carlson, N.L., Ming, V.L., and Deweese, M.R. (2012). Sparse codes for speech predict spectrotemporal receptive fields in the inferior colliculus. *PLoS Comput. Biol.* **8**, e1002594.
- Chang, E.F., Rieger, J.W., Johnson, K., Berger, M.S., Barbaro, N.M., and Knight, R.T. (2010). Categorical speech representation in human superior temporal gyrus. *Nat. Neurosci.* **13**, 1428–1432.

- Chechik, G., Anderson, M.J., Bar-Yosef, O., Young, E.D., Tishby, N., and Nelken, I. (2006). Reduction of information redundancy in the ascending auditory pathway. *Neuron* 51, 359–368.
- Chi, T., Ru, P., and Shamma, S.A. (2005). Multiresolution spectrotemporal analysis of complex sounds. *J. Acoust. Soc. Am.* 118, 887–906.
- Christianson, G.B., Sahani, M., and Linden, J.F. (2008). The consequences of response nonlinearities for interpretation of spectrotemporal receptive fields. *J. Neurosci.* 28, 446–455.
- Cichy, R.M., Khosla, A., Pantazis, D., Torralba, A., and Oliva, A. (2016). Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Sci. Rep.* 6, 27755.
- Dau, T., Kollmeier, B., and Kohlrausch, A. (1997). Modeling auditory processing of amplitude modulation. I. Detection and masking with narrow-band carriers. *J. Acoust. Soc. Am.* 102, 2892–2905.
- David, S.V., Mesgarani, N., Fritz, J.B., and Shamma, S.A. (2009). Rapid synaptic depression explains nonlinear modulation of spectro-temporal tuning in primary auditory cortex by natural stimuli. *J. Neurosci.* 29, 3374–3386.
- de Heer, W.A., Huth, A.G., Griffiths, T.L., Gallant, J.L., and Theunissen, F.E. (2017). The hierarchical cortical organization of human speech processing. *J. Neurosci.* 37, 6539–6557.
- Depireux, D.A., Simon, J.Z., Klein, D.J., and Shamma, S.A. (2001). Spectrotemporal response field characterization with dynamic ripples in ferret primary auditory cortex. *J. Neurophysiol.* 85, 1220–1234.
- Eickenberg, M., Gramfort, A., Varoquaux, G., and Thirion, B. (2017). Seeing it all: Convolutional network layers map the function of the human visual system. *Neuroimage* 152, 184–194.
- Evans, S., and Davis, M.H. (2015). Hierarchical organization of auditory and motor representations in speech perception: Evidence from searchlight similarity analysis. *Cereb. Cortex* 25, 4772–4788.
- Fischl, B. (2012). FreeSurfer. *Neuroimage* 62, 774–781.
- Formisano, E., De Martino, F., Bonte, M., and Goebel, R. (2008). “Who” is saying “what”? Brain-based decoding of human voice and speech. *Science* 322, 970–973.
- Garofolo, J.S., and Consortium, L.D. (1993). TIMIT: Acoustic-Phonetic Continuous Speech Corpus (Linguistic Data Consortium).
- Glasberg, B.R., and Moore, B.C.J. (1990). Derivation of auditory filter shapes from notched-noise data. *Hear. Res.* 47, 103–138.
- Güçlü, U., and van Gerven, M.A.J. (2015). Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *J. Neurosci.* 35, 10005–10014.
- Hershey, S., Chaudhuri, S., Ellis, D.P.W., Gemmeke, J.F., Jansen, A., Moore, C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B., et al. (2017). CNN architectures for large-scale audio classification. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135.
- Humphries, C., Liebenthal, E., and Binder, J.R. (2010). Tonotopic organization of human auditory cortex. *Neuroimage* 50, 1202–1211.
- Huth, A.G., de Heer, W.A., Griffiths, T.L., Theunissen, F.E., and Gallant, J.L. (2016). Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature* 532, 453–458.
- Jones, E., Oliphant, T.E., and Peterson, P. (2001). SciPy: Open source scientific tools for Python.
- Kaas, J.H., and Hackett, T.A. (2000). Subdivisions of auditory cortex and processing streams in primates. *Proc. Natl. Acad. Sci. USA* 97, 11793–11799.
- Kawahara, H., Morise, M., Takahashi, T., Nisimura, R., Irino, T., and Banno, H. (2008). TANDEM-STRAIGHT: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, F0, and aperiodicity estimation. In *Acoustics, Speech and Signal Processing (IEEE Internal Conference)*, pp. 3933–39355.
- Khaligh-Razavi, S.-M., and Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain IT cortical representation. *PLoS Comput. Biol.* 10, e1003915.
- King, A.J., and Nelken, I. (2009). Unraveling the principles of auditory cortical processing: can we learn from the visual system? *Nat. Neurosci.* 12, 698–701.
- Klindt, D., Ecker, A.S., Euler, T., and Bethge, M. (2017). Neural system identification for large populations separating “what” and “where”. *arXiv*, arXiv:1711.02653, <https://arxiv.org/abs/1711.02653>.
- Leaver, A.M., and Rauschecker, J.P. (2010). Cortical representation of natural complex sounds: effects of acoustic features and auditory object category. *J. Neurosci.* 30, 7604–7612.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521, 436–444.
- Lehky, S.R., and Sejnowski, T.J. (1988). Network model of shape-from-shading: neural function arises from both receptive and projective fields. *Nature* 333, 452–454.
- Liebenthal, E., Binder, J.R., Spitzer, S.M., Possing, E.T., and Medler, D.A. (2005). Neural substrates of phonemic perception. *Cereb. Cortex* 15, 1621–1631.
- Lukoševičius, M., and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* 3, 127–149.
- McDermott, J.H., and Simoncelli, E.P. (2011). Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis. *Neuron* 71, 926–940.
- McDermott, J.H., Schemitsch, M., and Simoncelli, E.P. (2013). Summary statistics in auditory perception. *Nat. Neurosci.* 16, 493–498.
- Mesgarani, N., Cheung, C., Johnson, K., and Chang, E.F. (2014). Phonetic feature encoding in human superior temporal gyrus. *Science* 343, 1006–1010.
- Miller, L.M., Escabí, M.A., Read, H.L., and Schreiner, C.E. (2002). Spectrotemporal receptive fields in the lemniscal auditory thalamus and cortex. *J. Neurophysiol.* 87, 516–527.
- Mlynarski, W., and McDermott, J.H. (2018). Learning midlevel auditory codes from natural sound statistics. *Neural Comput.* 30, 631–669.
- Montufar, G., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Z. Gharamani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, eds. (Curran Associates), pp. 2924–2932.
- Morosan, P., Rademacher, J., Schleicher, A., Amunts, K., Schormann, T., and Zilles, K. (2001). Human primary auditory cortex: Cytoarchitectonic subdivisions and mapping into a spatial reference system. *Neuroimage* 13, 684–701.
- Mustafa, K., and Bruce, I.C. (2006). Robust formant tracking for continuous speech with speaker variability. *IEEE Trans. Audio Speech Lang. Process.* 14, 435–444.
- Naselaris, T., Kay, K.N., Nishimoto, S., and Gallant, J.L. (2011). Encoding and decoding in fMRI. *Neuroimage* 56, 400–410.
- Norman-Haignere, S., Kanwisher, N., and McDermott, J.H. (2013). Cortical pitch regions in humans respond primarily to resolved harmonics and are located in specific tonotopic regions of anterior auditory cortex. *J. Neurosci.* 33, 19451–19469.
- Norman-Haignere, S., Kanwisher, N.G., and McDermott, J.H. (2015). Distinct cortical pathways for music and speech revealed by hypothesis-free voxel decomposition. *Neuron* 88, 1281–1296.
- Obleser, J., Leaver, A.M., Vanmeter, J., and Rauschecker, J.P. (2010). Segregation of vowels and consonants in human auditory cortex: evidence for distributed hierarchical organization. *Front. Psychol.* 1, 232.
- Okada, K., Rong, F., Venezia, J., Matchin, W., Hsieh, I.H., Saberi, K., Serences, J.T., and Hickok, G. (2010). Hierarchical organization of human auditory cortex: evidence from acoustic invariance in the response to intelligible speech. *Cereb. Cortex* 20, 2486–2495.
- Oliphant, T.E. (2006). Guide to NumPy (Brigham Young University).
- Overath, T., McDermott, J.H., Zarate, J.M., and Poeppel, D. (2015). The cortical analysis of speech-specific temporal structure revealed by responses to sound quilts. *Nat. Neurosci.* 18, 903–911.

- Patil, K., Pressnitzer, D., Shamma, S., and Elhilali, M. (2012). Music in our ears: the biological bases of musical timbre perception. *PLoS Comput. Biol.* **8**, e1002759.
- Paul, D.B., and Baker, J.M. (1992). The design for the Wall Street Journal-based CSR corpus. In Proceedings of the Workshop on Speech and Natural Language (Association for Computational Linguistics), pp. 357–362.
- Peelle, J.E., Johnsrude, I.S., and Davis, M.H. (2010). Hierarchical processing for speech in human auditory cortex and beyond. *Front. Hum. Neurosci.* **4**, 51.
- Pinto, N., Doukhan, D., DiCarlo, J.J., and Cox, D.D. (2009). A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput. Biol.* **5**, e1000579.
- Rauschecker, J.P., and Scott, S.K. (2009). Maps and streams in the auditory cortex: nonhuman primates illuminate human speech processing. *Nat. Neurosci.* **12**, 718–724.
- Rauschecker, J.P., Tian, B., and Hauser, M. (1995). Processing of complex sounds in the macaque nonprimary auditory cortex. *Science* **268**, 111–114.
- Razavian, A.S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In IEEE Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 512–519.
- Recanzone, G.H., and Cohen, Y.E. (2010). Serial and parallel processing in the primate auditory cortex revisited. *Behav. Brain Res.* **206**, 1–7.
- Santoro, R., Moerel, M., De Martino, F., Goebel, R., Ugurbil, K., Yacoub, E., and Formisano, E. (2014). Encoding of natural sounds at multiple spectral and temporal resolutions in the human auditory cortex. *PLoS Comput. Biol.* **10**. Published online January 2, 2014. <https://doi.org/10.1371/journal.pcbi.1003412>.
- Schönwiesner, M., and Zatorre, R.J. (2009). Spectro-temporal modulation transfer function of single voxels in the human auditory cortex measured with high-resolution fMRI. *Proc. Natl. Acad. Sci. USA* **106**, 14611–14616.
- Schoppe, O., Harper, N.S., Willmore, B.D.B., King, A.J., and Schnupp, J.W.H. (2016). Measuring the performance of neural models. *Front. Comput. Neurosci.* **10**. Published online February 10, 2016. <https://doi.org/10.3389/fncom.2016.00010>.
- Spearman, C. (1904). The proof and measurement of association between two things. *Am. J. Psychol.* **15**, 72–101.
- Spearman, C. (1910). Correlation calculated from faulty data. *Br. J. Psychol.* **3**, 271–295.
- Staeren, N., Renvall, H., De Martino, F., Goebel, R., and Formisano, E. (2009). Sound categories are represented as distributed patterns in the human auditory cortex. *Curr. Biol.* **19**, 498–502.
- Tierney, A., Dick, F., Deutsch, D., and Sereno, M. (2013). Speech versus song: multiple pitch-sensitive areas revealed by a naturally occurring musical illusion. *Cereb. Cortex* **23**, 249–254.
- Uppenkamp, S., Johnsrude, I.S., Norris, D., Marslen-Wilson, W., and Patterson, R.D. (2006). Locating the initial stages of speech-sound processing in human temporal cortex. *Neuroimage* **31**, 1284–1296.
- Wessinger, C.M., VanMeter, J., Tian, B., Van Lare, J., Pekar, J., and Rauschecker, J.P. (2001). Hierarchical organization of the human auditory cortex revealed by functional magnetic resonance imaging. *J. Cogn. Neurosci.* **13**, 1–7.
- Woods, K.J.P., Siegel, M.H., Traer, J., and McDermott, J.H. (2017). Headphone screening to facilitate web-based auditory experiments. *Atten. Percept. Psychophys.* **79**, 2064–2072.
- Yamins, D.L.K., and DiCarlo, J.J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nat. Neurosci.* **19**, 356–365.
- Yamins, D.L., Hong, H., Cadieu, C.F., Solomon, E.A., Seibert, D., and DiCarlo, J.J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proc. Natl. Acad. Sci. USA* **111**, 8619–8624.
- Zipser, D., and Andersen, R.A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature* **331**, 679–684.
- Zoph, B., Vasudevan, V., Shlens, J., and Le, Q.V. (2017). Learning transferable architectures for scalable image recognition. arxiv, arXiv:1707.07012, <https://arxiv.org/abs/1707.07012>.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and Algorithms		
Numpy	Oliphant, 2006	http://www.numpy.org/
Scipy	Jones et al., 2001	https://www.scipy.org/
TensorFlow	Abadi, et al., 2016	https://www.tensorflow.org/
FreeSurfer	Fischl, 2012	http://www.freesurfer.net
Psychophysics Toolbox	Brainard, 1997	http://psychtoolbox.org/
NSL MATLAB Toolbox	Chi et al., 2005	http://www.isr.umd.edu/Labs/NSL/Software.htm
Other		
Wall Street Journal Speech Corpus	Paul and Baker, 1992	https://catalog.ldc.upenn.edu/ldc93s6a
TIMIT Speech Corpus	Garofolo and Consortium, 1993	https://catalog.ldc.upenn.edu/ldc93s1
Million Song Dataset	Bertin-Mahieux et al., 2011	https://labrosa.ee.columbia.edu/millionsong/

CONTACT FOR REAGENT AND RESOURCE SHARING

Further information and requests for resources should be directed to and will be fulfilled by the Lead Contact, Alex Kell (alexkell@mit.edu).

EXPERIMENTAL MODEL AND SUBJECT DETAILS

Word & genre recognition psychophysics

For the word recognition psychophysics, eighteen subjects participated (12 female, mean age: 23 years, range: 18–33 years). For the genre recognition psychophysics, we performed experiments both in-lab ($n = 31$; 20 female; mean age: 26 years, range: 19–43) and on Amazon's Mechanical Turk ($n = 80$; 26 female; mean age: 34 years, range: 19–67). Mechanical Turk was used to supplement the in-lab data once it became clear that it would be useful to estimate a confusion matrix (which required a relatively large number of participants). All subjects had self-reported normal hearing, and provided informed consent. The Massachusetts Institute of Technology Committee on the Use of Humans as Experimental Subjects approved experiments.

fMRI cortical responses to natural sounds

The fMRI data analyzed here is a subset of the data in [Norman-Haignere et al. \(2015\)](#), only including the subjects who completed three scanning sessions. Eight participants (four female, mean age: 22 years, range: 19–25; all right-handed; one participant was author SNH) completed three scanning sessions (each ~2 hours). Subjects were non-musicians (no formal training in the five years preceding the scan), native English speakers, and had self-reported normal hearing. Two other subjects only completed two scans and were excluded from these analyses, and three additional subjects were excluded due to excessive head motion or inconsistent task performance. The decision to exclude these five subjects was made before analyzing any of their fMRI data. All participants provided informed consent, and the Massachusetts Institute of Technology Committee on the Use of Humans as Experimental Subjects approved experiments.

METHOD DETAILS

Tasks for network training

The training datasets consisted of more than two and a half million labeled exemplars for each task, where each exemplar was a clip of speech or music excerpted from a large, labeled corpus and embedded in background noise ([Figure 1A](#)).

Word task

The word task was a 587-way classification task. We counted the occurrence of all words in the TIMIT ([Garofolo and Consortium, 1993](#)) and WSJ ([Paul and Baker, 1992](#)) corpora, and included all words at least four characters long that were uttered between 25–100 times in the TIMIT corpus or 75–200 times in the WSJ corpus. The lower limit was intended to ensure a sufficient number of examples per word; the upper limit was intended to help create training sets with some degree balance in the number of examples of each word. These criteria yielded 587 unique words (see [Table S1](#) for list of all words). We then excerpted two-second clips from

these corpora in which one of the 587 target words occurred during the halfway point of the clip (i.e., the word overlapped the one-second mark; note that it did not have to be centered at one second and in general was not). We generated more than two and a half million total excerpts. To make the classification task both more realistic and difficult, we superimposed these speech excerpts on one of three different kinds of background noise: (1) “auditory scenes,” (2) music, or (3) two-speaker speech “babble.” Auditory scenes were real-world recordings of everyday acoustic environments (e.g., a bus station, an office, a restaurant, a supermarket), and were drawn from the corpus used for the 2013 IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events. Music exemplars were drawn from an in-house corpus of monophonic and polyphonic instrumental music recordings (solo guitar, orchestral ensemble, big band, movie soundtracks, etc.). Speech babble was generated by mixing together a pair of speakers randomly drawn from a large corpus of public domain audiobook recordings (<https://librivox.org>). These background clips were added to the speech excerpts at signal-to-noise ratios selected to roughly match recognition difficulty across background type for human listeners (based on pilot psychophysical experiments). To increase heterogeneity within the training stimuli, the exact signal-to-noise ratio (SNR) for each exemplar was drawn randomly from a Gaussian with a standard deviation of 2 dB SNR and a mean of -3 dB (for auditory scenes and speech babble) or -6 dB (for music).

Genre task

The genre task was a 41-way classification task. The Million Song Dataset (Bertin-Mahieux et al., 2011) consists of nearly a million tracks of various types. More than 300,000 of these tracks have user-generated “tags” from the MusicBrainz open-source music encyclopedia (<https://musicbrainz.org>). Tags are at the artist level, not the track level (e.g., all of Marvin Gaye’s tracks have the same tags), and these 300,000 tracks are by $\sim 9,000$ unique artists. There were 2,321 unique tags for these artists, but many of them occurred with low frequency. We first culled all tags that were linked to at least ten different artists, yielding 277 tags, and then screened out tags that did not obviously correspond to a genre (e.g., “American,” “London,” “Male Vocalist,” “Ninja Tune”), leaving a total of 73 tags. More than 180,000 tracks by more than 4,400 different artists had one of these 73 tags. Many of these tags were synonymous (e.g., “hip hop,” “hip-hop,” and “hiphop”) and others could plausibly be considered of a similar genre (e.g., “heavy metal,” “thrash metal,” “black metal”). To group these tags into genre classes, we computed a co-occurrence matrix of how often an artist was shared between two tags, normalized this 73×73 matrix by the base rates of each tag (so each element was the proportion of overlap), and hierarchically clustered this normalized co-occurrence matrix, grouping together tags that overlapped substantially. This procedure yielded 41 genre clusters (see Table S2 for list of genres and the tags associated with each genre).

To generate training exemplars for the genre task, we randomly excerpted approximately two and a half million two-second clips from these 180,000 tracks. As with the word task, to make the classification task more realistic and difficult, we embedded these two-second excerpts in one of four different background noises: (1) auditory scenes, (2) two-speaker speech babble, (3) eight-speaker speech babble, or (4) music-shaped noise. Auditory scenes and two-speaker babble were generated in the same way as they were for the word recognition task. Eight-speaker babble was generated analogously to the two-speaker babble, but included eight distinct speakers. Music-shaped noise consisted of a two-second clip of noise that was matched to the average spectrum of its corresponding two-second clip of music. SNRs for background noise in the genre task were also selected based on pilot behavioral results in humans to yield roughly equal performance across noise types. The genre training clips were presented at substantially higher SNR relative to the word recognition task due to the difficulty of genre recognition for human listeners. The mean SNR for each of the four background types was 12 dB, and the exact SNR for each training example was drawn randomly from a Gaussian with a standard deviation of 2 dB. For both the genre and the word task, all waveforms were downsampled to 16 kHz.

Stimulus preprocessing for networks: Cochleograms

The input to the network was a cochleagram of each training exemplar. A cochleagram is a time-frequency decomposition of a sound that mimics aspects of cochlear processing – it is similar to a spectrogram, but with a frequency resolution like that thought to be present in the cochlea, and with a compressive nonlinearity applied to the amplitude in each time-frequency bin.

Cochleograms were generated similarly to those in previous work (McDermott et al., 2013; McDermott and Simoncelli, 2011). Each two-second waveform was passed through a bank of 203 bandpass filters. Filters were zero-phase with frequency response equal to the positive portion of a single period of a cosine function. The center frequencies ranged from 30 Hz to 7860 Hz. The filters were evenly spaced on an Equivalent Rectangular Bandwidth (ERB)_N scale, approximately replicating the frequency-dependence of bandwidths believed to characterize the human cochlea (Glasberg and Moore, 1990). Filters were designed to perfectly tile the spectrum – the summed squared response across all frequencies was flat – and to achieve this tiling, four low-pass and four high-pass filters were included. Adjacent filters overlapped in frequency by 87.5%. With the lowpass and highpass filters on the ends of the spectrum, there were in total 211 filters.

The envelope of each filter subband was computed as the magnitude of the analytic signal (via the Hilbert transform). To simulate basilar membrane compression, these envelopes were raised to the power of 0.3. Compressed envelopes were downsampled to 200 Hz, yielding a cochleagram representation that was 211 by 400 (frequency \times time). Finally, these cochleograms were resampled (with an antialiasing filter) to 256×256 to input to the networks. Cochleagram generation was done in Python, making heavy use of the numpy and scipy libraries (Jones et al., 2001; Oliphant, 2006).

Definition of constituent operations of a convolutional neural network (CNN)

The 256×256 cochleograms were passed into convolutional neural networks (CNNs), which are a feedforward cascade of linear and nonlinear operations – the output of each layer is passed as the input to the subsequent layer. In our CNNs there were four different kinds of layers: (1) a *convolutional layer*, (2) a *normalization layer*, (3) a *pooling layer*, and (4) a *fully connected layer*. Below we define the operations of each type of layer.

Convolutional layer

Each *convolutional layer* consisted of a bank of linear filters and a pointwise nonlinearity. The input to a *convolutional layer* is a three-dimensional array of shape $(n_{in}, n_{in}, d_{channels_in})$. Where n_{in} is the spectral and temporal dimensionality of the input to that layer. In the case of the first layer, n_{in} is the number of time or frequency bins of the preprocessed cochleogram (i.e., 256). $d_{channels_in}$ is the number of channels from the previous layer. In the case of the first convolutional layer, $d_{channels_in} = 1$, because the cochleogram representation had a single value for each time-frequency bin.

The *convolutional layer* is defined by the following five parameters:

1. k_s : The size of the convolutional kernels
2. n_k : The number of different kernels
3. I_s : The length of the stride of the convolution
4. K : The kernel weights for each of the n_k kernels; this is an array of dimensions $(k_s, k_s, d_{channels_in}, n_k)$.
5. b : The bias vector, of length n_k

For any input array X of shape $(n_{in}, n_{in}, d_{channels_in})$, the output of a *convolutional layer* is an array Y of shape $(n_{in}/I_s, n_{in}/I_s, n_k)$:

$$Y(i, j, k) = \text{relu}\left(b[k] + \frac{1}{k_s^2} \sum K[:, :, :, k] \odot N_{ks}(X, I_s \cdot i, I_s \cdot j)\right),$$

where i and j range in $(1, \dots, n_{in}/I_s)$, \odot denotes the pointwise array multiplication, $N_{ks}(X, q, r)$ selects the square neighborhood across adjacent time and frequency bins of size k_s by k_s , centered at location q, r in X , returning an array of shape $(k_s, k_s, d_{channels_in})$. The sum is over all elements of this 3d array. The convolution is done with “same” mode, meaning that the edges are padded with zeros to produce an output that would be the same dimensionality of the input if the stride were set to 1. Finally, *relu* denotes the rectified linear operator, which is a pointwise nonlinearity applied to every element of the output:

$$\text{relu}(x) = \max(0, x).$$

Normalization layer

A *normalization layer* implements divisive normalization of a unit by its neighboring filters at the identical time-frequency bin. It is defined by three parameters: α , β , and $n_{adjacent}$, which had values of 0.001, 0.75, 5, respectively. It operates on an array of X of shape $(n_{in}, n_{in}, d_{channels_in})$ and returns an array Y of the identical shape:

$$Y(i, j, k) = \frac{X[i, j, k]}{\left(1 + \alpha \sum_{l \in A} X[i, j, l]^2\right)^{\beta}},$$

where A is the set of the neighboring input channels whose responses are included in the normalization procedure and k indexes filter kernels. A is defined as $[k - (n_{adjacent} - 1)/2, k - (n_{adjacent} - 1)/2 + 1, \dots, k, \dots, k + (n_{adjacent} - 1)/2 - 1, k + (n_{adjacent} - 1)/2]$, which in our case of $n_{adjacent} = 5$ results in $A = [k-2, k-1, k, k+1, k+2]$. This procedure thus normalizes a filter’s in a time-frequency bin by other filter responses for different filters at that same time-frequency bin. We handled boundaries by zero-padding.

Pooling layer

A *pooling layer* downsamples its input by aggregating values across nearby time and frequency bins:

1. p_s , the size of the pooling kernel
2. p_o , the pooling order (in our case: 1, 2, or infinity)
3. I_s , the length of the stride

A *pooling layer* operates on an array X of shape $(n_{in}, n_{in}, d_{channels_in})$ and returns an array Y of shape of $(n_{in}/I_s, n_{in}/I_s, d_{channels_in})$, via the following function:

$$Y(i, j, k) = \left(\frac{1}{p_s^2} \sum N_{ps}(X^{p_o}, I_s \cdot i, I_s \cdot j)[:, :, k]\right)^{\frac{1}{p_o}},$$

where N_{ps} is the local square neighborhood in time and frequency. The sum is over all elements in the resulting (p_s, p_s) matrix. $p_o = 1$ yields pooling via the mean, $p_o = 2$ yields pooling via the root mean square (“L2 pooling”), and $p_o = \infty$ yields max pooling.

Fully connected layer

Unlike the previously described layers, a *fully connected layer* does not have a notion of localized frequency or time. It takes an input vector X and returns a vector Y of length of (n_{out}):

$$Y(i) = \text{relu}\left(\frac{1}{n_T} \sum_{j=1}^{n_T} w_{ij} X[j]\right),$$

where j iterates the number of input elements in the preceding layer, w_{ij} is a weight (a real-valued number), and the point-wise nonlinearity (*relu*) is as defined before. If the preceding layer is another *fully connected layer* the input is a vector of length (n_T). If the preceding layer is a *convolutional layer*, a *normalization layer* or a *pooling layer*, then the input is an array A of shape ($n_{in}, n_{in}, d_{channels_in}$) unwrapped into a single vector X of length $n_T = n_{in}^2 \times d_{channels_in}$.

Dropout during training

During training, we instantiated a dropout layer after each fully connected layer that was not the final classification layer. During each batch of training data, a randomly drawn fifty percent of the fully connected layer connections were eliminated (i.e., their connection weight was set to zero). Dropout is common in neural network training and can be seen as a form of model averaging. For evaluation, we followed the conventional procedure by removing this dropout layer and multiplying the output of each fully connected unit by the probability of dropout during training (i.e., 0.5).

Softmax classifier

The final layer in the network is a *fully connected layer* where n_{out} is the number of target classes (587 or 41, respectively in the case of the word or genre task) and the *relu* operator is replaced with a *softmax*, an operation that receives a vector as its input and whose element-wise output is defined as:

$$y(i) = \frac{\exp\left(\sum_{j=1}^{n_T} w_{ij} X_j\right)}{\sum_{k=1}^{n_{out}} \exp\left(\sum_{l=1}^{n_T} w_{kl} X_l\right)}.$$

Each element of the output of the softmax is nonnegative and together they sum to one, and can thus be interpreted as a probability distribution over the classes (either words or genres).

Convolving in frequency

The convolutional neural networks we used in this paper convolved filter kernels with the cochleagram in both time and frequency. Applying a given kernel in the first layer yields an $n_{spectral}$ by $n_{temporal}$ feature map – i.e., outputs values at $n_{temporal}$ different time bins for each of $n_{spectral}$ different frequencies. Convolving in time is natural for a model of the auditory system because sound waveforms are naturally ordered in a temporal sequence. The naturalness of convolving in frequency may be less obvious. However, convolution in frequency can be conceptualized as having $n_{spectral}$ different model units, each centered at a different center frequency, where each of the $n_{spectral}$ units is constrained to have the same filter weights. Instantiating the same filter at different frequencies might be reasonable given that sounds are to some extent translation invariant in frequency, and is present in standard models of spectrotemporal filtering (Chi et al., 2005). Furthermore, we empirically found in pilot experiments that task performance was better when convolution was applied in frequency as well as time, likely because this substantially reduces the number of parameters to be learned and thus may have acted as an useful form of regularization.

CNN architectural optimization and filter weight training

Filter weight training

During training, the filter weights in each *convolutional layer* and each *fully connected layer* were adjusted to improve classification performance via stochastic gradient descent (SGD). Training was performed on 5.1 million sounds and performance was monitored on a left-out validation set of 400,000 sounds. We used the cross-entropy loss function and a batch size of 256 stimuli. Error on this loss function was backpropagated to update the weights in each *convolutional layer* and *fully connected layer* to decrease the loss function.

Network architecture selection: Overview

We selected the architecture for our network via a two-stage procedure. First, we defined a broad set of architectural hyperparameters and searched for architectures that performed well on either the word or genre task separately (Figure 1B). Then we searched across ways of merging the best single-task architectures into a single network to perform both tasks (Figure 1C). We divided the architecture search into these two stages for practical reasons – simultaneously searching over both base architecture and branch point would have been prohibitively computationally expensive.

Network architecture selection: Family of potential architectures

To optimize the model architecture, we defined a space of potential architectures by creating a distribution over architectural hyperparameters. This hyperparameter space was defined as follows:

- The first six stages of processing consisted of the following layers in the following order: *convolution*, *normalization*, *pooling*, *convolution*, *normalization*, *pooling*. Although this order was identical across all architectures we considered, we varied some of the architectural hyperparameters that define each of these stages, as we describe below.

- After these first six stages, there were a variable number of additional *convolutional layers* (between 0 and 3)
- Following the above, there were one or two stages of *fully connected layers*. During training, a *dropout layer* was added after each of these layers.
- And the final stage was a *fully connected layer* with either 587 or 41 units, whose outputs were passed through a softmax and interpreted as a probability distribution over classes (words or genres)

Additionally, the number of filters in each of these potential *convolutional layers* were fixed (in order of *convolutional layer* number): 96, 256, 512, 1024, and 512. The number of fully connected units (other than the output layer) was set to 4096. The pooling window size was set to three, and normalization was performed over neighborhoods of five filters. To simplify our search over architectures, we also fixed the convolutional filters to be “square,” with equal spectral and temporal extent.

The choice of this template architecture was made based on pilot experiments and experience with networks trained on visual object recognition tasks. Within this template architecture, there were a number of degrees of freedom that we searched over:

- Total number of convolutional layers: [2, 3, 4, 5]
- Number of fully connected layers before the softmax layer: [1, 2]
- Convolutional filter kernel sizes: [3, 5, 7, 9, 11, 13]
- Stride of each convolutional filter: [1, 2, 3, 4]
- The pooling order: [1, 2, ∞] (i.e., average, root mean square, or max)

Network architecture selection, first stage: Single-task networks

To search over these architectural hyperparameters parameters, we set a uniform probability distribution over each of these choices, which acted as our prior over architectures. Then we drew 100 sample architectures from this prior and randomly assigned each architecture to be trained on either the word or genre task. We optimized the *convolutional layer* and *fully connected layer* filter weights with SGD for 28 epochs (complete passes over the training data). We then evaluated the performance of each architecture on the task on which it was trained using left-out validation data. We used tree-structured Parzen estimation to update our probability distribution over each hyperparameter, assigning more probability mass to those hyperparameter values that produced better task performance. Specifically, to update the probability distribution for a given architectural hyperparameter (e.g., convolutional filter size), we first separated the 100 original architectures into two groups: the 50 that yielded above median performance and the 50 that yielded below median performance. We then fit one truncated Gaussian ($q(x)$) to the values of that given architectural hyperparameter in the above median group, and separately fit another truncated Gaussian ($r(x)$) to those hyperparameters for the below median performance group. To bias selection of hyperparameter values toward those that yielded higher performance, we updated the distribution over each architecture hyperparameter as the ratio of those two truncated Gaussians (i.e., $q(x) / r(x)$).

We then drew an additional 40 architectures from this new distribution over hyperparameters, initialized the weights for two networks for each of these 40 architectures (one for the word and one for the genre task), and optimized the filter weights in each one of these 80 new networks for their respective task. We trained all 180 networks for a total of 42 epochs (the original 100 and these latter 80). We then evaluated the validation set performance for each of these 180 architectures on the task that it was trained. We found that the same architecture performed best on both the word and the genre task.

It is unclear the extent to which an explicit hyperparameter optimization procedure, as used here, is more useful than random search over architectures. Moreover, there are obvious limitations to the optimization procedure that we used. For instance, our optimization procedure updates the probability distribution over each hyperparameter separately, and thus assumes that performance as a function of architectural hyperparameter is separable across hyperparameters, which is not obviously the case. Note, however, that the architecture that performed best on each task was drawn from the hyperparameter distribution after the update to the prior, potentially suggesting some benefit from the optimization procedure.

Network architecture selection, second stage: Merging optimal architectures by selecting the branch point

We sought a single architecture to perform both word recognition and genre classification, which could share some number of early layers before splitting into two branches, one for word recognition and one for genre classification. Finding this shared network architecture was simplified by the fact that the same architecture produced the best performance for each task separately, such that the architectural decision was reduced to selecting the location of the “branch point” (Figure 1C).

Although there were twelve layers of processing for the network, only seven of them were convolutional or fully connected, and thus had weights that would be optimized during SGD. The other five layers were normalization or pooling layers, and the operations of those layers were not altered by task training. Therefore, normalization and pooling layer operations remained the same regardless of whether they were shared between tasks or split into separate branches, and thus it was nonsensical to consider branch points at these layers. We considered branch points ranging from before the first layer (i.e., yielding two fully separate networks), to just prior to the softmax classification layers (i.e., a branch point after layer fc6). This network with a branch point after fc6 had nearly half as many parameters as the entirely separate networks, because of the shared processing. To minimize the total number of network parameters while maximizing performance, we sought the latest branch point for which performance was not significantly lower than performance for the fully separate networks, with significance levels determined via a bootstrap over both classes (i.e., words or genres) and stimuli.

We generated networks with all seven possible branch points (from totally separate networks to branching after layer fc6), and optimized the filter weights from scratch in each of these seven networks for both the word and genre tasks. We found that performance on the validation set was not significantly affected by sharing up to three convolutional layers of processing (Figure 1D).

Network architecture selection: Final CNN architecture

The final architecture consisted of 12 layers of processing that split into separate streams after the conv3 layer, culminating in word classification or genre classification softmax layers (Figure 1E). The layers in the two streams are denoted by _W or _G. The architectural hyperparameters of parallel layers in the two streams are identical except for the final classification layers (which differ in dimensionality due to the different number of classes for each task).

- Input (256x256): Cochleagram: 256 frequency bins x 256 time bins
- conv1 (85x85x96): Convolution of 96 kernels with a kernel size of 9 and a stride of 3
- mnrm1 (85x85x96): Response normalization over 5 adjacent kernels
- pool1 (42x42x96): Max pooling over window size of 3x3 and a stride of 2
- conv2 (22x22x256): Convolution of 256 kernels with a kernel size of 5 and a stride of 2
- mnrm2 (22x22x256): Response normalization over 5 adjacent kernels
- pool2 (11x11x256): Max pooling over a window size of 3x3 and a stride of 2
- conv3 (13x13x512): Convolution of 512 kernels with a kernel size of 3 and a stride of 1
- conv4_W & conv4_G (15x15x1024): Convolution of 1024 kernels with a kernel size of 3 and a stride of 1
- conv5_W & conv5_G (17x17x512): Convolution of 512 kernels with a kernel size of 3 and a stride of 1
- pool3_W & pool3_G (8x8x512): Mean pooling over a window size of 3 and a stride of 2
- fc1_W & fc1_G (4096): A fully connected layer
- fctop_W & fctop_G (587 or 41): A fully connected layer whose outputs are passed through a softmax function and interpreted as a probability distribution over either words ($n = 587$) or genres ($n = 41$).

During training, there was a dropout layer after fc1_W and fc1_G.

Although most of our analyses are based on this final network, we also report voxel prediction results for the two totally separate networks (i.e., that shared no layers), to examine the effects of training on only one of the two tasks (Figure S3E).

Control model: A spectrotemporal modulation filter bank

To compare the neural network to an existing model of auditory cortex, we also performed analyses with a standard spectrotemporal filter model (Chi et al., 2005). The model consists of a bank of linear filters tuned to spectrotemporal modulations at different acoustic frequencies, spectral scales, and temporal rates (see Figure 1F for example filters). We used the NSL MATLAB Toolbox (<http://www.isr.umd.edu/Labs/NSL/Software.htm>) implementation of the spectrotemporal model, with 63 audio frequencies (ranging between 20 Hz and 8 kHz), 17 temporal rates (logarithmically spaced between 0.5 Hz and 128 Hz), and 13 spectral scales (logarithmically spaced between 0.125 and 8 cycles/octave), yielding a spectrotemporal modulation filter bank with 29,736 filters – 1071 temporal modulation filters (17 rates by 63 frequencies), 819 spectral modulation filters (13 rates by 63 frequencies), and 27,846 spectrotemporal filters (17 temporal rates by 13 spectral rates by 63 frequencies, by two orientations corresponding to upward and downward frequency modulations). The filters were applied to a cochleagram. The cochleograms that were used as input to the CNN were resampled in frequency to match the log spacing of the spectrotemporal model. To evaluate the model response we passed a sound through the model, extracted the magnitude (absolute value) of each filter's response at each time step and averaged these magnitudes over time, to get 29,736 measures for each sound. To ensure that the comparison with the spectrotemporal filter model was fair, we verified that this parameterization of the spectrotemporal filter model saturated the amount of variance that a spectrotemporal filter model could explain in the voxel data (see Figure S2C, and methods below for more details).

Psychophysics comparing human listeners and the network

As an initial test of the plausibility of the trained neural network as a model of human auditory cortex, we compared the performance characteristics of the model with that of human listeners on the two tasks we used to train the network.

Human psychophysics: Word recognition

We measured word recognition performance in twenty-six different conditions – five different background types at five different SNRs along with a clean condition without any added background noise. The five background types were: (1) music, (2) two-speaker speech babble, (3) eight-speaker speech babble, (4) speaker-shaped noise, and (5) auditory scenes. The five SNRs were: -9, -6, -3, 0, and +3 dB SNR. Speaker-shaped noise (included to maximize “energetic” masking of the target speech by the background) was generated for each clip by estimating the average amplitude spectrum for the clip's speaker from that speaker's exemplars in the corpus, and synthesizing a noise sample with the same spectrum (by replacing the Fourier amplitudes of a white noise signal with the speaker's average amplitude spectrum). All other background types were generated with the same procedure used for the network training stimuli.

Each subject completed a two-hour in-lab experimental session. Sounds were played via the sound card on a MacMini at a sampling rate of 16 kHz, via a Behringer HA400 amplifier. The Psychtoolbox for MATLAB (Brainard, 1997) was used to present the stimuli. Subjects heard the sounds via Sennheiser HD280 headphones (circumaural) in a soundproof booth (Industrial Acoustics). All stimuli were presented at 70 dB SPL.

Each trial consisted of a two-second clip generated according to the network training data generation procedure described above. Participants were instructed to report the word that occurred during the middle of the clip (i.e., during the one-second mark). The network had a closed-set recognition task (587 possible words), and human subjects performed an analogous 587 alternative forced choice (AFC) task. To facilitate performance with such a large number of classes, we familiarized participants with the 587 words before their behavioral session by allowing them to look over the list, and we programmed an interface that only allowed responses from the 587-word dictionary. Participants typed responses but could enter in a part of a word, hit the spacebar key, and see all words in the dictionary that had the typed characters. Trials were grouped into runs of 78 trials between which subjects could take a break. Subjects performed up to seven runs. Across our 18 subjects, the average number of trials per subject was 451 (range: 312–546).

Human psychophysics: Genre classification

For the genre recognition task, we measured human genre classification performance in twenty-one different conditions – four background types at five different SNRs and a “clean” (background-less) condition. The four background types were: (1) auditory scenes, (2) clip-shaped noise, (3) two-speaker babble, and (4) eight-speaker babble. All backgrounds were generated as they were for the network training stimuli. We measured performance at the following SNRs: −9, −6, −3, 0, +3 dB.

Listeners heard a two-second clip randomly excerpted from a track from the Million Song Dataset embedded in background noise. Subjects had to report the genre they thought the clip belonged to. Because of the difficulty of the task (due to the brief stimulus and overlap between genres), subjects performed a “top 5” task in which they selected the five genres the clip was most likely to belong to, in order of confidence (the most likely genre first, then the second-most likely genre, and so on). All forty-one genres were presented in a numbered list on the screen during the response period and participants entered the five numbers corresponding to their five top choices. To familiarize participants with the genres, we played the in-lab subjects three examples per genre just prior to the experimental session. For the Turk subjects, we sought to insure their attention by having 41 familiarization trials (one for each genre) before the experimental session. Turk subjects heard three examples for each genre in succession and performed a two-way AFC on the genre, deciding between the true genre and an alternative (selected by hand to be substantially different; e.g., for “hip hop” the distractor was “opera”). Feedback was provided during this familiarization phase only, not during experimental trials.

In lab, stimuli were presented at 70 dB SPL. It was impractical to control the level for our Turk subjects. To help ensure that Turk subjects were wearing headphones, we required that the Turk subjects pass a headphone check task previously demonstrated to screen out listeners not wearing headphones or earphones (Woods et al., 2017). Turk subjects listened to three pure tones in a row and had to report which tone was quietest. Two of the tones were in phase across the two stereo channels and a third was in anti-phase. One of the in-phase tones was 5 dB less intense than the other two. However, if subjects were listening over speakers rather than headphones the antiphase tone would be expected to be quietest due to in-air phase cancellation. All Turk subjects performed six of these loudness discrimination trials, and subjects who failed to get five of six trials correct were excluded from participating in the Turk experiment. We ran 300 subjects on Turk, but 146 (49%) failed the headphone check. Of the remaining 154 subjects, we considered data only from those who completed a minimum of 84 trials (i.e., four trials per background noise type and SNR pair), leaving a total of 80 Turk subjects.

Figure 2A plots the mean performance of all subjects for each condition in the word recognition task. Figure 2G plots mean performance of all subjects for each condition in the genre recognition task (a trial was counted as correct if the correct genre was included in any of the five guesses from the subject).

Neural network psychophysics: Word and genre classification

To evaluate model behavior on these two psychophysical tasks, we presented the same set of stimuli to the model that we presented to human listeners, extracting the model unit responses from the relevant top layer (FCTop_W for the word stimuli and FCTop_G for the genre stimuli). For each word stimulus, we took the argmax across the 587 model unit responses and recorded whether the unit with the highest response corresponded to the target word or not (Figure 2B; scatter comparing human and network word performance in Figure 2C). For genre stimuli, we noted which units had the five highest responses (5 highest probabilities of a genre conditioned on the input). If the correct genre was included in the five largest softmax values, then that trial was counted as correct when computing performance for Figures 2H and 2I.

Word psychophysics control analysis: Cochleagram distortion analyses

To better understand the similarity in the pattern of word recognition performance between the humans and the network across conditions, we explored whether performance could be explained by the extent to which different background noises distort the speech signal. We did not perform distortion analyses for the genre task because performance did not vary substantially across background types, and was thus determined primarily by SNR as measured in the waveform domain. We measured distortion for each stimulus by generating cochleograms of each target speech signal with and without the assigned background noise, and then computed the absolute difference between corresponding time-frequency bins in the two cochleograms (Figure 2D). We took the mean of this absolute difference over both time and frequency, aggregating this distortion metric by taking its mean across all stimuli for each background condition. Using the root mean square difference instead of the mean absolute difference yielded nearly identical results

(Pearson's r between MAE and RMS is 0.98, $p < 10^{-17}$). Figure 2E shows this measure of mean absolute distortion and Figure 2F shows the scatter of this distortion measure versus human behavior for each condition.

A limitation of the above distortion analysis is that it measures distortion in all time-frequency bins, but it is possible that distortion that overlaps temporally and spectrally with the speech signal may be particularly detrimental to word recognition abilities. To control for this possibility, we conducted a modified distortion analysis, where for each stimulus we only measured distortion in cochleagram bins that had speech signal within a certain number of decibels (dB) from peak signal power (Figure S1A). We varied our selection criterion from 50 dB down to 10 dB down in increments of 10 dB.

Word psychophysics control analysis: Network response distortion analyses

Given that cochleagram distortion did not predict human performance particularly well, we examined whether there would be a closer relationship with human performance for an analogous measure of distortion computed in different layers of the trained network model. We presented target speech signals with and without background noise to the network and recorded the model unit responses in each of the layers. We then computed the mean absolute difference between these unit responses for each stimulus and each layer, aggregating over background type. Figure S1B shows scatters of human performance versus distortion for each layer of the network.

Genre psychophysics comparisons: Confusion matrices and control models

Given that performance across background types did not vary substantially for the genre task, we instead compared error patterns for both the network and the humans. This comparison was feasible for the genre task because there were only 41 classes, yielding a total of just under 1700 bins (41×41). We did not perform this analysis for the word task because it would have required orders of magnitude more behavioral data – the full confusion matrix would have more than 340,000 bins (587×587).

We generated two 41-by-41 confusion matrices, one for the human behavioral data and one for the network (Figures 2J and 2K, respectively). Each column of the matrix contained the responses for a genre. The elements of each column contained the proportion of trials on which a genre tag was in the top five guesses for the genre. Each column was z-scored to control for response biases (which might be expected to be different for the network and human listeners). To compare confusion matrices, we unwrapped each into a vector and measured the Spearman correlation between the resulting vectors.

fMRI data analysis: Preprocessing and voxel selection

Natural sound stimuli

The stimuli were a set of 165 two-second sounds selected to span the sorts of sounds that listeners most frequently encounter in day-to-day life (Norman-Haignere et al., 2015). All sounds were recognizable – i.e., classified correctly at least 80% of the time in a ten-way alternative forced choice task run on Amazon Mechanical Turk, with 55–60 participants per sound. Turk participants also assigned each of these 165 sounds to one of eleven categories (instrumental music, music with vocals, English speech, foreign speech, non-speech vocal sound, animal vocalization, human non-vocal sound, animal non-vocal sound, nature sound, mechanical sound, or environmental sound; 30–33 participants per sound). Category assignments were highly reliable (split-half kappa of 0.93). In analyses that we describe later in this methods section where we excluded speech and music stimuli from the fMRI dataset, we relied upon these judgments of third-party listeners to determine whether a stimulus contained speech or music (i.e., we excluded the stimuli the Turk subjects classified as instrumental music, music with vocals, English speech, or foreign speech). See Table S3 for names of all stimuli and category assignments. To download all 165 sounds, see the McDermott lab website: <http://mcdermottlab.mit.edu/downloads.html>.

Sounds were presented using a block design. Each block included five presentations of the identical two-second sound clip. After each two-second sound, a single fMRI volume was collected (“sparse scanning”), such that sounds were not presented simultaneously with the scanner noise. Each acquisition lasted one second and stimuli were presented during a 2.4 s interval (200 ms of silence before and after each sound to minimize forward/backward masking by scanner noise). Each block lasted 17 s (five repetitions of a 3.4 s TR). This design was selected based on pilot results showing that it gave more reliable responses than an event-related design given the same amount of overall scan time. Blocks were grouped into eleven runs, each with fifteen stimulus blocks and four blocks of silence. Silence blocks were the same duration as the stimulus blocks and were spaced randomly throughout the run. Silence blocks were included to enable estimation of the baseline response.

To encourage subjects to attend equally to each sound, subjects performed a sound intensity discrimination task. In each block, one of the five sounds was 7 dB lower than the other four (the quieter sound was never the first sound). Subjects were instructed to press a button when they heard the quieter sound. Sounds were presented through MR-compatible earphones (Sensimetrics S14) at 75 dB SPL (68 dB SPL for the quieter sounds).

fMRI data acquisition

MR data were collected on a 3T Siemens Trio scanner with a 32-channel head coil at the Athinoula A. Martinos Imaging Center of the McGovern Institute for Brain Research at MIT. Each functional volume consisted of fifteen slices oriented parallel to the superior temporal plane, covering the portion of the temporal lobe superior to and including the superior temporal sulcus. Repetition time (TR) was 3.4 s (although acquisition time was only 1 s), echo time (TE) was 30 ms, and flip angle was 90 degrees. For each run, the five initial volumes were discarded to allow homogenization of the magnetic field. In-plane resolution was 2.1×2.1 mm (96 \times 96 matrix), and slice thickness was 4 mm with a 10% gap, yielding a voxel size of $2.1 \times 2.1 \times 4.4$ mm. iPAT was used to minimize acquisition time. T1-weighted anatomical images were collected in each subject (1mm isotropic voxels) for alignment and surface reconstruction.

fMRI data preprocessing

Functional volumes were preprocessed using FSL and in-house MATLAB scripts. Volumes were corrected for motion and slice time. Volumes were skull-stripped, and voxel time courses were linearly detrended. Each run was aligned to the anatomical volume using FLIRT and BBRegister. These preprocessed functional volumes were then resampled to vertices on the reconstructed cortical surface computed via FreeSurfer, and were smoothed on the surface with a 3mm FWHM 2D Gaussian kernel to improve SNR. All analyses were done in this surface space, but for ease of discussion we refer to vertices as “voxels” throughout this paper. For each of the three scan sessions, we estimated the mean response of each voxel (in the surface space) to each stimulus block by averaging the response of the second through the fifth acquisitions after the onset of each block (the first acquisition was excluded to account for the hemodynamic lag). Pilot analyses showed similar response estimates from a more traditional GLM. These signal-averaged responses were converted to percent signal change (PSC) by subtracting and dividing by each voxel’s response to the blocks of silence. These PSC values were then downsampled from the surface space to a 2mm isotropic grid on the FreeSurfer-flattened cortical sheet. For summary maps, we registered each subject’s surface to Freesurfer’s fsaverage template.

Voxel selection

For individual subject analyses, we used the same voxel selection criterion as [Norman-Haignere et al. \(2015\)](#), selecting voxels with a consistent response to sounds from a large anatomical constraint region encompassing the superior temporal and posterior parietal cortex. Specifically, we used two criteria: (1) a significant response to sounds compared with silence ($p < 0.001$); and (2) a reliable response to the pattern of 165 sounds across scans. The reliability measure was as follows:

$$r = 1 - \frac{\|\mathbf{v}_{12} - \text{proj}_{\mathbf{v}_3}\mathbf{v}_{12}\|_2}{\|\mathbf{v}_{12}\|_2},$$

$$\text{proj}_{\mathbf{v}_3}\mathbf{v}_{12} = \left(\frac{\mathbf{v}_3^T}{\|\mathbf{v}_3\|_2} \mathbf{v}_{12} \right) \mathbf{v}_3$$

where \mathbf{v}_{12} is the response of a single voxel to the 165 sounds averaged across the first two scans (a vector), and \mathbf{v}_3 is that same voxel’s response measured in the third. The numerator in the second term in the first equation is the magnitude of the residual left in \mathbf{v}_{12} after projecting out the response shared with \mathbf{v}_3 . This “residual magnitude” is divided by its maximum possible value (the magnitude of \mathbf{v}_{12}). The measure is bounded between 0 and 1, but differs from a correlation in assigning high values to voxels with a consistent response to the sound set, even if the response does not vary substantially across sounds. We found that using a more traditional correlation-based reliability measure excluded many voxels in primary auditory cortex because some of them exhibit only modest response variation across natural sounds. We included voxels with a value of this modified reliability measure of 0.3 or higher, which when combined with the sound responsive t test yielded a total of 7694 voxels across the eight subjects (mean number of voxels per subject: 961.75; range: 637-1221).

For summary maps, which aggregated across individuals, voxels were included if at least four subjects had a reliability of at least 0.3.

Region of interest (ROI) selection: Overview

We localized four regions of interest (ROIs) in each participant, consisting of voxels selective for (1) frequency (i.e., tonotopy), (2) pitch, (3) speech, and (4) music. In each case we ran a “localizer” statistical test and selected the top 5% most significant individual voxels in each subject and hemisphere (including all voxels identified by the sound-responsive and reliability criteria described above). We excluded voxels that were identified in this way by more than one localizer (see [Figure S3A](#) for amount of overlap between ROIs before this exclusion criterion was applied). Key results were robust to varying the ROI selection criterion to 2% or 10% ([Figures S3B](#) and [S3C](#)). The frequency, pitch, and speech localizers required acquiring additional imaging data, and were collected either during extra time during the natural sound stimuli scan sessions or on additional sessions on different days. Scanning acquisition parameters were identical to those used to acquire the natural sounds data. Throughout this paper we refer to voxels chosen by these criteria as “selective,” for ease and consistency. Heatmaps displaying ROI voxel counts across subject are at the bottom of [Figure 3B](#). Red indicates at least one subject had a voxel and yellow indicates two or more, except for the “all” ROI in which red indicates one subject and yellow indicates four or more.

ROI selection: Frequency-selective voxels

To identify frequency-selective voxels, we measured responses to pure tones in six different frequency ranges (center frequencies: 200, 400, 800, 1600, 3200, 6400 Hz) ([Humphries et al., 2010](#); [Norman-Haignere et al., 2013](#)). For each voxel, we ran a one-way ANOVA on its response to each of these six frequency ranges and selected voxels that were significantly modulated by pure tones (top 5% of all selected voxels in each subject). Although there was no spatial contiguity constraint built into our selection method, in practice most selected voxels were contiguous and centered around Heschl’s gyrus ([Figure 3B](#)).

ROI selection: Pitch-selective voxels

To identify pitch-selective voxels, we measured responses to harmonic tones and spectrally-matched noise ([Norman-Haignere et al., 2013](#)). For each voxel we ran a one-tailed t test evaluating the response to tones was greater than that to noise. We selected the top 5% of individual voxels in each subject that had the lowest p values for this contrast.

ROI selection: Speech-selective voxels

To identify speech-selective voxels, we measured responses to German speech and to temporally scrambled (“quilted”) speech stimuli generated from the same German source recordings (Overath et al., 2015). We used foreign speech to identify responses to speech acoustical structure, independent of linguistic structure. Note that two of the subjects had studied German in school and for one of these subjects we used Russian utterances instead of German. The other subject was tested with German because the Russian stimuli were not available at the time of the scan. For each voxel we ran a one-tailed t test evaluating whether responses were higher to intact speech than to statistically matched quilts. We selected the top 5% of all selected voxels in each subject.

ROI selection: Music-selective voxels

To identify music-selective voxels, we used the music component derived by Norman-Haignere et al. (2015). We inferred the “voxel weights” for each voxel to all six of the components from its response to the 165 sounds:

$$\mathbf{w} = \mathbf{C}'\mathbf{v},$$

where \mathbf{w} contains the inferred voxel weights (a vector of length 6), \mathbf{C}' is the Moore-Penrose pseudoinverse of the “response components” (a 6 by 165 matrix), and \mathbf{v} is the measured response of a given voxel (vector of length 165). We assessed the significance of each voxel’s music component weight via a permutation test. During each iteration, we shuffled all the component elements, recomputed this new matrix’s pseudoinverse, and re-computed each voxel’s weights via the matrix multiply above. We performed this procedure 10,000 times, and fit a Gaussian to each voxel’s null distribution of music weights. We then calculated the likelihood of the empirically observed voxel weight from this null distribution, and took the top 5% of voxels with the lowest likelihood under this null distribution.

Using neural network layers as voxelwise encoding models to predict cortical responses

Feature extraction from CNN

We first generated cochleograms for each of the 165 sounds from the fMRI experiment and passed them through the CNN, recording each model unit’s response in each layer to each sound (schematic in Figure 3A). We performed this procedure for all sounds and layers.

Because the neural network’s input (a cochleogram) had a temporal dimension, the extracted responses from all layers (except for the fully connected layers) had a temporal dimension. This temporal component was critical to the computation of the features throughout the layers of the deep network, as temporal information at each layer is integrated by model filters at the succeeding layer to compute a signal that depends on both time and frequency. However, the hemodynamic signal to which we were comparing the model blurs the temporal variation of the cortical response, thus a fair comparison of the model to the fMRI data involved predicting each voxel’s time-averaged response to each sound from time-averaged model responses. We therefore averaged the model responses over the temporal dimension after extraction. As a result of the *relu* and *softmax* operators, all responses in all layers were nonnegative, such that averaging preserved the mean response amplitude. For each layer that had a temporal dimension (i.e., each *convolutional layer*, *normalization layer*, and *pooling layer*), we first extracted a three-dimensional array of responses of shape ($n_{spectral}$, $n_{temporal}$, $n_{kernels}$). We then reshaped this 3d array into a matrix where each row corresponded to a kernel at a given frequency and each column corresponded to a time point. We finally took the average of each row over columns, yielding for each sound a vector whose length was the product of $n_{spectral}$ and $n_{kernels}$.

As a result of this procedure, each layer produced the following number of regressors for each sound: conv1 (8160), rnorm1 (8160), pool1 (4032), conv2 (5632), rnorm2 (5632), pool2 (2816), conv3 (6656), conv4_W (15360), conv4_G (15360), conv5_W (8704), conv5_G (8704), pool5_W (4096), and pool5_G (4096). Each *fully connected* layer did not have a temporal dimension, and so we simply extracted each model unit’s response yielding the following number of features: fc1_W (4096), fc1_G (4096), fctop_W (587), and fctop_G (41). We used each of these 17 feature sets to predict the fMRI responses to the natural sound stimuli.

Voxelwise modeling: Regularized linear regression and cross validation

We modeled each voxel’s time-averaged response as a linear combination of a layer’s time-averaged unit responses. We first generated 10 randomly selected train/test splits of the 165 sound stimuli into 83 training sounds and 82 testing sounds. For each split, we estimated a linear map from model units to voxels on the 83 training stimuli and evaluated the quality of the prediction using the remaining 82 testing sounds (described below in greater detail). For each voxel-layer pair, we took the median across the 10 splits.

The linear map was estimated using regularized linear regression. Given that the number of regressors (i.e., time-averaged model units) typically exceeded the number of sounds used for estimation (83), regularization was critical. We used L2-regularized (“ridge”) regression, which can be seen as placing a zero-mean Gaussian prior on the regression coefficients. Introducing the L2-penalty on the weights results in a closed-form solution to the regression problem, which is similar to the ordinary least-squares regression normal equation:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + n\lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y},$$

where \mathbf{w} is a d-length column vector (the number of regressors – i.e., the number of time-averaged units for the given layer), \mathbf{y} is an n-length column vector containing the voxel’s mean response to each sound (length 83), \mathbf{X} is a matrix of regressors (n stimuli by d regressors), n is the number of stimuli used for estimation (83), and \mathbf{I} is the identity matrix (d by d). We demeaned each column of the

regressor matrix (i.e., each model unit's response to each sound), but we did not normalize the columns to have unit norm. By not constraining the norm of each column to be one, we implemented ridge regression with a non-isotropic prior on each unit's learned coefficient. Under such a prior, units with larger norm were expected *a priori* to contribute more to the voxel predictions. In pilot experiments, we found that this procedure led to more accurate and stable predictions in left-out data, compared with a procedure where the columns of the regressor matrices were normalized (i.e., with an isotropic prior).

Performing ridge regression requires selecting a regularization parameter (denoted above by λ) that trades off between the fit to the (training) data and the penalty for weights with high coefficients. To select this regularization parameter, we used leave-one-out cross validation within the set of 83 training sounds. Specifically, for each of 50 logarithmically-spaced regularization parameter values ($10^0, 10^{-1}, \dots, 10^{-49}, 10^{-49}$), we measured the squared error in the resulting prediction of the left out sound using regression weights derived from the other sounds in the training split. We computed the average of this error (across the 83 training sounds) for each of the 50 potential regularization parameter values. We then selected the regularization parameter that minimized this mean squared error. Finally, with the regularization parameter selected, we used all 83 training sounds to estimate a single linear mapping from a layer's features to a given voxel's response. We then used this linear mapping to predict the response to the left-out 82 test sounds, and evaluated the Pearson correlation of the predicted voxel response with the observed voxel response. We squared this correlation coefficient to yield a measure of variance explained. We found that the selected regularization parameter values never fell on the boundaries of the search grid, suggesting that the range of the search grid was appropriate. We emphasize that the 82 test sounds on which predictions were ultimately evaluated were not incorporated into the procedure for selecting the regularization parameter nor for estimating the linear mapping from layer features to a voxel's response – i.e., the procedure was fully cross-validated.

Selecting regularization coefficients independently for each voxel-layer regression was computationally expensive, but seemed important for our scientific goals given that the optimal regularization parameter could vary across voxel-layer pairs. For instance, differences in the extent to which the singular value spectrum of the feature matrix is uniform or peaked (which influences the extent to which the $\mathbf{X}^T \mathbf{X} + n\mathbf{I}$ matrix in the normal equation above is well-conditioned) can lead to differences in the optimal amount of regularization. Measurement noise, which varies across voxels (as seen in the variation in test-retest reliability across voxels in [Figure S2A](#)) can also influence the degree of optimal regularization. By allowing different feature sets (layers) to have different regularization parameters we are enabling each feature set to make the best possible predictions, which is appealing given that the prediction quality is the critical dependent variable that we compare across voxels and layers. Varying the regularization parameter across feature sets while predicting the same voxel response will alter the statistics of the regression coefficients across feature sets, and thus would complicate the analysis and interpretation of regression coefficients. However, we are not analyzing the regression coefficients in this work.

Voxelwise modeling: Correcting for reliability of the measured voxel response

The use of explained variance as a metric for model evaluation is inevitably limited by measurement noise. To correct for the effects of measurement noise we computed the reliability of both the measured voxel response and the predicted voxel response. Correcting for the reliability of the measured response is important to make comparisons across different voxels, because, as seen in [Figure S2A](#), the reliability of the BOLD response varies across voxels. This variation can occur for a variety of reasons (e.g., distance from the head coil elements). Not correcting for the reliability of the measured response will downwardly bias the estimates of variance explained and will do so differentially across voxels. This differential downward bias could lead to incorrect inferences about how well a given set of model features explains the response of voxels in different parts of auditory cortex.

Our data were relatively reliable for an fMRI experiment: the median test-rest reliability across all included voxels (i.e., correlation coefficient between pairs of single scans) was 0.33; the median estimated test-retest reliability of the average response across all three scans (estimated via the Spearman-Brown correction) was 0.59 (top of [Figure S2A](#)). Moreover, our estimates of the reliability were themselves reliable (bottom of [Figure S2A](#)): the correlation of voxel reliability measured in different pairs of scans was $r = 0.88$. Individual voxel predictions were nearly all significant, as well, even under the relatively stringent Bonferroni correction for multiple comparisons ([Figure S2B](#)).

Voxelwise modeling: Correcting for reliability of the predicted voxel response

Measurement noise corrupts the test data to which model predictions are compared (which we accounted for by correcting for the reliability of the measured voxel response, as described above), but noise is also present in the training data and thus also inevitably corrupts the estimates of the regression weights mapping from model features to a given voxel. This second influence of measurement noise is often overlooked, but can be addressed by correcting for the reliability of the predicted response. Doing so is important for two reasons. First, as with the reliability of the measured voxel response, not correcting for the predicted voxel response can yield incorrect inferences about how well a model explains different voxels. Second, the reliability of the predicted response for a given voxel can vary across feature sets, and failing to account for these differences can lead to incorrect inferences about which set of features best explains that voxel's response.

Differences in the reliability of the predicted response across layers are due in part to differences in the singular value spectra of features from the different layers. A flatter, more uniform distribution of singular values will lead to more reliable predictions, whereas a more peaked distribution will lead to less reliable predictions. More peaked singular value spectra will inflate the contribution of noise to the regression weights due to the matrix inversion in the regularized least-squares normal equation above. Adding regularization “flattens” the singular value spectrum and thus reduces the contribution of noise.

It was thus in practice important to correct for the reliability of the predicted voxel response. By correcting for both the reliability of the measured voxel response and the reliability of the predicted response, the ceiling of our measured r-squared values was 1 for all voxels and all layers, enabling comparisons of voxel predictions across all voxels and all neural network layers.

Voxelwise modeling: Corrected measure of variance explained

To correct for the reliability, we employ the correction for attenuation (Spearman, 1904). It is a standard technique in many fields, and is becoming more common in neural data analysis. The correction estimates the correlation between two variables independent of measurement noise (here the measured voxel response and the model prediction of that response). The result is an unbiased estimator of the correlation coefficient that would be observed from noiseless data.

Our corrected measure of variance explained was the following:

$$r_{\text{v.v}}^{2*} = \frac{r(\mathbf{v}_{123}, \hat{\mathbf{v}}_{123})^2}{r'_v r'_v},$$

where \mathbf{v}_{123} is the voxel response to the 82 left-out sounds averaged over the three scans, $\hat{\mathbf{v}}_{123}$ is the predicted response to the 82 left-out sounds (with regression weights learned from the other 83 sounds), r is a function that computes the correlation coefficient, r'_v is the estimated reliability of that voxel's response to the 83 sounds and r'_v is the estimated reliability of that predicted voxel's response. r'_v is the median of the correlation between all 3 pairs of scans (scan 0 with scan 1; scan 1 with scan 2; and scan 0 with scan 2), which is then Spearman-Brown corrected to account for the increased reliability that would be expected from tripling the amount of data (Spearman, 1910). Figure S2A shows the histograms of the pairwise correlation and the Spearman-Brown corrected correlation (i.e., the estimate of the reliability of the average data across three scans). r'_v is analogously computed by taking the median of the correlations for all pairs of predicted responses and Spearman-Brown correcting this measure. Note that for very noisy voxels, this division by the estimated reliability can be unstable and can cause for corrected variance explained measures that exceed one. To ameliorate this problem, we limited both the reliability of the prediction and the reliability of the voxel response to be greater than some value k (Huth et al., 2016). For $k = 1$, the denominator would be constrained to always equal one and thus the “corrected” variance explained measured would be identical to uncorrected value. For $k = 0$, the corrected estimated variance explained measure is unaffected by the value k . This k -correction can be seen through the lens of a bias-variance tradeoff: this correction reduces the amount of variance in the estimate of variance explained across different splits of stimuli, but does it at the expense of a downward bias of those variance explained metrics (by inflating the reliability measure for unreliable voxels). We used a k of 0.128, which is the $p < 0.05$ significance threshold for the correlation of two 165-dimensional Gaussian variables (i.e., with the same length as our 165-dimensional voxel response vectors).

Voxelwise modeling: Summary

We repeated this procedure for each layer and voxel ten times, once each for 10 random train/test splits, and took the median explained variance across the ten splits for a given layer-voxel pair. We performed this procedure for all 17 layers and all 7694 selected individual voxels. For comparison, we performed an identical procedure with the layers of a random-filter network with the same base architecture as our main network (Figures 3B, 4A, 4D, 4F, S2D, and S3B), single-task networks (Figure S2E), and for the time-averaged magnitudes for a bank of spectrotemporal filters (Figures 3B, 4A, 4B, S2C, and S3B–S3E). Additionally, we performed the regression for 78 randomly chosen random-filter networks with architectures that were not selected, and we summarized these predictions in ROIs by taking the median across all networks (Figures 3B and S3B).

When computing mean r^2 values throughout this paper, we averaged the values after Fisher transforming the correlation values. That is, we took the square root of each of the r^2 values we sought to average, performed the Fisher r-to-z transform, took the average across these values, performed the inverse of the Fisher z-to-r transform, and then squared the result. Averaging z-transformed values is appealing because the sampling distribution of correlation coefficients is skewed, and averaging in z-space reduces the bias of the estimate of the true mean.

All regression and analysis code was written in Python, making heavy use of the numpy and scipy libraries (Jones et al., 2001; Oliphant, 2006).

Varying the parameterization of the spectrotemporal filter bank

For voxelwise predictions, our baseline model was a spectrotemporal filter bank (Chi et al., 2005). To give as generous of comparison as possible, we sought to maximize the ability of the spectrotemporal model to explain cortical responses. We generated thirty-six different parameterizations of the spectrotemporal model; each with 63 different acoustic frequencies, and we varied across six different sets of spectral scales and six different temporal rates, all of which were logarithmically spaced.

Temporal rates: 3 rates (0.5, 4, 32 Hz), 5 rates (0.5, 2, 8, 32, 128 Hz), 9 rates (0.5, 1, 2, 4, 8, 16, 32, 64, 128 Hz), 13 rates (0.5, 0.79, 1.26, 2, 3.17, 5.04, 8, 12.70, 20.16, 32, 50.80, 80.63, 128 Hz), 17 rates (0.5, 0.71, 1, 1.41, 2, 2.83, 4, 5.66, 8, 11.31, 16, 22.63, 32, 45.25, 64, 90.51, 128 Hz), 23 rates (0.5, 0.66, 0.87, 1.15, 1.52, 2, 2.64, 3.48, 4.59, 6.06, 8, 10.56, 13.93, 18.38, 24.25, 32, 42.22, 55.72, 73.52, 97.01, 128 Hz).

Spectral scales: 2 scales (0.25, 2 cycles/octave), 4 scales (0.125, 0.5, 2, 8 cycles/octave), 7 scales (0.125, 0.25, 0.5, 1, 2, 4, 8 cycles/octave), 10 scales (0.125, 0.20, 0.32, 0.5, 0.79, 1.26, 2, 3.17, 5.04, 8 cycles/octave), 13 scales (0.125, 0.18, 0.25, 0.35, 0.5, 0.71, 1.0, 1.41, 2.0, 2.83, 4.0, 5.66, 8 cycles/octaves), 16 scales (0.125, 0.16, 0.22, 0.29, 0.38, 0.5, 0.66, 0.87, 1.15, 1.52, 2, 2.64, 3.48, 4.59, 6.06, 8 cycles/octave).

We examined the voxel prediction abilities of models with all 36 conjunctions (Figure S2C). We computed the variance explained for each of the resulting 36 different spectrotemporal models for each voxel from each subject, computed the median for each subject, and took the mean across subjects. These parameterizations varied in the number of parameters, from around 1,000 to over 40,000, and we examined the relationship between number of features in the spectrotemporal filter model and the cortical variance explained. We found that the variance explained by the model reached an asymptote by 30,000 features and thus for our control model we used the parameterization with 17 temporal rates, 13 spectral scales, and 63 audio frequencies.

Examining the effect of the random seed for the random-filter network

Throughout the paper we compare the trained network to a single untrained random-filter network. To examine the effect of the particular filter weights generated from a given randomization seed, we examined the predictions of each layer from ten other random-filter networks (i.e., with ten different seeds), as seen in Figure S2D.

ROI analyses

To estimate the voxel response variance explained by the network within functionally-defined regions of interests (ROIs), we selected the most predictive layer for the ROI in each individual subject using a leave-one-subject-out procedure. For a given subject and a given ROI, we computed the median variance explained by each network layer across the voxels in each of the other seven subject's ROI. We then took the average of this value across those subjects, yielding the mean variance explained by each layer for that ROI from the other subjects. We selected the layer that was most predictive in these seven subjects and measured the variance explained in the left-out subject. This cross validation across subjects avoided issues of non-independence. We iterated over subjects and ROIs and report the mean across subjects in Figures 3B and S3B.

To examine the variance explained by each layer across all of auditory cortex (Figure 4A), we took the median variance explained by each layer of the trained network for each subject across all selected voxels for that subject and then computed the mean of this measure across subjects. We performed the identical procedure for the random-filter networks and the spectrotemporal filter model. Similarly, to examine the median variance explained by each layer within the voxels in each ROI (Figures 4F, S3C, and S3D), we took the median variance explained by each layer over voxels in each subject's ROI and computed the mean. Black lines in Figure 4F shows analogous plots for the random-filter CNN. Additionally, to examine the effects of optimizing a network for either task alone, we performed an identical procedure with networks trained either for word or genre recognition alone (i.e., the baseline model in Figure 1D, with a branch point before any processing), and plot the results in Figure S3E.

Summary maps

For summary maps, we predicted responses in individuals and then aggregated results across subjects (with either the mean or median), after they were aligned in a common coordinate system (i.e., the fsaverage surface from FreeSurfer).

Summary maps: Variance explained by best-predicting layer

To explore how well the network predicted responses across all of the brain that we measured, we examined the variance explained by the best-predicting layer for each voxel, without employing an inclusion criterion. For the summary map of the variance explained by the best layer in the network (Figure S4A), we first computed the r^2 value for each session separately for each subject, correcting for the reliability of the voxel's response and of the layer's prediction of that response as estimated from the three pairs of individual scans. To measure the variance explained by the best-predicting layer of the network for each voxel, we compute the average variance explained across two scans, selected which layer had the highest r^2 for this pair of scans, and measured the r^2 for that layer in the left-out scan. We took the median of this left-out r^2 value across all three scans, and then took the mean over subjects. For comparison, we performed the same procedure on the "raw" variance explained measures (i.e., uncorrected for reliability, but Spearman-Brown corrected to account for using one-third the amount of data that we used for other maps); the result is shown in Figure S4B.

Summary maps: Difference between intermediate and higher layer

To compare the variance explained by different layers of the network, we selected an intermediate and higher layer and computed the difference in the r^2 value for each voxel for each individual, subtracting the value of the intermediate layer from that for the higher layer, and then averaging this value across subjects. Figure 4C shows this difference map for the conv5 and conv3 layers. Three example individual subjects (sub0, sub2, sub5) can be seen on the right of Figure 4C; Figure S5B shows all eight. Figure S5A shows the group-summary maps for other pairs of layers; the general form of the map is consistent across the particular layers used. For conv5 and other layers located after the branch point (that thus had separate layers for the genre and word streams), we took the mean of the r^2 values for the layers in the two streams. Figure 4D shows analogous maps for the same layers of the random-filter network. Figure 4E shows an analogous plot for the trained neural network but for the predictions of the voxel responses when all speech and music stimuli were excluded from the fMRI dataset.

Summary maps: Best-predicting layer

We also examined which layer best predicted each layer's response (an "argmax" analysis), which we summarized by taking the median across subjects for each voxel (Figure 4B). Individual subject maps are show in Figure S4C.

Examining representations in the network

To probe the representations of different layers of the network, we conducted the series of analyses shown in Figures 5, 6, S6, and S7. Although the networks used to generate Figures 1, 2, 3, 4, and 7 were trained using in-house software, these network representational analyses were conducted on a network retrained in TensorFlow (Abadi et al., 2016). TensorFlow, which became available only during the late stages of the project, facilitated these analyses and was anticipated to facilitate dissemination/distribution of

the network. This retrained network was similar to the in-house trained network, but had 1024 units in fc6 rather than 4096 (it was also trained from different random initial conditions). This TensorFlow network is the one that we analyze for [Figures 5, 6, S6, and S7](#), and that will be posted on the McDermott lab website (<http://mcdermottlab.mit.edu/>).

Layerwise predictions of cochlear and spectrotemporal filters

We tested the ability of units in different network layers to predict responses to simulated cochlear filters and spectrotemporal modulation filters. The cochlear filter responses that we predicted were the time-average of each of the cochleagram channels ($n = 256$), which approximate the power spectrum of a sound; the spectrotemporal filters were taken from the same parameterization of the spectrotemporal filter model that we used elsewhere in paper ($n = 29,736$). We used the same regularized regression procedure that we used for the voxel predictions, modeling each cochlear or spectrotemporal filter's response to each of the 165 natural sounds as a linear combination of network units in a given layer ([Figures 5A and 5B](#)). We performed the identical procedure with the units from the untrained, random-filter network.

Layerwise performance on word, genre, and speaker tasks: Training stimulus sets

To examine the task relevance of the features in different network layers, we examined the ability of each layer to support performance of the word and genre task that we trained the network to perform. For training stimuli, we used more than 1.5 million examples for each task (i.e., a subset of those that were used for network optimization).

To examine the generality of the network's learned representations, we tested the ability of features in different network layers to perform a speaker identification task on which the network was not trained. We trained and evaluated classifiers used a subset of the speech stimuli that were used as training stimuli for the word task. Because we wanted a large number of examples per speaker, we used the subset from the WSJ corpus where each of 199 speakers had at least seventy-five hundred examples in our training set, leaving us a dataset of more than two million clips.

Layerwise performance on word, genre, and speaker tasks: Classifier optimization

We trained linear (softmax) classifiers for each layer on each of three tasks: word, genre, and speaker classification. We fixed all of the weights in the network and optimized only a softmax classifier that took a given layer's output as its as input. We used a cross-entropy loss function and updated weights with stochastic gradient descent and a batch size of 256. Because optimization hyperparameters can affect classification performance, we tested two learning rates: 10^{-4} and 10^{-5} . These two learning rates were selected based on pilot experiments. The one exception was for layer fc6 of the untrained network, for which a learning rate of 10^{-3} was used (lower learning rates produced much worse performance). For each layer of the trained network and the untrained, random-filter network, as well as for the spectrotemporal filter model and the cochlear model, we trained a classifier with both learning rates for twelve epochs (full passes over the training set). For each feature set, we selected the classifier that had the lowest loss on held-out validation stimuli. If the classifier overfit during the training procedure, we employed early stopping and used the classifier weights from the epoch at which the classifier minimized the loss on the validation stimuli during the training run.

We then evaluated each of these classifiers on a separate, unseen test set. For the word and genre tasks, we used the same stimuli on which we measured human psychophysical performance for the graphs of [Figure 2](#). For the speaker task, we used the subset of the word psychophysical stimuli that included speakers on which the classifier was trained. The layerwise performance for the word, genre, and speaker tasks is shown in [Figures 5C–5E](#), respectively. An additional analysis showing the effect of background noise on the word classifiers is shown in [Figures 6A and S7A](#).

Layerwise representational similarity analysis: Word, genre, and speaker

To supplement the layerwise performance analyses, we employed representational similarity analysis to further explore the representations learned by different network layers ([Figure S6](#)). We selected 18 words, randomly selected 50 examples per word, and computed the correlation matrix for each layer's response to each word example (the "representational similarity matrix"). Separately, we computed the Levenshtein edit distance between each pair of words as a measure of word similarity, and then compared each layer's similarity matrix with the resulting word similarity matrix by correlating the two for each layer. We performed the identical procedure with the untrained, random-filter network, the spectrotemporal features, and the cochleagram.

We performed analogous procedures for the genre and speaker tasks. For the genre task, we used all 41 genres (40 stimuli per genre), and estimated intrinsic genre similarity as the proportion of artists shared between genres (based on the human-annotated tags). For the speaker task, we used 20 examples for each of 16 speakers. To estimate similarity between speakers we measured the mean and standard deviation over time of the F0 for each speaker (using STRAIGHT; [Kawahara et al. 2008](#)), and the mean frequency of the first formant during "schwa" utterances (using the Mustafa-Bruce formant tracker; [Mustafa and Bruce, 2006](#)). We z-scored each of these statistics across speakers and then computed the (Euclidean) distance between speakers in this three-dimensional space, as shown in the speaker distance matrix.

Analyzing network responses to more and less stationary sounds

To test whether responses to stationary sounds might be attenuated in later network layers, we examined the response of all layers to more and less stationary stimuli. We excluded speech and music stimuli from this stationarity analysis to help ensure that any effect we observed was not due to speech or music-selectivity that might be present in later layers of the network or in non-primary auditory cortical regions.

Measuring the stationarity of natural sounds

Stationarity was operationalized as the variability of sound statistics measured in short time windows. For each sound, we computed the cochleagram, divided it into temporal bins, and measured the following statistics in each of these bins: (i) the mean of each frequency channel (capturing the spectrum); (ii) the correlation across different frequency channels; and (iii) the power in a set of temporal modulation filters applied to the envelopes. To capture stationarity across different timescales, we varied the bin size: 50, 100, 200, and 400 ms. As a summary measure of stationarity we computed the standard deviation of each of these statistics over time, averaging across statistics and bin width to get a single measure of temporal variability for each sound. A schematic of this procedure can be found in [Figure 6B](#).

Using this measure of stationarity, we selected a set of more noise-like sounds ($n = 38$; top third according to the stationarity measure) and a set of less noise-like sounds ($n = 38$; bottom third) from the 113 natural sounds (of the original set of 165) that were neither speech nor music.

Measuring response of network units to more and less stationary stimuli

We measured each unit's response in each layer to each of these 76 sounds (38 more stationary, 38 less stationary). For each unit in a given layer, we took the ratio of the mean response to the less stationary sounds over the mean response to the more stationary sounds. We took the median across all units in a given layer, excluding “dead” units (i.e., units that did not respond at all to either sound set). For the main text figure we took the median over both branches for branched layers; for the supplemental figure we took the median over each branch separately. We performed the identical procedure with the untrained network. The results are shown in [Figures 6C](#) and [S7B](#).

Measuring voxel responses to more and less stationary stimuli

Analogously to our analysis of network units, we measured the mean response of each voxel to the 38 less stationary sounds and the 38 more stationary sounds, and took the ratio (less stationary over more stationary). In [Figure 6D](#) we show the map of the median of this value over subjects for each voxel. In [Figure S7C](#), we show individual maps.

Examining relationship between network task performance and auditory cortical variance explained

To examine the relationship between how well a network performed the task on which it was trained and how well it predicted auditory cortical responses, we examined task performance and voxelwise variance explained in a subset of the networks that were generated in the process of our first step of architectural hyperparameter search ([Figure 7](#)). We examined a random subset of networks rather than all networks because of practical constraints due to the amount of compute time and disk space required to perform the following analysis. We examined fifty-seven different architectures at fourteen different time points during task training (for a total of 798 different networks). Each network was trained for either word recognition or genre classification (respectively, 392 and 406 networks), and we measured how well each network performed the task for which it was trained using approximately 25,000 left-out validation stimuli. On each of these 798 networks, we trained a softmax classifier to convergence (using SGD) while holding the rest of the network parameters constant.

For each of these 798 networks we measured the median voxelwise variance explained, across all selected individual-subject voxels in auditory cortex. For a given voxel and a given network layer, we performed a similar split-half (across sounds) cross validation scheme that we describe above, except that we predicted responses for each scan separately. We determined which layer best predicted each voxel by averaging the variance explained across the predictions of each layer for two sessions, taking the argmax across layers, and then noting the variance explained by that layer in the left-out third session. We repeated this procedure three times, leaving each session out. We took the median of the explained variance for the three left-out splits. For each network, we iterated this procedure over all 7694 selected voxels, and summarized how well that network predicted auditory cortical responses by taking the median across these 7694 voxelwise variance explained measures. We iterated this procedure over all 798 neural networks.

[Figure 7A](#) shows the results across the 392 word-trained networks. [Figure 7C](#) shows the results for the 406 genre-trained networks. [Figures 7B](#) and [7D](#) show the same data, but highlight the trajectories of individual architectures over training time – each subplot highlights the 14 network checkpoints over training for a given architecture.

QUANTIFICATION AND STATISTICAL ANALYSIS

Branch point selection

The branch point for the multitask network was selected with the goal of sharing as many layers as possible without seeing a significant decrement in task performance. Significance was determined by bootstrapping mean performance on the validation data over resamples of the classes (i.e., words or genres) and stimuli ([Figure 1D](#)).

Psychophysics statistics

For the plots of human psychophysical performance on the word and genre tasks ([Figures 2A](#) and [2G](#); also in [Figures 2C](#), [2F](#), and [2I](#)), error bars are within-subject SEM. For corresponding plots of network psychophysical performance ([Figures 2B](#) and [2H](#); also in [Figures 2C](#) and [2I](#)), error bars are SEM, bootstrapped over stimuli and classes (either words or genres). For distortion plots ([Figures 2E](#), [2F](#), [S1A](#), and [S1B](#)), error bars are bootstrapped over stimuli. To examine the similarity between pairs of genre confusion matrices

([Figures 2J and 2K](#)), we computed the Spearman correlation coefficient between unwrapped matrices, and evaluated the significance via NHST p values.

ROI analysis statistics

For all ROI (region of interest) analyses ([Figures 3B, 4A, 4F, and S3B–S3E](#)), differences were evaluated with either paired t tests or ANOVAs (specified in main text).

Network analysis statistics

For regressions to cochlear and spectrotemporal filter responses ([Figures 5A and 5B](#)), error bars are SEM, bootstrapped over filters and train-test splits of sounds. For layerwise classifiers ([Figures 5C–5E](#)), error bars are SEM, bootstrapped over class (i.e., words, genres, or speakers) and stimuli.

For the similarity of layerwise representational similarity matrices with the target matrices ([Figure S6](#)), error bars are SEM, analytically computed for the correlation coefficient.

For layerwise classification of words, broken down by background noise level ([Figures 6A and S7A](#)), error bars are SEM, bootstrapped over stimuli and classes (i.e., words). For the layerwise ratio of response to less v. more stationary sounds ([Figures 6C, 6D, and S7B](#)), error bars are SEM, bootstrapped over sounds.

Significance of individual voxel predictions

The statistical significance of individual voxels was not directly relevant to most of the results described in the paper, because the key results involve pooling voxels together either explicitly across ROIs (either functionally-defined or reliability-defined, i.e., all reliable voxels in auditory cortex) or implicitly in coarse-scale patterns evident in the maps. Nevertheless, to assess significance of individual voxel predictions, we compared the variance explained in each voxel to a null model in which the procedure for calculating explained variance was repeated with random response and prediction vectors ([Figure S2B](#)).

We compared the observed raw r^2 values (i.e., those obtained before noise correction) with a null distribution obtained by correlating two 82-length vectors of Gaussian noise (because 82 sounds were used to measure r^2 values), taking the median across ten such samples (as we do with ten randomly selected sets of test stimuli), and repeating this procedure ten million times. The likelihood of observing an r^2 value by chance (i.e., when there is no correlation between the underlying variables) can be obtained from its probability under this null distribution. We set a criterion for determining that a single voxel is incorrectly considered significant ($p = 0.05$) and then applied a stringent correction for multiple comparisons (Bonferroni correction) by dividing this threshold by the total number of comparisons that we are making (7694 voxels across the eight subjects).

DATA AND SOFTWARE AVAILABILITY

The Tensorflow network implementation described above, including the trained filter weights, will be made available on the McDermott lab website.

Code and data are available by request to the Lead Contact (alexkell@mit.edu).

Review

Theories of Error Back-Propagation in the Brain

James C.R. Whittington^{1,2} and Rafal Bogacz^{1,*}

This review article summarises recently proposed theories on how neural circuits in the brain could approximate the error back-propagation algorithm used by artificial neural networks. Computational models implementing these theories achieve learning as efficient as artificial neural networks, but they use simple synaptic plasticity rules based on activity of presynaptic and postsynaptic neurons. The models have similarities, such as including both feedforward and feedback connections, allowing information about error to propagate throughout the network. Furthermore, they incorporate experimental evidence on neural connectivity, responses, and plasticity. These models provide insights on how brain networks might be organised such that modification of synaptic weights on multiple levels of cortical hierarchy leads to improved performance on tasks.

Deep Learning and Neuroscience

In the past few years, computer programs using **deep learning** (see [Glossary](#)) have achieved impressive results in complex cognitive tasks that were previously only in the reach of humans. These tasks include processing of natural images and language [1], or playing arcade and board games [2,3]. Since these recent deep learning applications use extended versions of classic **artificial neural networks** [4], their success has inspired studies comparing information processing in artificial neural networks and the brain. It has been demonstrated that when artificial neural networks learn to perform tasks such as image classification or navigation, the neurons in their layers develop representations similar to those seen in brain areas involved in these tasks, such as receptive fields across the visual hierarchy or grid cells in the entorhinal cortex [5–7]. This suggests that the brain may use analogous algorithms. Furthermore, thanks to current computational advances, artificial neural networks can now provide useful insights on how complex cognitive functions are achieved in the brain [8].

A key question that remains open is how the brain could implement the **error back-propagation** algorithm used in artificial neural networks. This algorithm describes how the weights of synaptic connections should be modified during learning, and its attractiveness, in part, comes from prescribing weight changes that reduce errors made by the network, according to a theoretical analysis. Although artificial neural networks were originally inspired by the brain, the modification of their synaptic connections, or weights, during learning appears biologically unrealistic [9,10]. Nevertheless, recent models have demonstrated that learning as efficient as in artificial neural networks can be achieved in distributed networks of neurons using only simple plasticity rules [11–14]. These theoretic studies are important because they overrule the dogma, generally accepted for the past 30 years, that the error back-propagation algorithm is too complicated for the brain to implement [9,10]. Before discussing this new generation of models in detail, we first provide a brief overview of how the back-propagation algorithm is used to train artificial neural networks and discuss why it was considered biologically implausible.

Highlights

The error back-propagation algorithm can be approximated in networks of neurons, in which plasticity only depends on the activity of presynaptic and postsynaptic neurons.

These biologically plausible deep learning models include both feedforward and feedback connections, allowing the errors made by the network to propagate through the layers.

The learning rules in different biologically plausible models can be implemented with different types of spike-time-dependent plasticity.

The dynamics and plasticity of the models can be described within a common framework of energy minimisation.

¹MRC Brain Network Dynamics Unit, Nuffield Department of Clinical Neurosciences, University of Oxford, Oxford OX3 9DU, UK

²Wellcome Centre for Integrative Neuroimaging, Centre for Functional Magnetic Resonance Imaging of the Brain, University of Oxford, Oxford OX3 9DU, UK

*Correspondence:
[\(R. Bogacz\).](mailto:rafal.bogacz@ndcn.ox.ac.uk)

Artificial Neural Networks and Error Back-Propagation

To effectively learn from feedback, the synaptic connections often need to be appropriately adjusted in multiple hierarchical areas simultaneously. For example, when a child learns to name letters, the incorrect pronunciation may be a combined result of incorrect synaptic connections in speech, associative, and visual areas. When a multi-layer artificial neural network makes an error, the error back-propagation algorithm appropriately assigns credit to individual synapses throughout all levels of hierarchy and prescribes which synapses need to be modified and by how much.

How is the back-propagation algorithm used to train artificial neural networks? The algorithm is trained on a set of examples, each consisting of an **input pattern** and a **target pattern**. For each such pair, the network first generates its prediction based on the input pattern and then the synaptic weights are modified to minimise the difference between the target and the **predicted pattern**. To determine the appropriate modification, an error term is computed for each neuron throughout the network. This describes how the activity of the neuron should change to reduce the discrepancy between the predicted and target pattern (Box 1). Each weight is modified by an amount determined by the product between the activity of the neuron it projects from and the error term of the neuron it projects to.

Although the described procedure is used to train artificial neural networks, analogous steps may take place during learning in the brain. For example, in the case of the child naming letters mentioned above, the input pattern corresponds to an image of a letter. After seeing an image, the child makes a guess at the name (predicted pattern) via a neural network between visual and speech areas. On supervision by his or her parent of the correct pronunciation (target pattern), synaptic weights along the processing stream are modified so that it is more likely that the correct sound will be produced when seeing that image again.

Biologically Questionable Aspects of the Back-Propagation Algorithm

Although the algorithmic process described above appears simple enough, there are a few problems with implementing it in biology. Below, we briefly discuss three key issues.

Lack of Local Error Representation

Conventional artificial neural networks are only defined to compute information in a forward direction, with the back-propagating errors computed separately by an external algorithm. Without local error representation, each synaptic weight update depends on the activity and computations of all downstream neurons. Since biological synapses change their connection strength based solely on local signals (e.g., the activity of the neurons they connect), it appears unclear how the synaptic plasticity afforded by the back-propagation algorithm could be achieved in the brain. Historically, this is a major criticism; thus it is a main focus of our review article.

Symmetry of Forwards and Backwards Weights

In artificial neural networks, the errors are back-propagated using the same weights as those when propagating information forward during prediction. This weight symmetry suggests that identical connections should exist in both directions between connected neurons. Although bidirectional connections are significantly more common in cortical networks than expected by chance, they are not always present [15]. Furthermore, even if bidirectional connections were always present, the backwards and forwards weights would still have to correctly align themselves.

Glossary

Anti-Hebbian plasticity: synaptic weight modifications proportional to the negative product of the activity of the pre- and postsynaptic neurons. Thus, if both neurons are highly active, the weight of connection between them is reduced.

Apical dendrite: a dendrite emerging from the apex of a pyramidal neuron (i.e., from the part of a cell body closest to the surface of the cortex).

Artificial neural networks: computing systems loosely based on brain networks. They consist of layers of ‘neurons’ communicating with each other via connections of different weights. Their task is to transform input patterns to particular target patterns. They are trained to predict target patterns in a process in which weights are modified according to the error back-propagation algorithm.

Deep learning: learning in artificial neural networks with more than two layers (often >10). Deep networks have shown much promise in the field of machine learning.

Equilibrium propagation: a principled framework for determining network dynamics and synaptic plasticity within energy-based models.

Error back-propagation: the main algorithm used to train artificial neural networks. It involves computations of errors associated with individual neurons, which determine weight modifications.

Error node: neuron type of predictive coding networks. They compute the difference between a value node and its higher-level prediction.

Hebbian plasticity: synaptic weight modifications proportional to the product of the activity of the pre- and postsynaptic neurons. It is called Hebbian in computational neuroscience, as it captures the idea of Donald Hebb that synaptic connections are strengthened between co-active neurons.

Input pattern: a vector containing the activity levels to which the neurons in the input layer are set. For example, in the handwritten digit classification problem, an input pattern corresponds to a picture of a digit. Here, the input pattern is a

Box 1. Artificial Neural Networks

A conventional artificial neural network consists of layers of neurons, with each neuron within a layer receiving a weighted input from the neurons in the previous layer (Figure 1A). The input layer is first set to be the input pattern and then a prediction is made by propagating the activity through the layers, according to **Equation 1.1**, where \mathbf{x}_l is a vector denoting neurons in layer l and \mathbf{W}_{l-1} is a matrix of synaptic weights from layer $l-1$ to layer l . An activation function f is applied to each neuron to allow for nonlinear computations.

During learning, the synaptic connections are modified to minimise a cost function quantifying the discrepancy between the predicted and target patterns (typically defined as in **Equation 1.2**). In particular, the weights are modified in the direction of steepest decrease (or gradient) of the cost function (Figure 1D). Such modification is described in **Equation 1.3**, where δ_{l+1} is a vector of error terms associated with neurons \mathbf{x}_{l+1} . The error terms for the last layer L are defined in **Equation 1.4** as the difference between the target activity \mathbf{t} and the predicted activity. Thus, the error of an output neuron is positive if its target activity is higher than the predicted activity. For the earlier layers, the errors are computed according to **Equation 1.5** as a sum of the errors of neurons in the layer above weighted by the strengths of their connections (and further scaled by the derivative of the activation function; in **Equation 1.5** · denotes element-wise multiplication). For example, an error of a hidden unit is positive if it sends excitatory projections to output units with high error terms, so increasing the activity of such a hidden neuron would reduce the error on the output. Once the errors are computed, each weight is changed according to **Equation 1.3** in proportion to the product of the error term associated with a postsynaptic neuron and the activity of a presynaptic neuron.

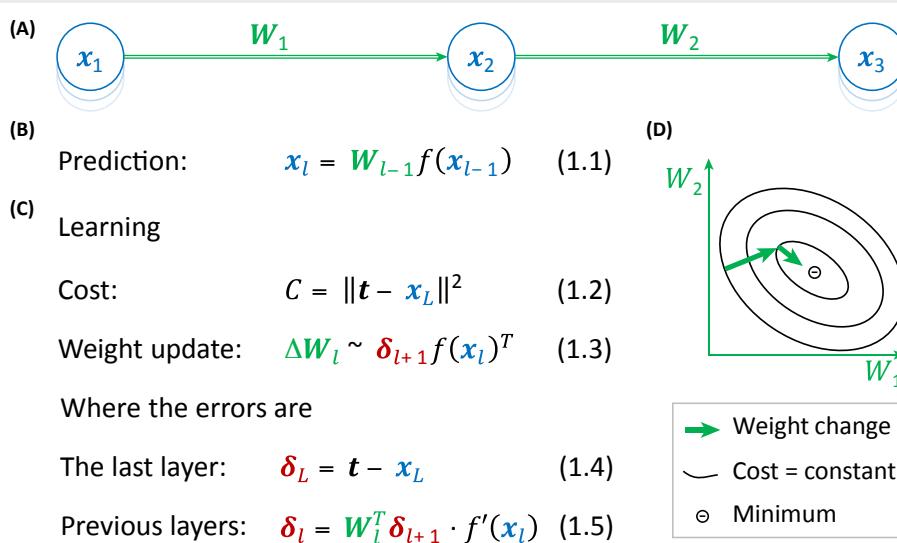


Figure 1. Artificial Neural Networks. (A) Layers of neuron-like nodes are represented by sets of stacked blue circles. Feedforward connections are indicated by green arrows. (B) Prediction. (C) Learning. (D) Schematic of the directions of two consecutive weight modifications (thick arrows) in the space of weights (for simplicity, only two dimensions are shown). Contours show points in weight space with equal cost function values.

Unrealistic Models of Neurons

Artificial neural networks use artificial neurons that send a continuous output (corresponding to a firing rate of biological neurons), whereas real neurons use spikes. Generalising the back-propagation algorithm to neurons using discrete spikes is not trivial, because it is unclear how to compute the derivative term found in the back-propagation algorithm (Box 1). Away from the back-propagation algorithm, the description of computations inside neurons in artificial neural networks is also simplified as a linear summation of inputs.

vector created by concatenating rows of pixels in the image, where each entry is equal to the darkness of the corresponding pixel.

Martinotti cells: small interneurons found in cortex.

Oscillatory rhythms: rhythmic patterns of neural activity, with activity of particular cells oscillating between higher and lower values.

Plateau potential: a sustained change in a membrane potential of a neuron, caused by persistent inward currents.

Predicted pattern: a vector of activities generated by the network in the output layer, by propagating the input pattern through layers. In the handwritten digit classification problem, the output layer has ten neurons corresponding to ten possible digits. The activity of each output neuron encodes the network's prediction for how likely the input pattern is to represent a particular digit.

Pyramidal neuron: an excitatory neuron with conically shaped cell body. Found in the cerebral cortex, hippocampus, and amygdala.

Spike-time-dependent plasticity: synaptic weight modification that depends on the relative timing between pre- and postsynaptic firing.

Supervised learning: a class of tasks considered in machine learning, where both an input and a target pattern are provided. The task for the algorithms is to learn to predict the target patterns from the input patterns.

Target pattern: a vector of activity in the output layer, which the network should generate for a given input pattern. For example, in the handwritten digit classification problem, the target pattern is equal to 1 at the position corresponding to the class of the corresponding image and is equal to 0 elsewhere.

Unsupervised learning: a class of tasks considered in machine learning where only an input pattern is provided (e.g., an image of a handwritten digit). The task for the learning algorithm is typically to learn an efficient representation of the data.

Value node: neuron type of predictive coding networks. Their activity represents the values computed by the network.

Models of Biological Back-Propagation

Each of the above-mentioned issues has been investigated by multiple studies. The lack of local error representation has been addressed by early theories by proposing that the errors associated with individual neurons are not computed, but instead the synaptic plasticity is driven by a global error signal carried by neuromodulators [16–19]. However, it has been demonstrated that learning in such models is slow and does not scale with network size [20]. More promisingly, in the past few years, several models have been proposed that do represent errors locally and thus more closely approximate the back-propagation algorithm. These models perform similarly to artificial neural networks on standard benchmark tasks (e.g., handwritten digit classification) [12–14,21,22], and we summarise several of them in more detail in the following sections.

The criticism of weight symmetry has been addressed by demonstrating that even if the errors in artificial neural networks are back-propagated by random connections, good performance in classification tasks can still be achieved [21,23–27]. This being said, there is still some concern regarding this issue [28]. With regard to the biological realism of neurons, it has been shown that the back-propagation algorithm can be generalised to neurons producing spikes [29] and that problems with calculating derivatives using spikes can be overcome [23]. Furthermore, it has been proposed that when more biologically realistic neurons are considered, they themselves may approximate a small artificial neural network in their dendritic structures [30].

There is a diversity of ideas on how the back-propagation algorithm may be approximated in the brain [31–36]; however, we review the principles behind a set of related models [11,13,14,37] that have substantial connections with biological data while closely paralleling the back-propagation algorithm. These models operate with minimal external control, as they can compute the errors associated with individual neurons through the dynamics of the networks. Thus, synaptic weight modifications depend only on the activity of presynaptic and postsynaptic neurons. Furthermore, these models incorporate important features of brain biology, such as **spike time-dependent plasticity**, patterns of neural activity during learning, and properties of **pyramidal neurons** and cortical microcircuits. We emphasise that these models rely on fundamentally similar principles. In particular, the models include both feedforward and feedback connections, thereby allowing information about the errors made by the network to propagate throughout the network without requiring an external program to compute the errors. Furthermore, these dynamics, as well as the synaptic plasticity, can be described within a common framework of energy minimisation. We divide the reviewed models in two classes differing in how the errors are represented, and we summarise them in the following sections.

Temporal-Error Models

This class of model encodes errors in the differences in neural activity across time. The first model in this class is the contrastive learning model [37]. It relies on an observation that weight changes proportional to an error (difference between predicted and target pattern) can be decomposed into two separate updates: one update based on activity without the target present and the other update with the target pattern provided to the output neurons [38] (Box 2). Thus, the error back-propagation algorithm can be approximated in a network in which the weights are modified twice: during prediction according to **anti-Hebbian plasticity** and then according to **Hebbian plasticity** once the target is provided and the network converges to an equilibrium (after the target activity has propagated to earlier layers via feedback connections) [37]. The role of the first modification is to ‘unlearn’ the existing association between input and prediction, while the role of the second modification is to learn the new association between input and target.

Box 2. Temporal-Error Models

Temporal-error models describe learning in networks with recurrent feedback connections to the hidden nodes (Figure 1A). The rate of change of activity of a given node is proportional to the summed inputs from adjacent layers, along with a decay term proportional to the current level of activity (Figure 1B). As the network is now recurrent, it is no longer possible to write a simple equation describing how the activity depends on other nodes (such as **Equation 1.1** in Box 1); instead, the dynamics of neurons is described by the differential **Equation 2.1** [72], where \dot{x}_l denotes the rate of change over time of x_l , (all equations in this figure ignore nonlinearities for brevity).

In the contrastive learning model, the weight modifications based on errors are decomposed into two separate changes occurring at different times. To understand learning in this model, it is easiest to consider how the weights connecting to the output layer are modified. Substituting **Equation 1.4** into **Equation 1.3**, we see in **Equation 2.2** that the weight modification required by the back-propagation algorithm can be decomposed into two terms. The first term corresponds to anti-Hebbian plasticity that should take place when the output activity is predicted based on the input propagated through the network. The second term corresponds to Hebbian plasticity that should take place when the output layer is set to the target pattern. O'Reilly [37] demonstrated that in the presence of backward connections, the information about the target pattern propagates to earlier layers, and an analogous sequence of weight modifications in the hidden layers also approximates a version of the back-propagation algorithm for recurrent networks [72].

In the continuous update model, the output nodes are gradually changed from the predicted pattern ($x_3|_{\neg t}$) towards the target values (t), as shown for a sample neuron in Figure 1D. Thus, the temporal derivative of output activity (\dot{x}_3) is proportional to $(t - x_3|_{\neg t})$, that is, to the error on the output (defined in **Equation 1.4**). Hence, the weight modification required by back-propagation is simply equal to the product of presynaptic activity and the rate of change of the postsynaptic activity (**Equation 2.3**).

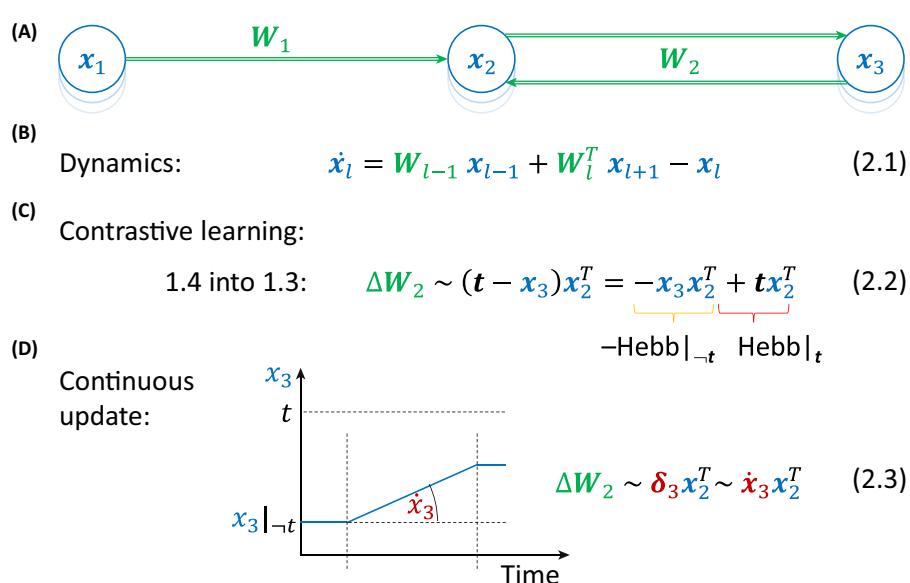


Figure 1. Temporal-Error Models. (A) Network architecture. (B) Dynamics. (C) Contrastive learning. (D) Continuous update.

Although the weight modifications in the contrastive learning model involve locally available information, implementing them biologically would require a global signal informing the network which phase it is in (whether the target pattern influences the network or not) as that determines whether the plasticity should be Hebbian or anti-Hebbian. It is not clear whether such a control

signal exists in the brain. This concern can be alleviated if the determination of learning phases is coordinated by information locally available in the **oscillatory rhythms** [39], such as hippocampal theta oscillations [40]. In these models, the neurons in the output layer are driven by feedforward inputs in one part of the cycle and forced to take the value of the target pattern in the other.

The complications of separate phases have been recently addressed in the continuous update model [11], where during training the output neuron activities are gradually changed from the predicted pattern towards the target. In this case, the rate of change of the output units is proportional to the error terms (Box 2). Consequently, the weight modification required by the back-propagation algorithm could arise from local plasticity based on the rate of change of activity. Although the continuous update model does not involve two different learning rules during prediction and learning, it still requires a control signal indicating whether the target pattern is present or not, because plasticity should not take place during prediction.

Explicit-Error Models

In this section, we describe alternative models that do not require control signals but as a trade-off have more complex architectures that explicitly compute and represent errors.

It has been recently noticed [14,41] that the error back-propagation algorithm can be approximated in a widely used model of information processing in hierarchical cortical circuits called predictive coding [42]. In its original formulation, the predictive coding model was developed for **unsupervised learning**, and it has been shown that when the model is presented with natural images, it learns representations similar to those in visual cortex [42]. Predictive coding models have also been proposed as a general framework for describing different types of information processing in the brain [43]. It has been recently shown that when a predictive coding network is used for **supervised learning**, it closely approximates the error back-propagation algorithm [14].

An architecture of a predictive coding network contains **error nodes** that are each associated with corresponding **value nodes**. During prediction, when the network is presented with an input pattern, activity is propagated between the value nodes via the error nodes. The network converges to an equilibrium, in which the error nodes decay to zero and all value nodes converge to the same values as the corresponding artificial neural network (Box 3). During learning, both the input and the output layers are set to the training patterns. The error nodes can no longer decrease their activity to zero; instead, they converge to values as if the errors had been back-propagated [14]. Once the state of the predictive coding network converges to equilibrium, the weights are modified, according to a Hebbian plasticity rule. These weight changes closely approximate that of the back-propagation algorithm.

An important property of the predictive coding networks is that they work autonomously: irrespective of the target pattern being provided, the same rules for node dynamics and plasticity are used. If the output nodes are unconstrained, the error nodes converge to zero, so the Hebbian weight change is equal to zero. Thus, the networks operate without any need for external control except for providing different inputs and outputs. However, the one-to-one connectivity of error nodes to their corresponding value nodes is inconsistent with diffused patterns of neuronal connectivity in the cortex.

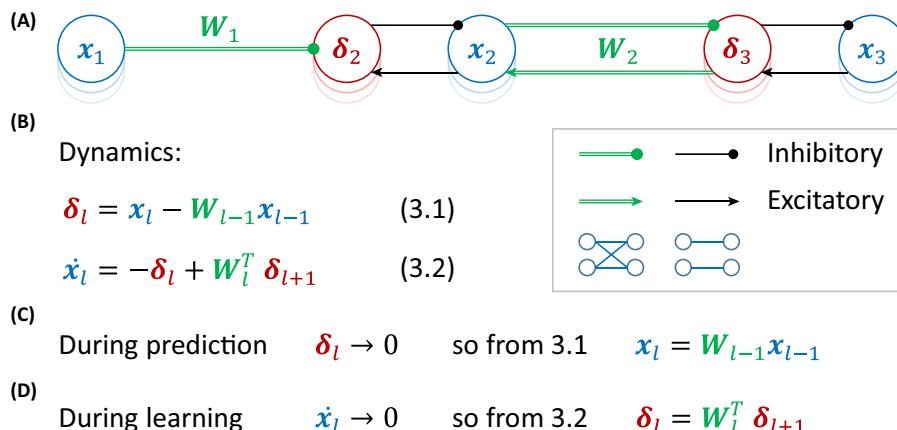
A solution to this inconsistency has been proposed in several models in which the error is represented in dendrites of the corresponding neuron [44–46]. In this review article, we focus on a popular model called the dendritic error model [13]. This model describes networks of

Box 3. Predictive Coding Model

Predictive coding networks include error nodes each associated with corresponding value nodes (Figure 1A). The error nodes receive inhibition from the previous layer and excitation from the corresponding value nodes and thus compute the difference between them (**Equation 3.1**). The value nodes get feedforward inhibition from corresponding error nodes and feedback from the error nodes in the next layer. In the predictive coding network, the value nodes act as integrators, so they add their input to their current activity level (**Equation 3.2**).

During prediction, when the network is presented only with an input pattern, the information is propagated between the value nodes via the error nodes. As the output layer is unconstrained, the activity of error nodes converges to zero, because the value nodes change their activity until the feedback they send to their corresponding error nodes balances the feedforward inhibition received by error nodes. At this state, the left side of **Equation 3.1** is equal to 0, and by rearranging terms (Figure 1C), we observe that the activity of value nodes is equal to the weighted sum of value nodes in the previous layer, exactly as in artificial neural networks [**Equation 1.1** with $f(x) = x$].

During learning, when the network is presented with both input and target patterns, the activity of error nodes may not decrease to zero. Learning takes place when the network is in equilibrium ($\dot{x}_l = 0$). At this stage the left side of **Equation 3.2** is equal to 0, and by rearranging terms (Figure 1D), we observe that the activity of error nodes is equal to a weighted sum of errors from the layer above, bearing the same relationship as in the back-propagation algorithm [**Equation 1.5** with $f(x) = x$]. At convergence, the weights are modified according to **Equation 1.3**, which here corresponds to Hebbian plasticity dependent on the activity of pre- and postsynaptic neurons.



Trends in Cognitive Sciences

Figure 1. Predictive Coding. (A) Network architecture. Blue and red circles denote the value and error nodes, respectively. Arrows and lines ending with circles denote excitatory and inhibitory connections, respectively; green double lines indicate connections between all neurons in one layer and all neurons in the next layer, while single black lines indicate within layer connections between a corresponding error and value node (see key). (B) Dynamics (for a simple case of linear function f ; for details of how nonlinearities can be introduced, see [14]). (C) Prediction. (D) Learning.

pyramidal neurons and assumes that the errors in the activity of pyramidal neurons are computed in their **apical dendrites**. In this model, the apical dendrites compare the feedback from the higher levels with a locally generated prediction of higher-level activity computed via interneurons.

An easy way to understand why such an architecture approximates the back-propagation algorithm is to notice that it is closely related to predictive coding networks, which approximate artificial neural networks. Simply rearranging the equations describing the dynamics of predictive coding model gives a description of a network with the same architecture as the dendritic error model, in which dendrites encode the error terms (Box 4).

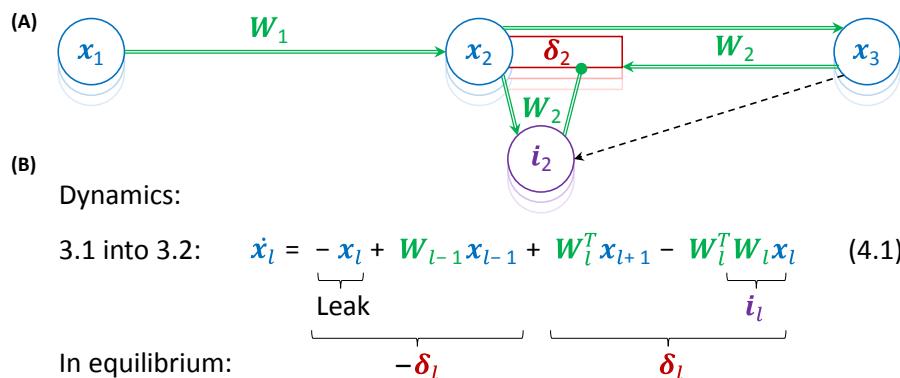
Box 4. Dendritic Error Model

The architecture of the dendritic error model [13] is shown in Figure 1A. In this network, the activity is propagated through the layers via connections between pyramidal neurons. The errors in the activity of pyramidal neurons are computed in their apical dendrites.

The relationship between predictive coding and dendritic error models can be established by observing that substituting the definition of error nodes from the predictive coding model, **Equation 3.1**, into **Equation 3.2**, produces **Equation 4.1**, which describes the dynamics of pyramidal neurons in Figure 1A. The right side of **Equation 4.1** consists of four terms corresponding to various connections in the figure. The first is simply a decay, the second is a feedforward input from the previous layer, the third is a feedback from the layer above, and the fourth term is a within layer recurrent input. This last term has a negative sign, while pyramidal neurons are excitatory, so it needs to be provided by interneurons. If we assume that the interneurons have activity $i_l = \mathbf{W}_l \mathbf{x}_l$, they need to be connected with the pyramidal neurons via weights \mathbf{W}_l .

The key property of this network is that when it converges to the equilibrium, the neurons with activity \mathbf{x}_l encode their corresponding error terms δ_l in their apical dendrites. To see why this is the case, note that the first two terms on the right of **Equation 4.1** are equal to $-\delta_l$ according to the definition of **Equation 3.1**. At equilibrium $\dot{\mathbf{x}}_l = 0$, the two last terms in **Equation 4.1** must be equal to δ_l (so that the right-hand side of **Equation 4.1** adds up to 0), and it is these two terms that define the input to the apical dendrite. As the errors δ_l are encoded in apical dendrites, the weight modification required by the back-propagation algorithm (**Equation 1.3**) only involves quantities encoded in pre- and postsynaptic neurons.

Appropriately updating weights between pyramidal and interneurons is more challenging. This is because the interneurons must learn to produce activity encoding the same information as the higher-level pyramidal neurons. To allow training of the interneurons, the dendritic error model includes special one-to-one connections to the interneurons from corresponding higher-level pyramidal neurons (black dashed arrows in Figure 1A).



Trends in Cognitive Sciences

Figure 1. Dendritic Error Model. (A) Network architecture. Blue circles indicate pyramidal neurons, red rectangles indicate their apical dendrites, and purple circles denote interneurons. (B) Dynamics.

As the error term is now encoded within a neuron's compartment, the update of weights between pyramidal neurons required by the back-propagation algorithm corresponds to local synaptic plasticity. Error information can be transmitted from the apical dendrite to the rest of the neuron through internal signals. For example, a recent computational model proposed that errors encoded in apical dendrites can determine the plasticity in the whole neuron [12]. The model is based on observations that activating apical dendrites induces **plateau potentials** via calcium influx, leading to a burst of spikes by the neuron [47]. Such bursts of spikes may subsequently trigger synaptic plasticity [48,49].

Although the dendritic error network makes significant steps to increase the biological realism of predictive coding models, it also introduces extra one-to-one connections (dotted arrow in **Box 4**) that enforce the interneurons to take on similar values to the neurons in next layer and thus help them to predict the feedback from the next level. Furthermore, the exact dynamics in the dendritic error model are much more complex than that given in **Box 4**, as it describes details of changes in membrane potential in multiple compartments. Nevertheless, it is important to highlight that the architecture of dendritic error networks can approximate the back-propagation algorithm, and it offers an alternative hypothesis on how the computations assumed by the predictive coding model could be implemented in cortical circuits.

Comparing the Models

Given the biological plausibility of the above-mentioned models, in this and the coming sections, we compare the models in terms of their computational properties (as more efficient networks may be favoured by evolution) and their relationships to experimental data (summarised in **Table 1**).

Computational Properties

For correct weight modification, the temporal-error models require a mechanism informing whether the target pattern constrains the output neurons, while the explicit-error models do not. However, as a trade-off, the temporal-error models have simpler architectures, while the explicit-error models need to have intricate architectures with certain constraints on connectivity, and both predictive coding and the dendritic error model include one-to-one connections in their network structure. As mentioned, there is no evidence for such one-to-one connectivity in the neocortex.

The models differ in the time required for signals to propagate through the layers. To make a prediction in networks with L layers, predictive coding networks need to propagate information through $2L - 1$ synapses, whereas the other models only need to propagate through $L - 1$ synapses. This is because in a predictive coding network, to propagate from one layer to the next, the information must travel via an error neuron, whereas in the other models the

Table 1. Comparison of Models

		Temporal-error model		Explicit-error model	
		Contrastive learning	Continuous update	Predictive coding	Dendritic error
Properties ^a	Control signal	Required	Required	Not required	Not required
	Connectivity	Unconstrained	Unconstrained	Constrained	Constrained
	Propagation time	L-1	L-1	2L-1	L-1
	Pre-training	Not required	Not required	Not required	Required
Error encoded in		Difference in activity between separate phases	Rate of change of activity	Activity of specialised neurons	Apical dendrites of pyramidal neurons
Data accounted for		Neural responses and behaviour in a variety of tasks	Typical spike-time-dependent plasticity	Increased neural activity to unpredicted stimuli	Properties of pyramidal neurons
MNIST performance ^b		~2–3	–	~1.7	~1.96

^aGreen indicates properties desired for biological plausibility, while red indicates less desired properties.

^bThese are error percentages reported on a testing set in a benchmark task of handwritten digit classification (lower is better), for predictive coding [14], dendritic error [13], and contrastive learning models [22] (in this simulation, the output neurons were not set to the target pattern, but slightly moved or 'nudged' towards it). We are not aware of reported simulations of the continuous update model on this benchmark problem. MNIST, Modified National Institute of Standards and Technology database.

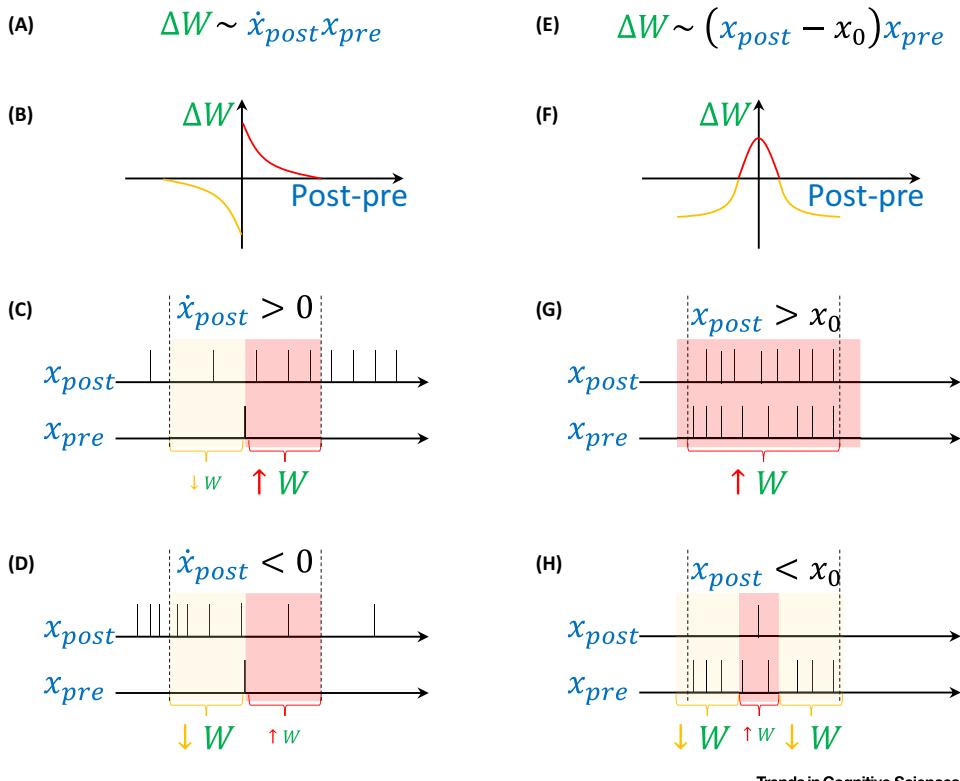
information is propagated directly to the neurons in the layer above. There is a clear evolutionary benefit to propagating information via fewer synapses, as it would result in faster responses and a smaller number of noise sources.

In the dendritic error model, for errors to be computed in the dendrites, the inhibitory interneurons first need to learn to predict the feedback from the higher level. Thus, before the network can learn feedforward connections, ideally the inhibitory neurons need to first be pre-trained. Although it has been shown that the feedforward and inhibitory weights can be learned in parallel, learning in the dendritic error model may well be slower as the reported number of iterations required to learn a benchmark task was higher for the dendritic error model [13] than for contrastive learning [22] and predictive coding [14] models. Such statements, however, should be taken with reservations as not only were simulations not necessarily comparable but also computations in standard von-Neumann computers may not be representative of computations in biological hardware.

Relationship to Experimental Data

The models differ in their predictions on whether errors should be explicitly represented in neural activity. In particular, the predictive coding model includes dedicated neurons encoding errors, and the dendritic error model suggests that errors computed in dendrites may trigger bursts of firing of pyramidal neurons, while in temporal models there is no direct association between error and the overall activity level at a given time. In line with the explicit-error models, increased neural activity has been observed when sensory input does not match the expectations encoded by higher-level areas. For example, responses of neurons in the primary visual cortex were increased at brief intervals in which visual input did not match expectation based on animal movements [50]. An increase in neural activity when expectations about stimuli were violated has also been found with fMRI [51]. Further details are discussed in several excellent reviews [52–55]. The two explicit models differ in predictions on whether errors and values are represented by separate neuronal populations or within the same neurons. Experimental data relevant to this question have been reviewed in an excellent chapter by Kok and de Lange [56]. Although they conclude that there is ‘no direct unequivocal evidence for the existence of separate populations’, they discuss several studies suggesting preferential encoding of errors and values by different neurons. For example, in a part of visual cortex (inferior temporal cortex), the inhibitory neurons tended to have higher responses to novel stimuli, while excitatory neurons typically produced highest response for their preferred familiar stimuli [57]. Kok and de Lange point that these responses may potentially reflect error and value nodes, respectively [56].

Each model accounts for specific aspects of experimental data. The models based on contrastive learning rules have been shown to reproduce neural activity and behaviour in a wide range of tasks [58]. The learning rule in the continuous update model (in which the synaptic modification depends on the rate of change of the postsynaptic neuron; *Figure 1A*), can be implemented with classic spike-time-dependent plasticity (*Figure 1B*) [11]. In this form of plasticity, the direction of modification (increase or decrease) depends on whether the spike of a presynaptic neuron precedes or follows the postsynaptic spike [59]. *Figure 1C* shows the effect of such plasticity in a case when the postsynaptic neuron increases its firing. If the postsynaptic spike follows the presynaptic spike, the synaptic weight is increased (pink area), while if the postsynaptic spike precedes the presynaptic spike, the weight is decreased (yellow area). If the postsynaptic neuron increases its firing rate (as in the example), there will be more postsynaptic spikes in pink than in yellow area on average, so the overall weight change will be positive. Analogously, the weight is weakened if the postsynaptic activity decreases (*Figure 1D*).



Trends in Cognitive Sciences

Figure 1. Relationship between Learning Rules and Spike-Time-Dependent Plasticity. (A) Plasticity dependent on the rate of change of postsynaptic activity, illustrated by the left column of panels. (B) Asymmetric spike-time-dependent plasticity often observed in cortical neurons [59]. The curve schematically shows the change in synaptic weights as a function of the difference between the timings of postsynaptic and presynaptic spikes. Red and orange parts of the curve correspond to increases and decreases in synaptic weights, respectively. (C) Strengthening of a synaptic weight due to increasing postsynaptic activity. Hypothetical spike trains of two neurons are shown. The top sequence corresponds to an output neuron, which increases its activity over time towards the target (see Figure 1D in Box 2). The bottom sequence corresponds to a neuron in the hidden layer; for simplicity, only a single spike is shown. The pink and yellow areas correspond to spike timings in which the weights are increased and decreased, respectively. In these areas the differences in spike timing result in weight changes indicated by red and orange parts of the curve in the panel B. (D) Weakening of weight due to decrease in postsynaptic activity. (E) Plasticity dependent on postsynaptic activity, illustrated by the right column of panels. In the equation, x_0 denotes the baseline firing rate. (F) Symmetric spike-time-dependent plasticity, where weight change depends on spike proximity. (G) Increase in synaptic weight due to high activity of the postsynaptic neuron. (H) Decrease in synaptic weight when the postsynaptic neuron is less active.

In summary, with asymmetric spike-time-dependent plasticity, the direction of weight change depends on the gradient of a postsynaptic neuron activity around a presynaptic spike, as in the continuous update model.

The relationship of spike-time-dependent plasticity to other models requires further clarifying work. Nevertheless, Vogels and colleagues [60] demonstrated that a learning rule in which the direction of modification depends on activity of neurons in equilibrium (Figure 1E), as in the predictive coding model, can arise from an alternate form of spike-time-dependent plasticity. They considered a form of plasticity where the weight is increased by nearly coincident pre- and postsynaptic spikes, irrespectively of their order, and additionally the weight is slightly decreased by each presynaptic spike. The overall direction of weight modification in this rule is shown in Figure 1F. Such a form of plasticity may exist in a several types of synapse in the

brain [61]. Figure 1G illustrates that with such plasticity, the weights are increased if the intervals between pre- and postsynaptic spikes are short, which is likely to occur when the two neurons have high activity. When the postsynaptic neuron is less active (Figure 1H), the short intervals (pink area) are less common, while longer intervals are more common (yellow area), so overall the weight change is negative. In summary, with symmetric spike-time-dependent plasticity the direction of weight change depends on whether the postsynaptic neuron activity is above or below a certain level (which may correspond to a baseline level typically denoted with zero in computational models), as in the predictive coding model.

The dendritic error model describes the computations in apical dendrites of pyramidal neurons and features of cortical micro-circuitry such as connectivity of a group of interneurons called the **Martinotti cells**, which receive input from pyramidal neurons in the same cortical area [62] and project to their apical dendrites [63]. Furthermore, there is some evidence that inhibitory interneurons also receive feedback from higher areas in the cortical hierarchy [64].

Integrating Models

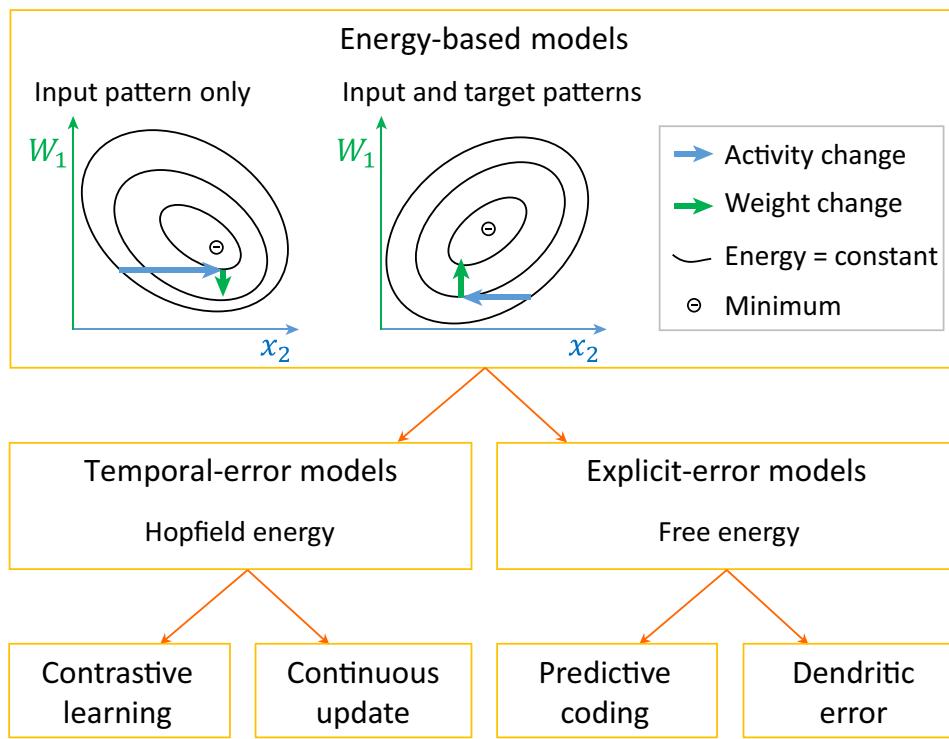
The above-mentioned comparison shows that each model has its own computational advantages, accounts for different data, and describes plasticity at different types of synapses. It is important to note that the cortical circuitry is much more complicated than any of the proposed models' architectures. Therefore, the models presented above need not be viewed as competitors but may be considered as descriptions of learning in different motifs of more complex brain networks.

Different classes of models may be more suited for different tasks faced by brain networks. One task engaging the primary sensory areas is predicting the next value of sensory input from the previous ones. A recent modelling study suggests that primary visual and auditory cortices may use an algorithm similar to back-propagation while learning to predict sensory input [65]. This study demonstrated that the temporal properties of receptive field in these areas are similar to those in artificial neural networks trained to predict the next video or audio frames on the basis of past history in clips of natural scenes [65]. In such sensory prediction tasks, the target (i.e., the next 'frame' of sensory input) always arrives, so the temporal-error models may be particularly suited for this task, as there is no need for the control signal indicating target presence.

The explicit-error models are suitable for tasks where the timing of target pattern presentation is more uncertain. Although the predictive coding and dendritic error networks are closely related, they also exhibit a trade-off: the predictive coding networks are slow to propagate information once trained, while the dendritic error networks are slower to train. It is conceivable that cortical networks include elements of predictive coding networks in addition to dendritic error motifs, as the cortical networks include many other interneuron types in addition to the Martinotti cells and have a much richer organisation than either model. Such a combined network could initially rely on predictive coding motifs to support fast learning and, with time, the dendritic error models could take over, allowing faster information processing. Thus, by combining different motifs, brain networks may 'beat the trade-offs' and inherit advantages of each model.

Furthermore, predictive coding models may describe information processing in subcortical parts of brain networks that do not include pyramidal cells and thus may not be able to support computations of the dendritic error model. Indeed, it has been recently suggested how the predictive coding model can be mapped on the anatomy of cerebellum [66], and the model may also describe aspects of information processing in basal ganglia, where the dopaminergic neurons are well known to encode reward prediction error in their activity [67].

As the brain networks may incorporate elements of different models, it is important to understand how individual models relate to each other and how they can be combined. Such insights have been revealed by a recently proposed framework called **equilibrium propagation** [22,68]. Here, it was noticed that the dynamics of many models of neuronal networks can be defined in terms of the optimisation of a particular function. This function is known as the network energy. For example, recurrently connected networks of excitatory neurons, such as the temporal-error models, under certain assumptions converge to an equilibrium in which strongly connected neurons tend to have similar levels of activity. Indeed, they minimise a function that summarises the dissimilarity in the activity of strongly connected nodes, called the Hopfield energy [69]. The predictive coding networks are also known to minimise a function during their dynamics, called the free energy [70]. The free energy has a particularly nice statistical interpretation, as its negative provides a lower bound on the log probability of predicting the target pattern by the network [70,71] (in case of supervised learning, this



Trends in Cognitive Sciences

Figure 2. Equilibrium Propagation. The framework considers networks with dynamics described by the minimisation of an energy function. As the activity of these networks converges to an equilibrium, the energy simultaneously decays (blue arrows) to a minimum given the current weights. Once in equilibrium, the weights are modified (green arrows). It has been shown that network error can be minimised if the synaptic weights are modified in two steps (schematically illustrated by the two displays in the top box; [22]). First, with only the input pattern provided, once the network converges, weights are modified in the direction in which the energy increases. Second, the output layer is additionally constrained to values closer to the target pattern (particular details described in [22]). Constraining the output nodes changes the energy landscape for the units in the middle layers. Once these units converge to a new equilibrium, weights are modified in the direction in which the energy decreases. Scellier and Bengio [22] noted that for temporal-error networks, this procedure gives the contrastive learning rule (**Equation 2.2**). The predictive coding networks, however, converge to an equilibrium in the first step where the free-energy function reaches its global minimum [14]; thus, there is no weight modification required by the equilibrium propagation framework. Therefore, only a single phase (i.e., the second phase) and a single weight update are required in the explicit-error models, and it only involves Hebbian plasticity.

probability is conditioned on the input patterns). Since the dendritic error models have approximately similar dynamics as the predictive coding models, all models reviewed above can be considered as energy-based models described within the equilibrium propagation framework (Figure 2).

The framework also prescribes how synaptic weights should be modified in any network that minimises energy, and the weight modifications in the reviewed models indeed follow this general rule (Figure 2). Importantly, the framework can describe learning in more complex networks, which could include the elements of the different models. For any network for which an energy function can be defined, the framework describes the plasticity rules of individual synapses required for efficient learning.

Nevertheless, the form of energy function minimised by a network may influence its performance. So far, the biologically plausible networks that perform best in a handwritten digit classification task are those that minimise energies analogous to the free energy (Table 1). The superior performance of networks minimising free energy may stem from the probabilistic interpretation of free energy, which ensures that the networks are trained to maximise the probability of predicting target patterns.

Concluding Remarks

This review article has not been exhaustive of all current biological models but nevertheless has described main classes of recent models; those that represent errors temporally and those that represent them explicitly, as well as a framework unifying these methods. These theoretic results elucidate the constraints required for efficient learning in hierarchical networks. However, much more work needs to be done both empirically and theoretically, for example, on how the networks scale to larger architectures [28], as well as linking theory to neurobiological data (see Outstanding Questions).

It is crucial to map the models implementing efficient deep learning on biological networks in the brain. In particular, mapping the nodes in the model on distinct cell types in the cortex may be a fruitful route to identifying their computational function. The framework of equilibrium propagation (or its future extensions) may prove particularly useful in this endeavour. Based on known patterns of connectivity, models could be defined and their energy function formulated. The framework could then be used to predict properties of synaptic plasticity that could be compared with experimental data, and the results of such comparisons could be iteratively used to improve the models.

Acknowledgements

This work was supported by Medical Research Council grant MC_UU_12024/5 and the Engineering and Physical Sciences Research Council. We thank Lindsey Drayton, Tim Vogels, Friedemann Zenke, Joao Sacramento, and Benjamin Scellier for thoughtful comments.

References

1. LeCun, Y. *et al.* (2015) Deep learning. *Nature* 521, 436–444
2. Mnih, V. *et al.* (2015) Human-level control through deep reinforcement learning. *Nature* 518, 529–533
3. Silver, D. *et al.* (2016) Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484–489
4. Rumelhart, D.E. *et al.* (1986) Learning representations by back-propagating errors. *Nature* 323, 533–536
5. Banino, A. *et al.* (2018) Vector-based navigation using grid-like representations in artificial agents. *Nature* 557, 429–433
6. Whittington, J.C.R. *et al.* (2018) Generalisation of structural knowledge in the hippocampal-entorhinal system. In *31st Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal
7. Yamins, D.L. and DiCarlo, J.J. (2016) Using goal-driven deep learning models to understand sensory cortex. *Nat. Neurosci.* 19, 356–365
8. Bowers, J.S. (2017) Parallel distributed processing theory in the age of deep networks. *Trends Cogn. Sci.* 21, 950–961

Outstanding Questions

Are biologically plausible deep learning implementations robust to the lack of symmetry between the feedforward and feedback connections? The four models reviewed use symmetric feed-forward and feedback weights. In these models, both sets of weights are modified during learning, and the plasticity rules maintain the symmetry. As mentioned, such symmetry does not exist in brain networks, so it is important to continue investigations into whether biologically plausible networks still perform robustly without weight symmetry.

How can researchers make biologically plausible deep learning implementations scale? Although the above-mentioned models perform well on some tasks, it is unclear whether they scale to larger problems. This is in part due to the multiple iterations required to update node activity via network dynamics. The number of iterations required does not currently scale well for larger networks. Further work optimising this process is required if high depth networks are to be trained.

How can efficient learning of temporal sequences be implemented in biological networks? The models reviewed above focus on a case of static input patterns, but the sensory input received by the brain is typically dynamic, and the brain has to learn to recognise sequences of stimuli (e.g. speech). To describe learning in such tasks, artificial neural networks have been extended to include recurrent connections among hidden units, which provide a memory of the past. It is important to extend the models reviewed above for learning through time.

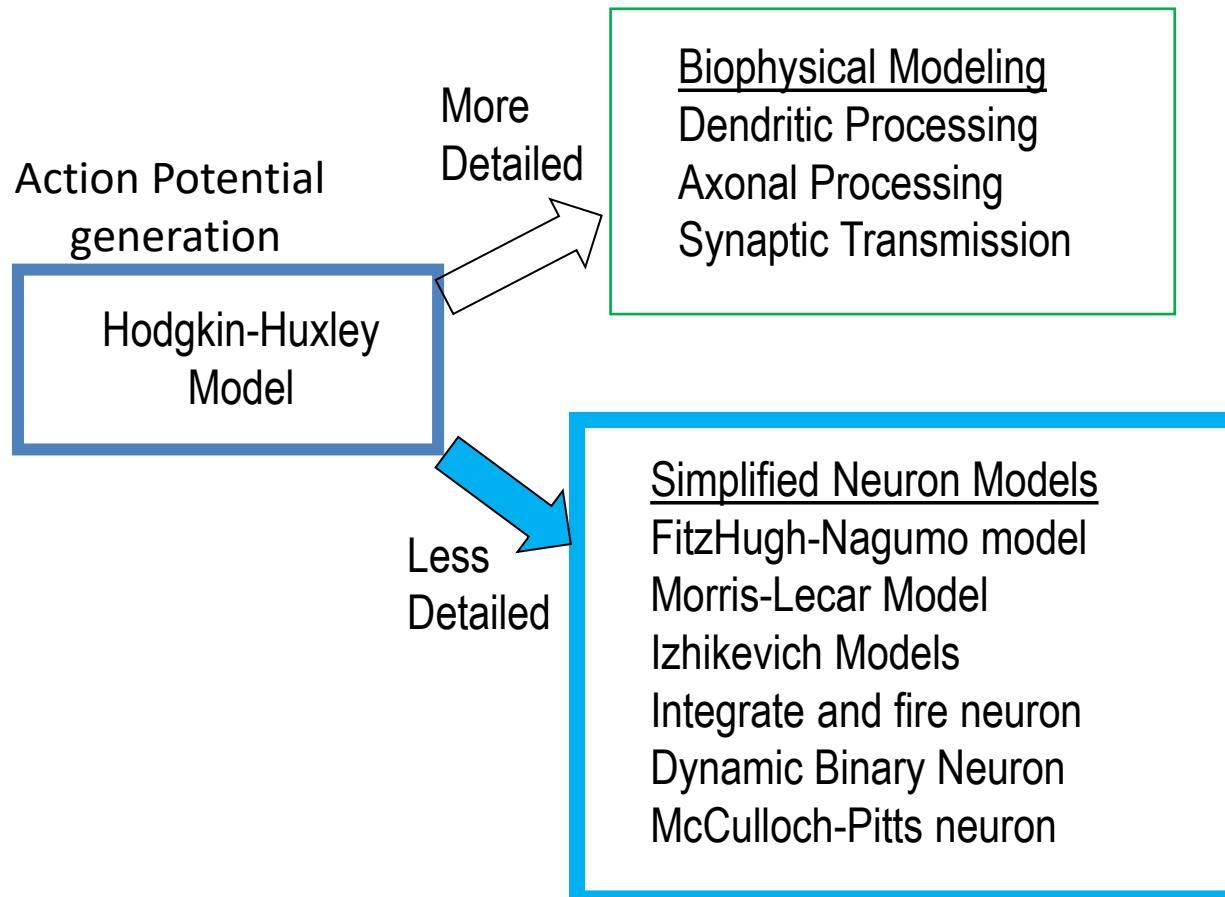
How can the dynamics of neural circuits be optimised to support efficient learning? This question can be first studied in models of primary sensory areas predicting sensory input from its past values. In such tasks, the dynamics will play an important role, as networks need to generate their predictions at the right time to compare it with incoming sensory data.

9. Crick, F. (1989) The recent excitement about neural networks. *Nature* 337, 129–132
10. Grossberg, S. (1987) Competitive learning: from interactive activation to adaptive resonance. *Cogn. Sci.* 11, 23–63
11. Bengio, Y. et al. (2017) STDP-Compatible approximation of backpropagation in an energy-based model. *Neural Comput.* 29, 555–577
12. Guergiev, J. et al. (2017) Towards deep learning with segregated dendrites. *eLife* 6, e22901
13. Sacramento, J. et al. (2018) Dendritic cortical microcircuits approximate the backpropagation algorithm. In *31st Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal
14. Whittington, J.C.R. and Bogacz, R. (2017) An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Comput.* 29, 1229–1262
15. Song, S. et al. (2005) Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biol.* 3, 507–519
16. Mazzoni, P. et al. (1991) A more biologically plausible learning rule for neural networks. *Proc. Natl. Acad. Sci. U. S. A.* 88, 4433–4437
17. Williams, R.J. (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8, 229–256
18. Unnikrishnan, K.P. and Venugopal, K.P. (1994) Alopex: a correlation-based learning algorithm for feedforward and recurrent neural networks. *Neural Comput.* 6, 469–490
19. Seung, H.S. (2003) Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron* 40, 1063–1073
20. Werfel, J. et al. (2005) Learning curves for stochastic gradient descent in linear feedforward networks. *Neural Comput.* 17, 2699–2718
21. Lillicrap, T.P. et al. (2016) Random synaptic feedback weights support error backpropagation for deep learning. *Nat. Commun.* 7, 13276
22. Scellier, B. and Bengio, Y. (2017) Equilibrium propagation: bridging the gap between energy-based models and backpropagation. *Front. Comput. Neurosci.* 11, 24
23. Zenke, F. and Ganguli, S. (2018) SuperSpike: supervised learning in multilayer spiking neural networks. *Neural Comput.* 30, 1514–1541
24. Mostafa, H. et al. (2017) Deep supervised learning using local errors. arXiv preprint arXiv:1711.06756
25. Scellier, B. et al. (2018) Generalization of equilibrium propagation to vector field dynamics. arXiv 1808.04873
26. Liao, Q. et al. (2016) How important is weight symmetry in backpropagation? In *AAAI Conference on Artificial Intelligence*, pp. 1837–1844, AAAI
27. Baldi, P. and Sadowski, P. (2016) A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Netw.* 83, 51–74
28. Bartunov, S. et al. (2018) Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *31st Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal
29. Sporea, I. and Grüning, A. (2013) Supervised learning in multilayer spiking neural networks. *Neural Comput.* 25, 473–509
30. Schiess, M. et al. (2016) Somato-dendritic synaptic plasticity and error-backpropagation in active dendrites. *PLoS Comput. Biol.* 12, e1004638
31. Balduzzi, D. et al. (2015) Kickback cuts backprop's red-tape: biologically plausible credit assignment in neural networks. In *AAAI Conference on Artificial Intelligence*, pp. 485–491, AAAI
32. Krotov, D. and Hopfield, J. (2018) Unsupervised learning by competing hidden units. arXiv preprint arXiv:1806.10181
33. Kuśmierz, Ł. et al. (2017) Learning with three factors: modulating Hebbian plasticity with errors. *Curr. Opin. Neurobiol.* 46, 170–177
34. Marblestone, A.H. et al. (2016) Toward an integration of deep learning and neuroscience. *Front. Comput. Neurosci.* 10, 94
35. Bengio, Y. (2014) How auto-encoders could provide credit assignment in deep networks via target propagation. arXiv preprint arXiv:1407.7906
36. Lee, D.-H. et al. (2015) Difference target propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 498–515, Springer
37. O'Reilly, R.C. (1996) Biologically plausible error-driven learning using local activation differences: the generalized recirculation algorithm. *Neural Comput.* 8, 895–938
38. Ackley, D.H. et al. (1985) A learning algorithm for Boltzmann machines. *Cogn. Sci.* 9, 147–169
39. Baldi, P. and Pineda, F. (1991) Contrastive learning and neural oscillations. *Neural Comput.* 3, 526–545
40. Ketz, N. et al. (2013) Theta coordinated error-driven learning in the hippocampus. *PLoS Comput. Biol.* 9, e1003067
41. Ororbia, A.G. and Mali, A. (2018) Biologically motivated algorithms for propagating local target representations. arXiv preprint arXiv:1805.11703
42. Rao, R.P.N. and Ballard, D.H. (1999) Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.* 2, 79–87
43. Friston, K.J. (2010) The free-energy principle: a unified brain theory? *Nat. Rev. Neurosci.* 11, 127–138
44. Richards, B.A. and Lillicrap, T.P. (2019) Dendritic solutions to the credit assignment problem. *Curr. Opin. Neurobiol.* 54, 28–36
45. Körding, K.P. and König, P. (2001) Supervised and unsupervised learning with two sites of synaptic integration. *J. Comput. Neurosci.* 11, 207–215
46. Körding, K.P. and König, P. (2000) Learning with two sites of synaptic integration. *Network* 11, 25–39
47. Larkum, M.E. et al. (1999) A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature* 398, 338–341
48. Pike, F.G. et al. (1999) Postsynaptic bursting is essential for 'Hebbian' induction of associative long-term potentiation at excitatory synapses in rat hippocampus. *J. Physiol.* 518, 571–576
49. Roelfsema, P.R. and Holtmaat, A. (2018) Control of synaptic plasticity in deep cortical networks. *Nat. Rev. Neurosci.* 19, 166
50. Attinger, A. et al. (2017) Visuomotor coupling shapes the functional development of mouse visual cortex. *Cell* 169, 1291–1302
51. Summerfield, C. et al. (2008) Neural repetition suppression reflects fulfilled perceptual expectations. *Nat. Neurosci.* 11, 1004
52. Summerfield, C. and de Lange, F.P. (2014) Expectation in perceptual decision making: neural and computational mechanisms. *Nat. Rev. Neurosci.* 15, 745–756
53. Bastos, A.M. et al. (2012) Canonical microcircuits for predictive coding. *Neuron* 76, 695–711
54. de Lange, F.P. et al. (2018) How do expectations shape perception? *Trends Cogn. Sci.* 22, 764–779
55. Clark, A. (2013) Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behav. Brain Sci.* 36, 181–204
56. Kok, P. and de Lange, F.P. (2015) Predictive coding in sensory cortex. In *An Introduction to Model-Based Cognitive Neuroscience*, pp. 221–244, Springer
57. Wołoszyn, L. and Sheinberg, D.L. (2012) Effects of long-term visual experience on responses of distinct classes of single units in inferior temporal cortex. *Neuron* 74, 193–205
58. O'Reilly, R.C. and Munakata, Y. (2000) *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*, MIT Press
59. Bi, G.Q. and Poo, M.M. (1998) Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* 18, 10464–10472

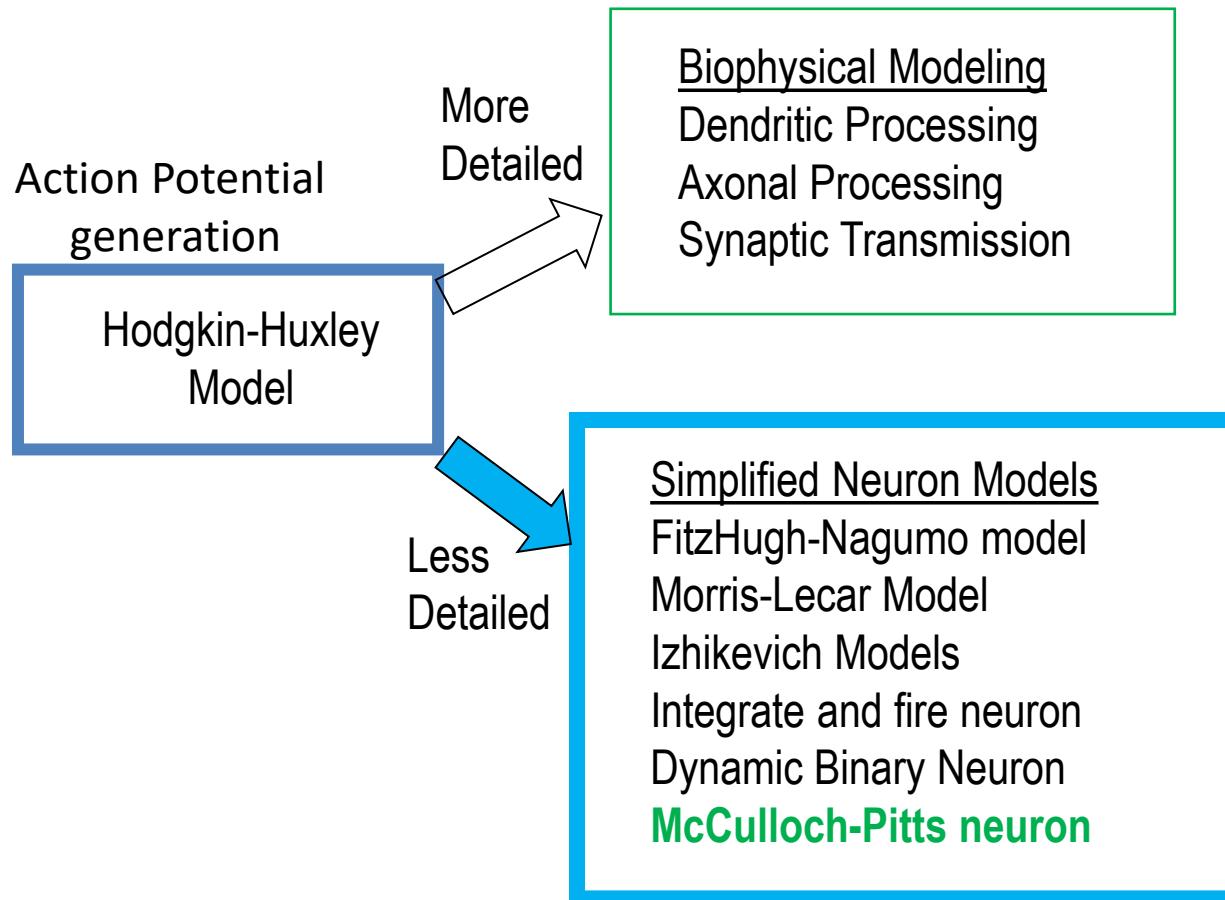
60. Vogels, T.P. *et al.* (2011) Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science* 334, 1569–1573
61. Abbott, L.F. and Nelson, S.B. (2000) Synaptic plasticity: taming the beast. *Nat. Neurosci.* 3, 1178–1183
62. Silberberg, G. and Markram, H. (2007) Disynaptic inhibition between neocortical pyramidal cells mediated by martinotti cells. *Neuron* 53, 735–746
63. Kubota, Y. (2014) Untangling GABAergic wiring in the cortical microcircuit. *Curr. Opin. Neurobiol.* 26, 7–14
64. Leinweber, M. *et al.* (2017) A sensorimotor circuit in mouse cortex for visual flow predictions. *Neuron* 95, 1420–1432.e5
65. Singer, Y. *et al.* (2018) Sensory cortex is optimised for prediction of future input. *eLife* 7, e31557
66. Friston, K. and Herreros, I. (2016) Active inference and learning in the cerebellum. *Neural Comput.* 28, 1812–1839
67. Schultz, W. *et al.* (1997) A neural substrate of prediction and reward. *Science* 275, 1593–1599
68. Scellier, B. and Bengio, Y. (2017) Equivalence of equilibrium propagation and recurrent backpropagation. arXiv preprint arXiv:1711.08416
69. Hopfield, J.J. (1984) Neurons with graded response have collective computational properties like those of 2-state neurons. *Proc. Natl. Acad. Sci. U. S. A.* 81, 3088–3092
70. Friston, K.J. (2005) A theory of cortical responses. *Philos. Trans. R. Soc. B Biol. Sci.* 360, 815–836
71. Bogacz, R. (2017) A tutorial on the free-energy framework for modelling perception and learning. *J. Math. Psychol.* 76, 198–211
72. Pineda, F.J. (1987) Generalization of back-propagation to recurrent neural networks. *Phys. Rev. Lett.* 59, 2229–2232

Perceptrons and Multi layered Perceptrons

Single Neuron Modeling



Single Neuron Modeling

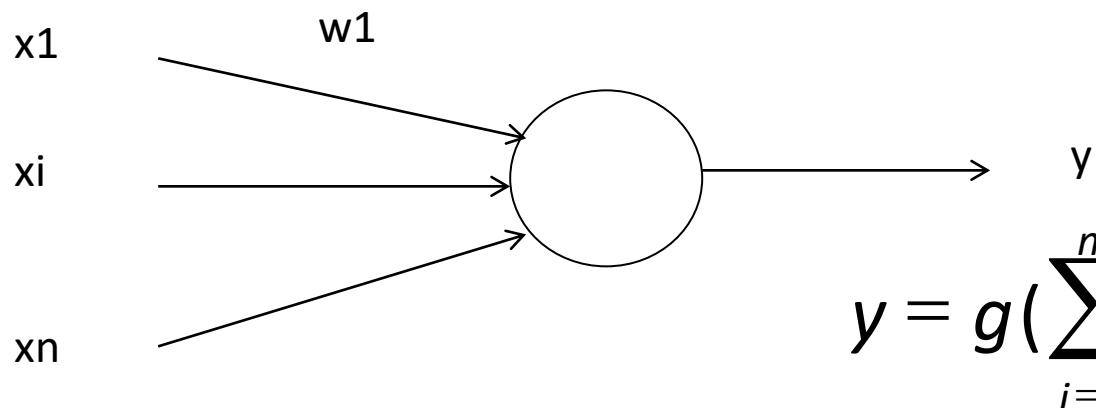


Perceptrons

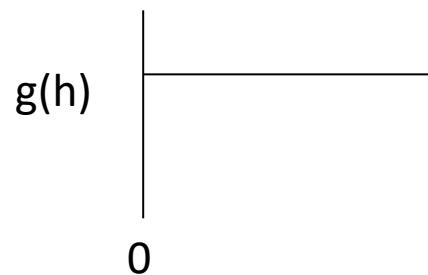
MP neuron

- McCulloch-Pitts neuron model (1943) takes inputs from many neurons and produces a single output.
- If the net effect of the external inputs is greater than a threshold, the neuron goes into excited state (1), else it remains in its resting state (0).

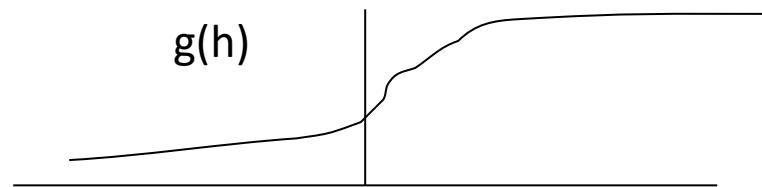
Basic Neuron Model



$$y = g\left(\sum_{i=1}^n w_i x_i - b\right)$$



or



MP neuron model

- McCulloch and Pitts thought that brain works like a computer and neurons are like logic gates
- Since the MP neurons are binary units it seemed worthwhile to check if the basic logical operations can be performed by these neurons.
- McCulloch and Pitts quickly showed that the MP neuron can implement the basic logic gates AND, OR and NOT simply by proper choice of the weights:

A Theory of how the Brain doesn't work

- McCulloch-Pitts neurons can implement logic gates.

- OR Gate: $y = g(x_1 + x_2 - 0.5)$ table:

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	1

- AND Gate: $y = g(x_1 + x_2 - 1.5)$ table:

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

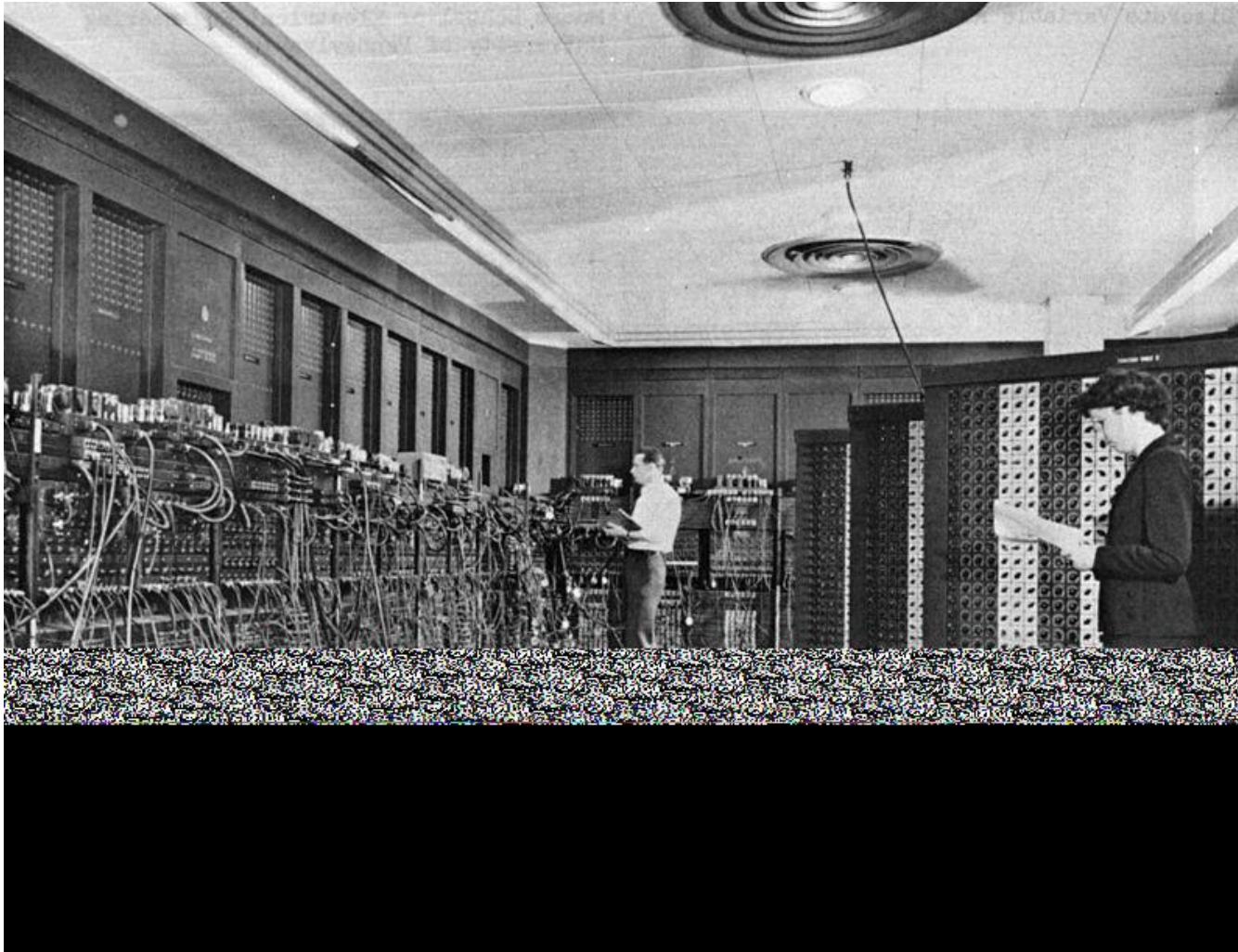
- NOT Gate: $y = g(-x + 0.5)$ table:

X	Y
0	1
1	0

- Hence Brain is a logical machine. (Wrong!)

ENIAC (1945)

(Electronic Numerical Integrator and Computer)



- The idea of considering neurons as logic gates and the brain itself as a large Boolean circuit *does not* satisfy other important requirements of a good theory of the brain.

The Brain and the Computer

- Computer
- Rigid inorganic 2D sheets matter
- Powered by DC mains
- Signals: 5volt DC 
- ✓ Extreme sensitivity to faults
- Clock cycle: $O(10^{-9})$ sec
- Centralized clock
- Power dissipation: 10^{-5} watts

✓ programmed



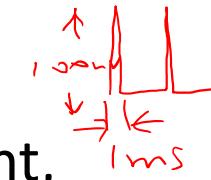
Brain

fault tolerant

Soft organic 3D tissue

Powered by ATP

100 mV pulsed



✓ Very fault tolerant,

1 mS (200-300Hz)

✓ 100 billion neurons

No centralized clock

10^{-12} watts

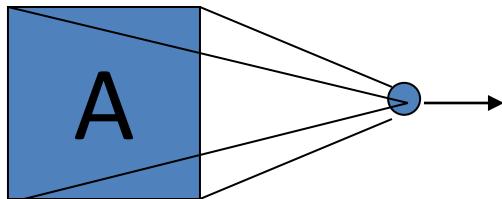
✓ learns from experience
phineas gage

ML

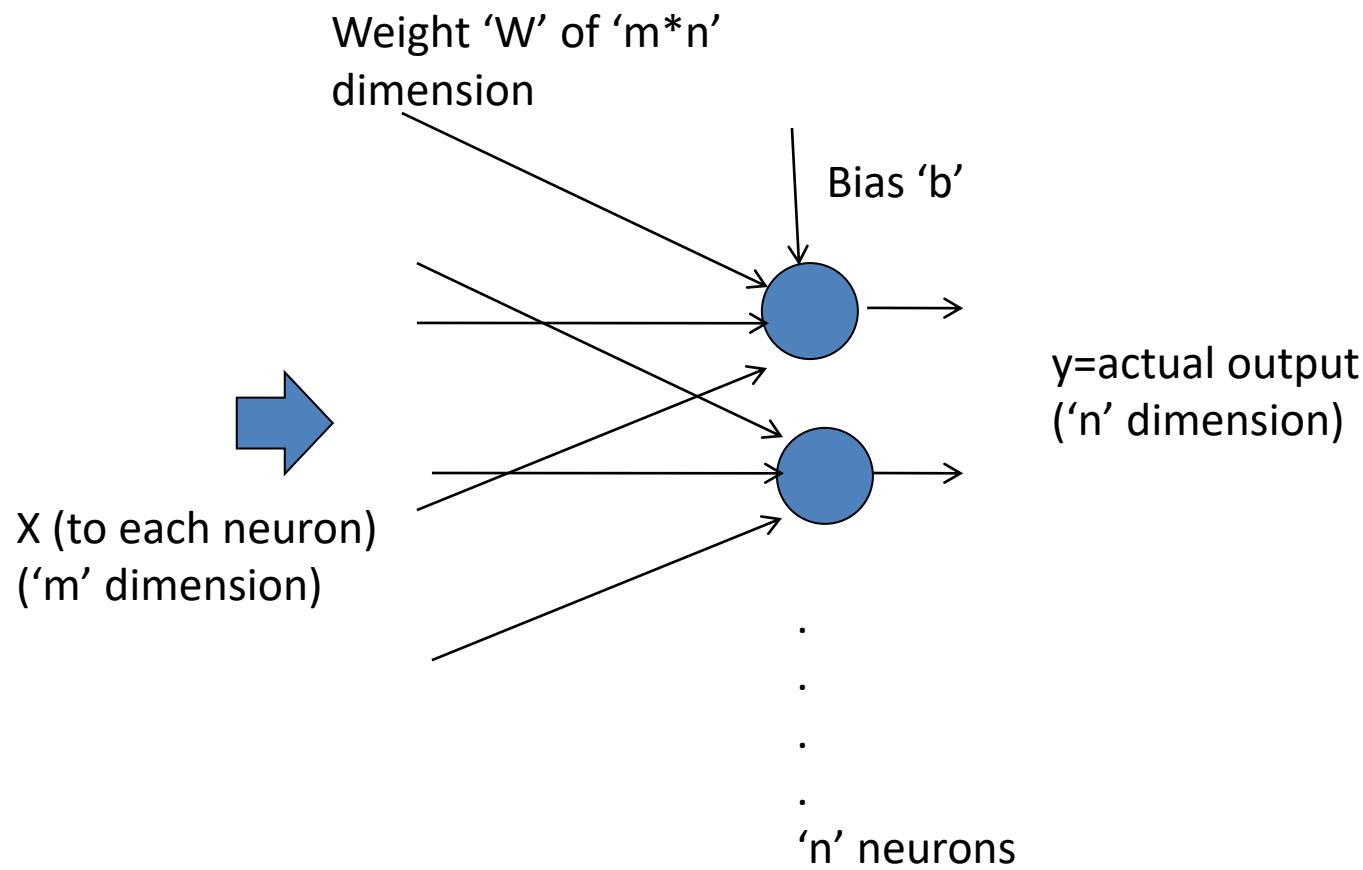
The Perceptron – A NN that learns (1960's)

Frank Rosenblatt

- Learns to recognize simple image patterns
- Uses a single layer of MP neurons



- Limitations:
 - Can classify only “linearly separable” patterns.



MP Neuron divides the input space into two semi-infinite halves

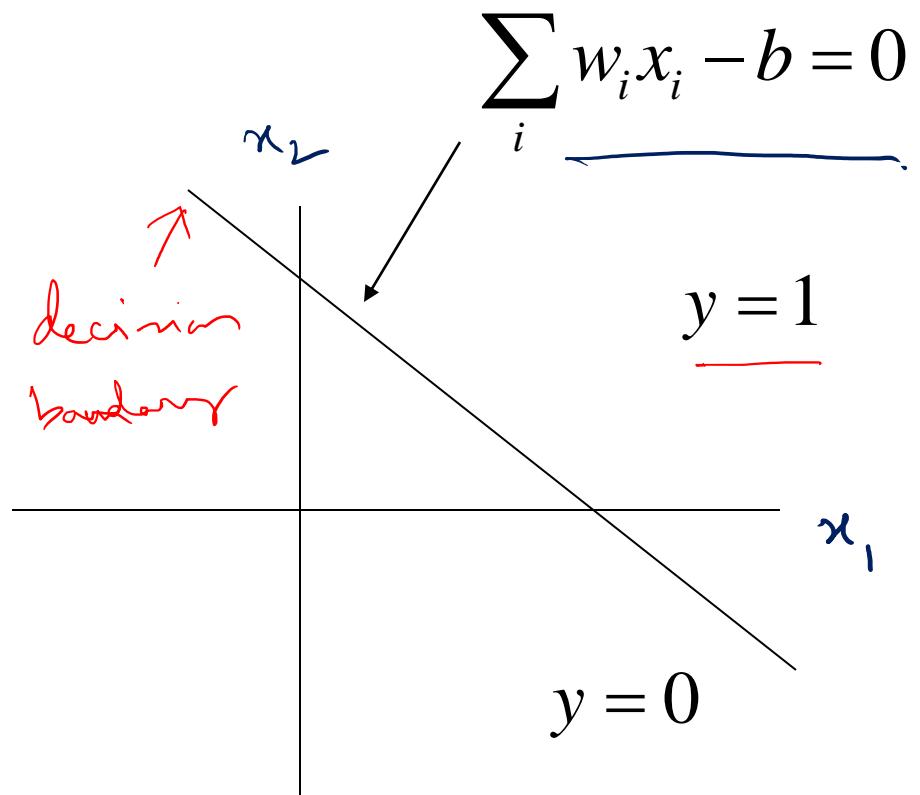
$$y = g\left(\sum_i w_i x_i - b\right)$$

$$\text{net} \equiv \sum_i w_i x_i - b$$

$$\text{net} < 0, y = 0$$

$$\text{net} > 0, y = 1$$

$$x \in \mathbb{R}^n$$



AN EXAMPLE OF HOW A PERCEPTRON LEARNS

Function to be learnt: OR gate

Training
set

X1	X2	d
0	0	0
0	1	1
1	0	1
1	1	1

1.5

Presenting (0,0): wrongly classified

$$\begin{aligned}w_1 &= 0.2 \\w_2 &= 0.3 \\b &= -0.2\end{aligned}$$

1

0.5

0

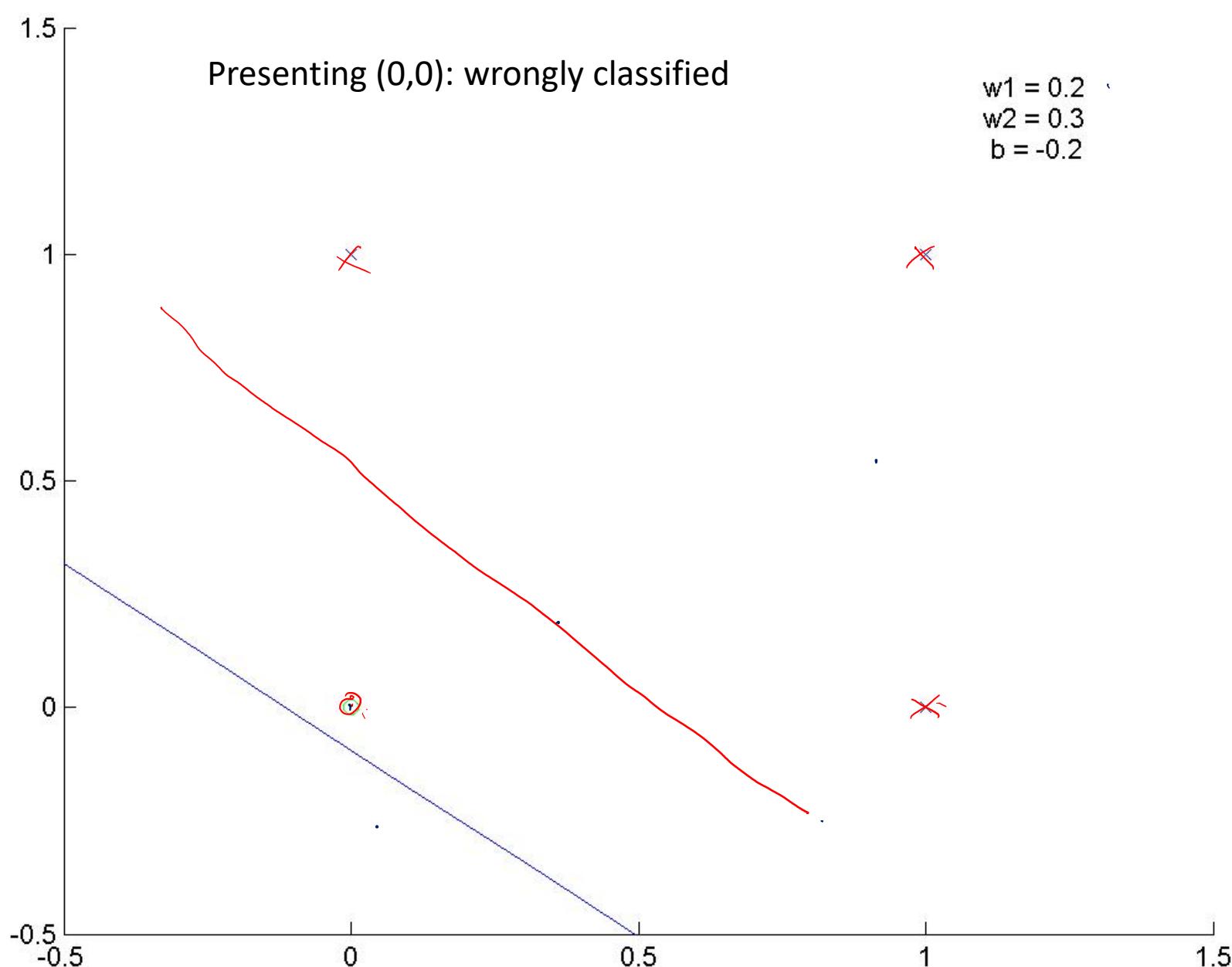
-0.5

0

0.5

1

1.5



1.5

Presenting (0,1): correctly classified

w1 = 0.2
w2 = 0.3
b = -0.0

1

0.5

0

-0.5

x

o

x

x

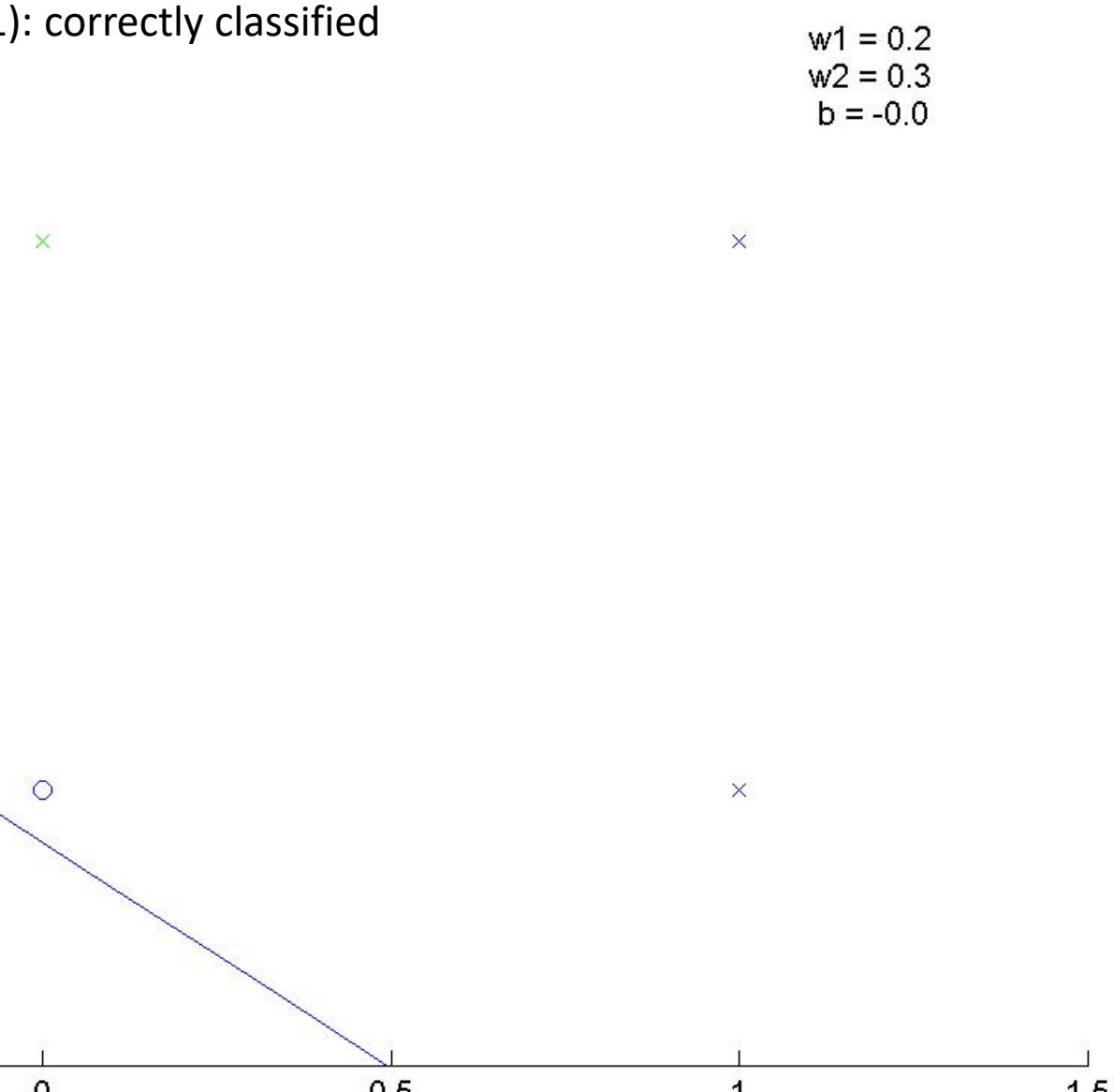
-0.5

0

0.5

1

1.5



1.5

Presenting (1,0): correctly classified

$w_1 = 0.2$
 $w_2 = 0.3$
 $b = -0.0$

1

0.5

0

-0.5

x

x

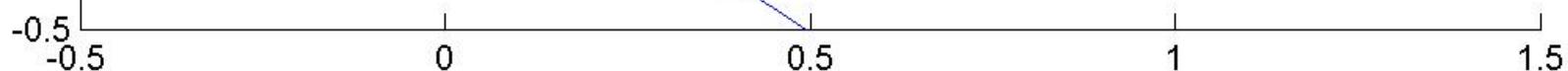
o

x

0

1

1.5



1.5

Presenting (1,1): correctly classified

w1 = 0.2
w2 = 0.3
b = -0.0

1

x

x

0.5

0

o

x

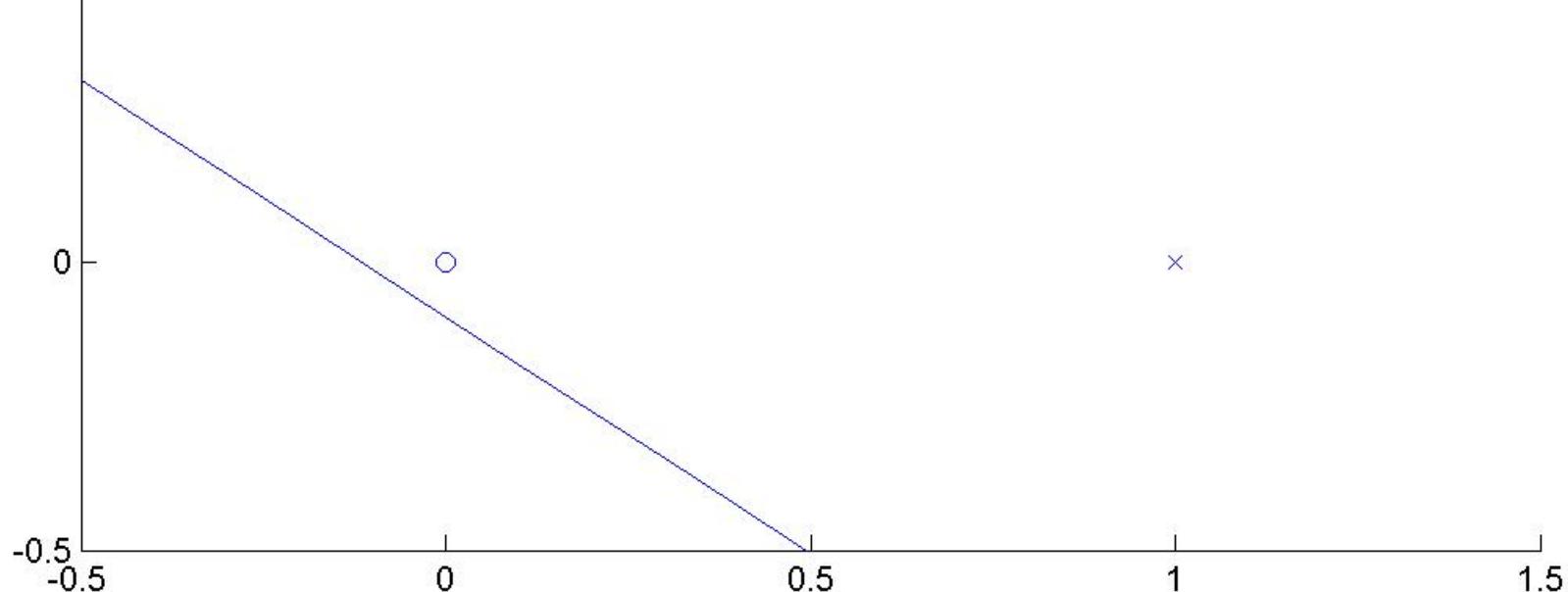
-0.5

0

0.5

1

1.5



3 out of 4 correctly classified

Presenting (0,0): correctly classified

Presenting (0,1): correctly classified

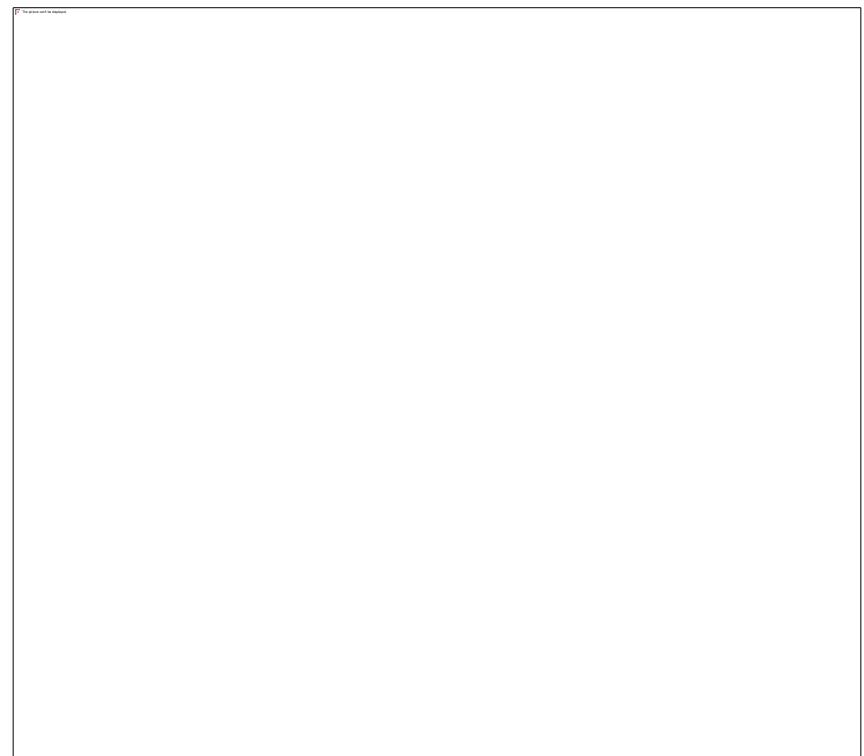
Presenting (1,0): correctly classified

Presenting (1,1): correctly classified

. When the training data is linearly separable, there can be an infinite number of solutions.



- In other words, a Perceptron classifies input patterns by dividing the input space into two semi-infinite regions using a hyperplane.



Perceptron Learning Rule:

- It is also called the LMS Rule or Delta Rule or Widrow-Hoff Rule.
- The steps involved in Perceptron learning are as follows:
 1. Initialization of weights: Set the initial values of the weights to 0. $\mathbf{w}(0) = 0$.
 2. Present the p^{th} training pattern, \mathbf{x} , and calculate the network output, y .

3. Using the desired output, d , for the input pattern x , adjust the weights by a small amount using the following learning rule:

$$w(t+1) = w(t) + \eta[d(t) - y(t)]x(t) \quad \text{--- } ①$$

where,

$$b(t+1) = b(t) - \eta[d(t) - y(t)]$$

$$d(n) = +1, x(t) \in C1$$

$$d(n) = -1, x(t) \in C2$$

--- ②

4. Go back to step 2 and continue until the network output error, $e = d - y$, is 0 for all patterns.

- The above training process will converge after N_{\max} iterations where,

$$\alpha = \min_{\substack{x(n) \in C1 \\ \underline{\text{---}}}} w_0^T x(n)$$

$$\beta = \max_{x(k) \in C1} \|x(k)\|^2$$

$$N_{\max} = \beta \|w_0\|^2 / \alpha$$

- See (Haykin 1999, Chapter 3, Section 3.9) for proof of convergence.

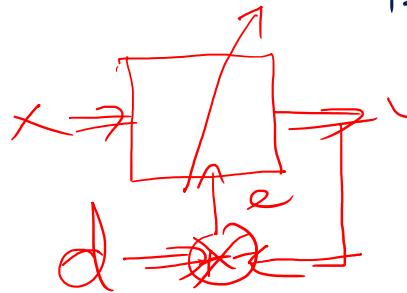
- Range of η : $0 < \underline{\eta} \leq 1$
- Averaging of past inputs leads to stable weight dynamics, which requires small η .
- Fast adaptation requires large η .

- Learning rule can also be derived from an error function:

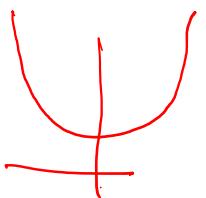
$$\Delta w = -\eta \nabla_w E \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = -[d - y] \frac{\partial y}{\partial w_i} = -[d - y] g' x_i$$

where E denotes the squared error over all patterns. The learning rule may be derived by performing gradient descent over the error function.



$$\frac{1}{1+e^{-\lambda x}}, \lambda > 1$$



$$\Delta w = -\eta \nabla_w e$$

$$\Delta b = -\eta \frac{\partial e}{\partial b}$$

$$y = g(\underbrace{\sum \omega_i x_i - b}_{=h})$$

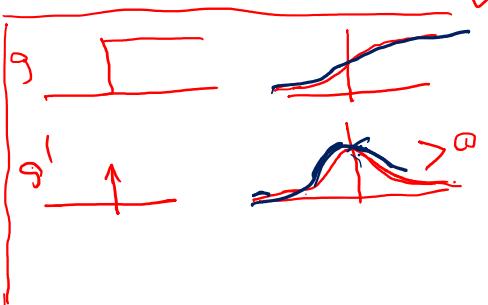
$$\begin{aligned} d &= +1 \quad x \in C_1 \\ &= 0 \quad x \in C_2 \end{aligned}$$

$$E = \frac{1}{2} \sum_p (d_p - y_p)^2$$

$$e_p = \frac{1}{2} (d_p - y_p)^2$$

$$E = \sum_p e_p$$

1



$$\frac{\partial E}{\partial \omega_i} = \frac{1}{2} \cdot 2 \cdot (d - y) (-1)$$

$$\begin{aligned} &g'(x_i) \\ &= -(d - y) g'(x_i) \end{aligned}$$

$$\checkmark \Delta \omega_i = \eta (d - y) g'(x_i)$$

$$\frac{\partial E}{\partial b} = \frac{-1}{2} \cdot 2 (d - y) g'(x_i) (-1)$$

$$\Delta b = -\eta (d - y) g'(h)$$

$$n' = n' g'(h)$$

$$\left\{ \begin{array}{l} \Delta \omega_i = n' (d - y) x_i \\ \Delta b = -n' (d - y) \end{array} \right.$$

- The last term in the above equation has g' , which is zero everywhere except at the origin if g is a hardlimiting nonlinearity.
- But if we take a smoother version of $g()$, which saturate at +1 and -1, like the $\tanh()$ function, the learning rule becomes,

$$\Delta w_i = \eta [d - y] g' x_i$$

- Since $g' > 0$ always for $\tanh()$ function, we can absorb it into h , considering it as a quantity that varies with x . We then have,

$$\Delta w_i = \eta' [d - y] x_i$$

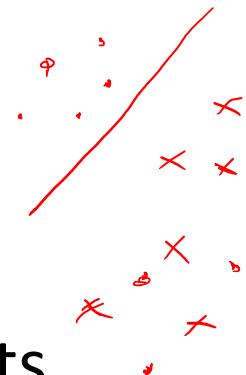
$$\Delta b = -\eta' [d - y]$$

Perceptron Learning
Rule

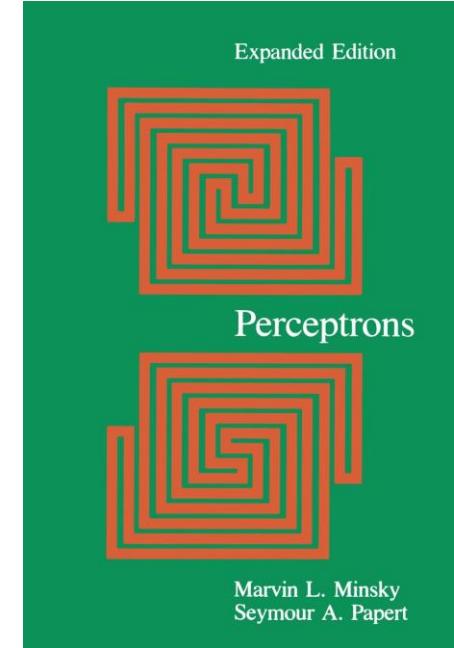
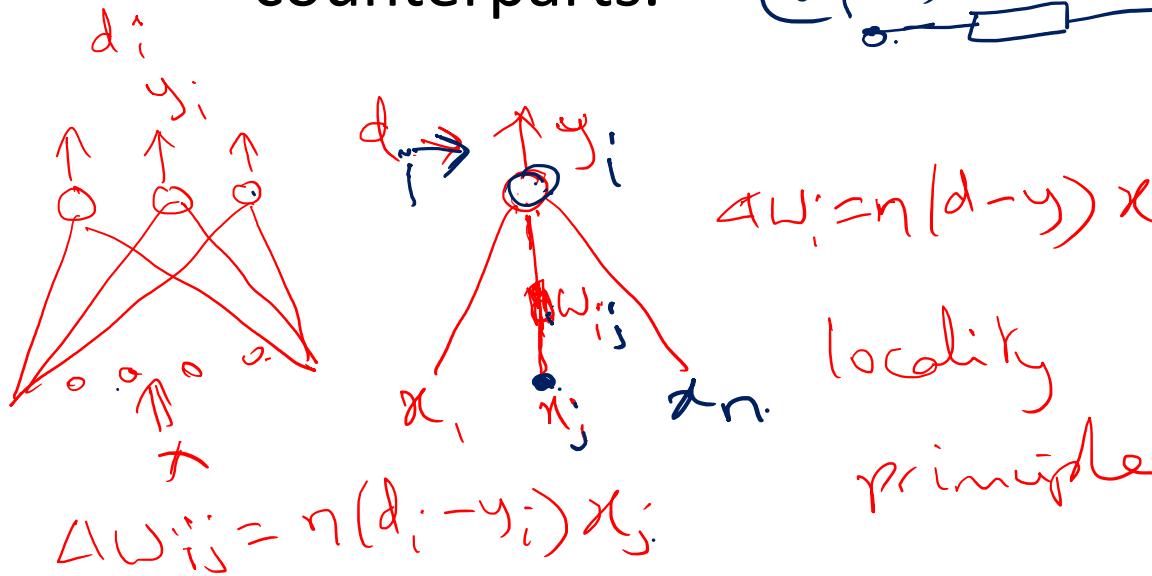
- Which is identical to the Perceptron learning rule given before above.

A not-very-constructive Criticism

- Minsky and Papert on Perceptron:
 - A single layer perceptron is so limited in its capabilities
 - It may not be worth studying its multilayer counterparts.



$$(d_i - y_i) \vec{w}_{ij} \leftarrow x_j$$



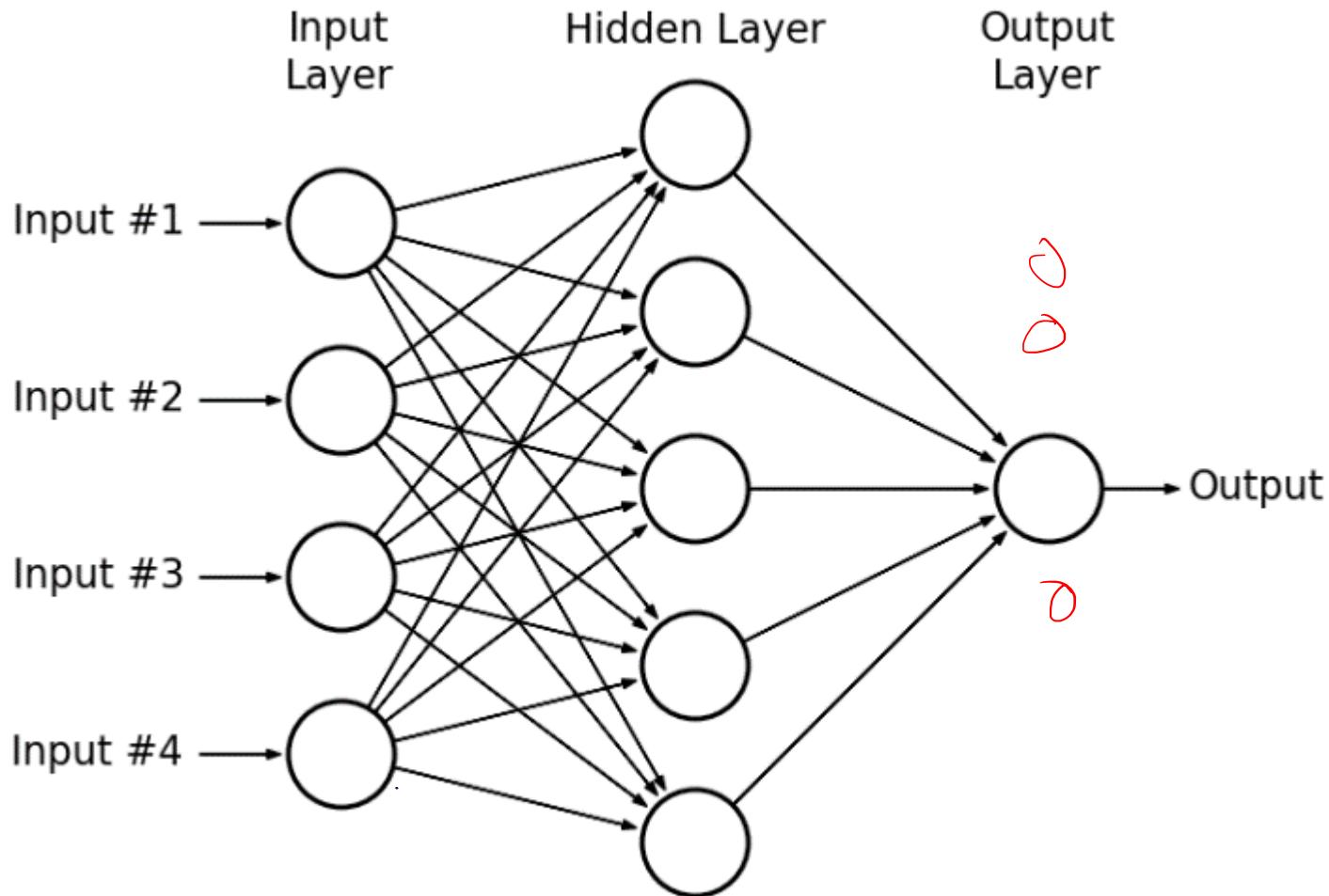
The Multi-layered Perceptron

- Improvements over Perceptron:

- Smooth nonlinearity – sigmoid

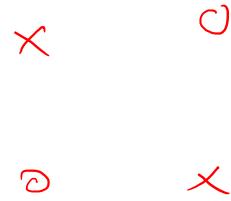
~~logistic~~ ($1/\alpha$)
tanh ($+1/-1$)

- 1 or more hidden layers



The Hidden Layer

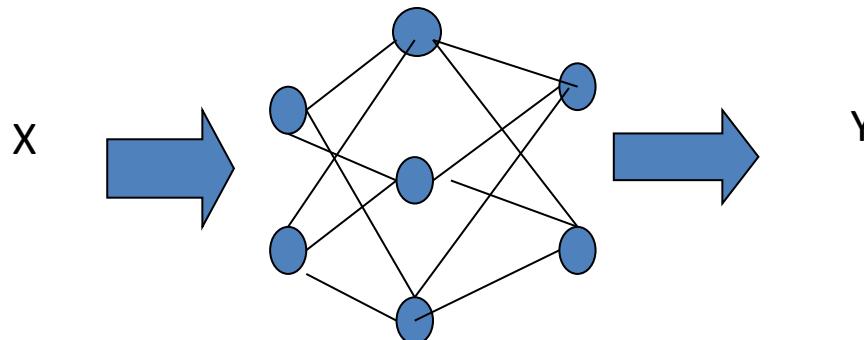
- The perceptron, which has no hidden layers, can classify only linearly separable patterns.
- The MLP, with at least 1 hidden layer can classify any linearly non-separable classes also.



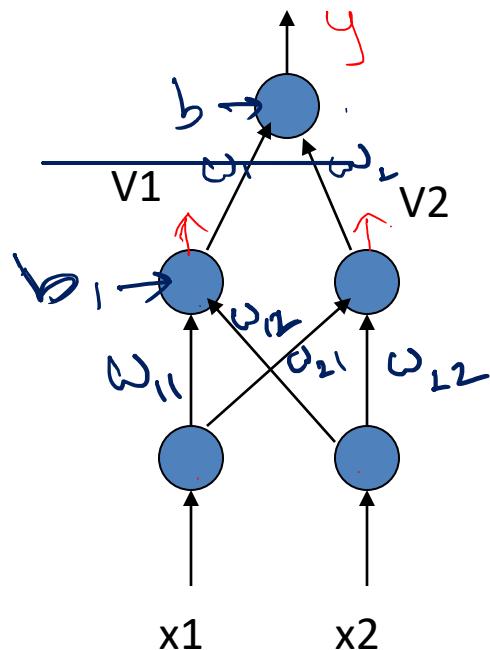
- It can easily be shown that the XOR problem which was not solvable by a Perceptron can be solved by a MLP with a single hidden layer containing two neurons.

Multi Layer Perceptron (MLP)

- Neurons are organized as layers
 - Input, output and hidden layer(s)
- Feedforward NN: Flow of info from input to output layer



MLP solves the EXOR problem



$$\overline{\omega_{11}} = \overline{\omega_{12}} = 1 \quad b_1$$

$g \equiv \text{step func}$

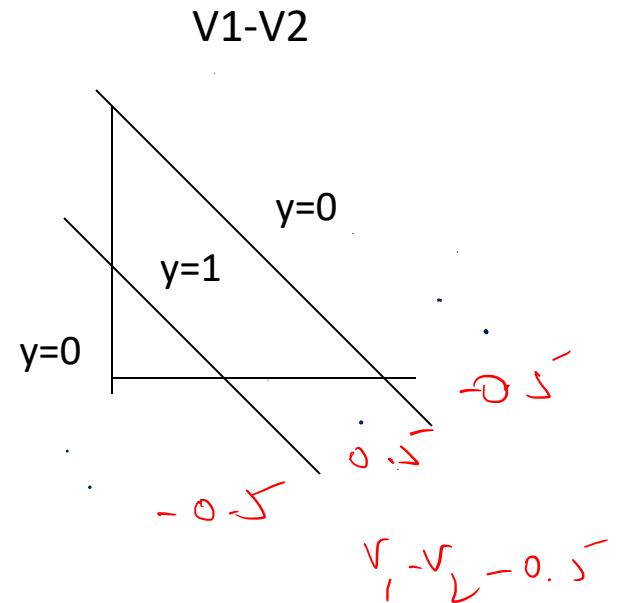
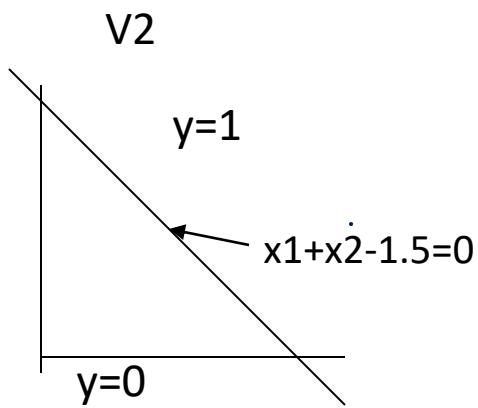
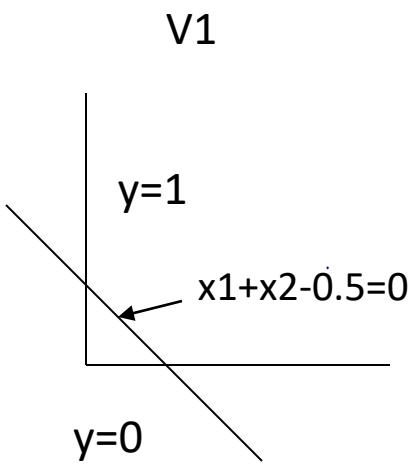
$$V_1 = g(x_1 + x_2 - 0.5)$$

$$V_2 = g(x_1 + x_2 - 1.5)$$

$$\omega_1 \quad \omega_2 \quad b$$

$$y = g(V_1 - V_2 - 0.5)$$

MLP can model an XOR gate



$$y = g(V_1 - V_2 - 0.5)$$

XOR gate!

MLP

- An MLP can approximate any continuous multivariate function to any degree of accuracy, provided there are sufficiently many hidden neurons (Cybenko, 1988; Hornik et al, 1989). A more precise formulation is given below.
- A serious limitation disappears suddenly by adding a single hidden layer.

All-powerful MLP

- MLP is a *universal approximator*
- “An MLP with a single hidden layer can approximate any continuous I/O function with arbitrary accuracy, over a finite input domain, given enough number of nodes in the hidden layer.” (Cybenko, 1988)

Taylor series

$$f(x_0 + h) = \underbrace{f(x_0)}_{\text{constant}} + h f'(x_0) +$$

Approximation theory.

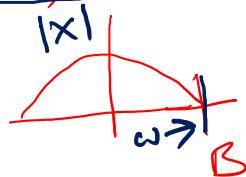
$$x(t) = \sum w_i \phi_i(t)$$

Fourier series

$$s(t) = s(t+T) \quad \forall t$$

$$s(t) = a_0 + \sum_{n=1}^{\infty} c_n \cos(n\omega_0 t + \phi_n)$$

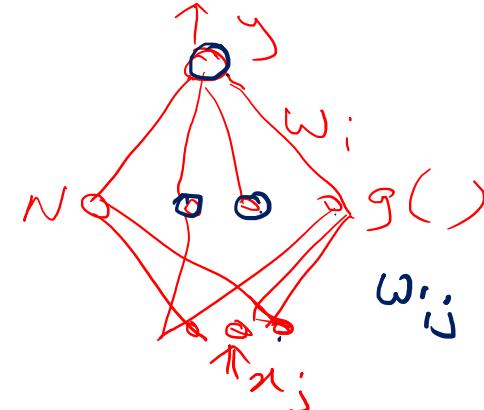
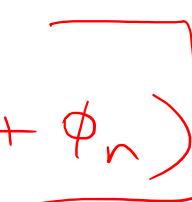
Sampling theorem



$$s \cdot f_r > 2B$$

$$\phi(x) = \frac{\sin x}{x}$$

$$s(t) = \sum w_i \phi_i(x)$$



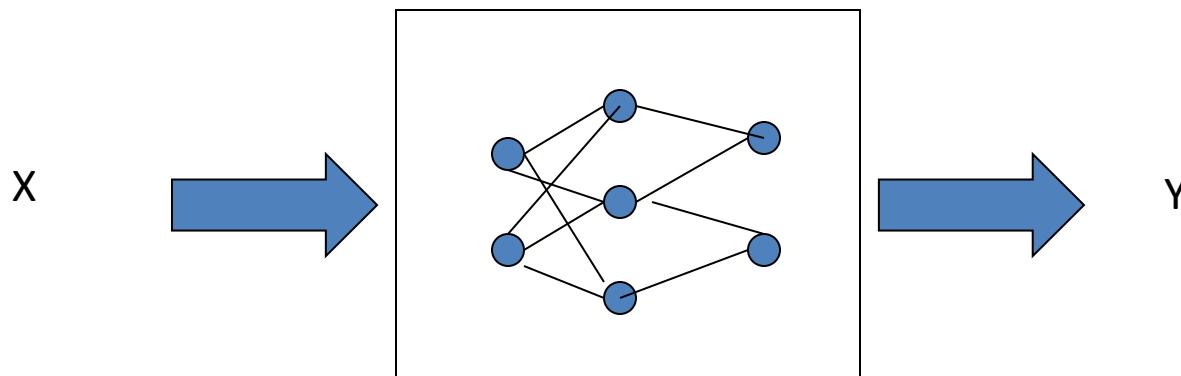
$$y = \sum_i w_i g(\sum_j w_{ij} x_j - b_i)$$

$$= \sum_{i=1}^N w_i g(w_i \cdot x - b_i)$$

$$x \rightarrow y \\ \in \mathbb{R}^n \quad \in \mathbb{R}$$

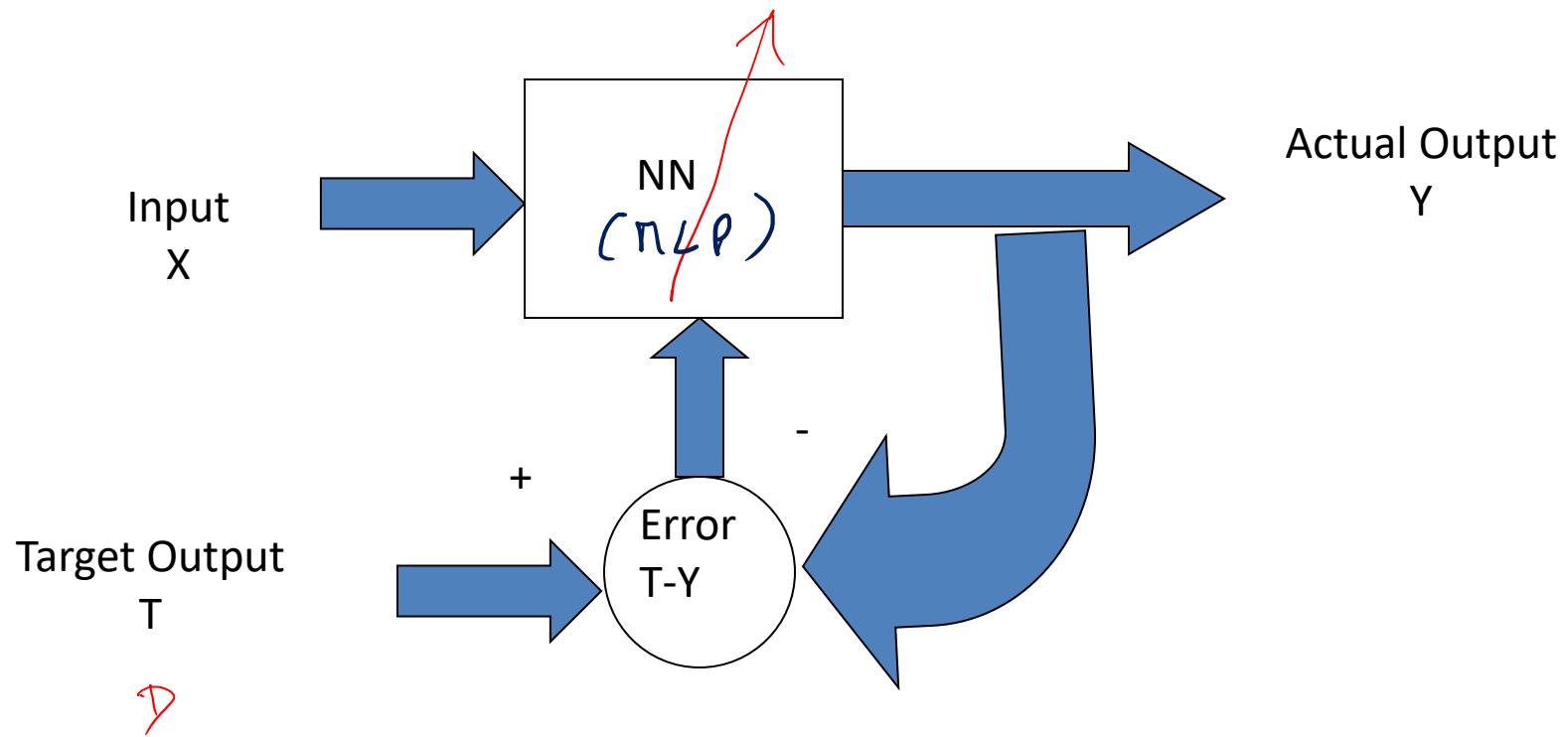
NN as an I/O Machine

- NN is a parametric model that can model arbitrary I/O relationships



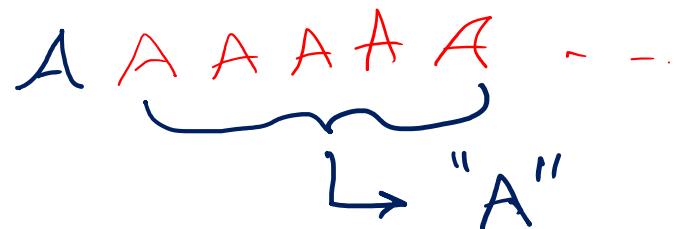
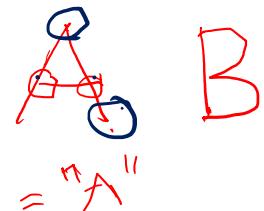
Learns from Experience

- NN can learn the unknown I/O relationship from sample data



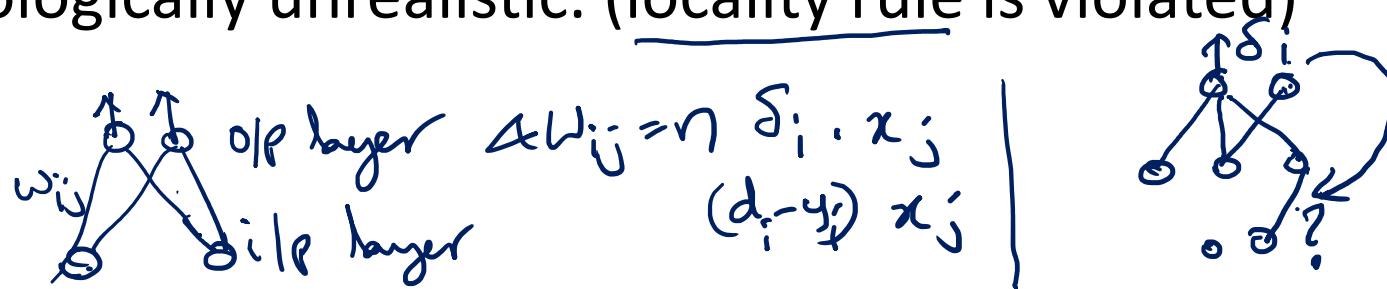
NN vs. Algorithmic Approach

- Traditional or Algorithmic Approach
 - Study and find out the steps to produce a final output from a given input
- NN Approach:
 - Learn from Experience



Training the hidden layer

- Not obvious how to train the hidden layer parameters.
- The error term is meaningful only to the weights connected to the output layer – credit assignment problem.
- In a large network with many layers, this implies that information is exchanged over distant elements of the network though they are not directly connected.
- Such an algorithm may be mathematically valid, but is biologically unrealistic. (locality rule is violated)



The Backpropagation Algorithm

- **History:**
- First described by Paul Werbos (1974) in his PhD thesis at MIT.
- Rediscovered by Rumelhart, McClelland and Williams (1986)
- Also discovered by Parker (1985) and LeCun (1985) in the same year.
- As in Perceptron, this training algorithm involves 2 passes:
 - The forward pass – outputs of various layers are computed
 - The backward pass – weight corrections are computed

Back Propagation Algorithm

- Consider a simple 3-layer network with a single neuron in each layer.

Total output error over all patterns: $E = \sum_p E_p$

Squared Output error for the p'th pattern: $E_p = \frac{1}{2} \sum_i e_i^2$

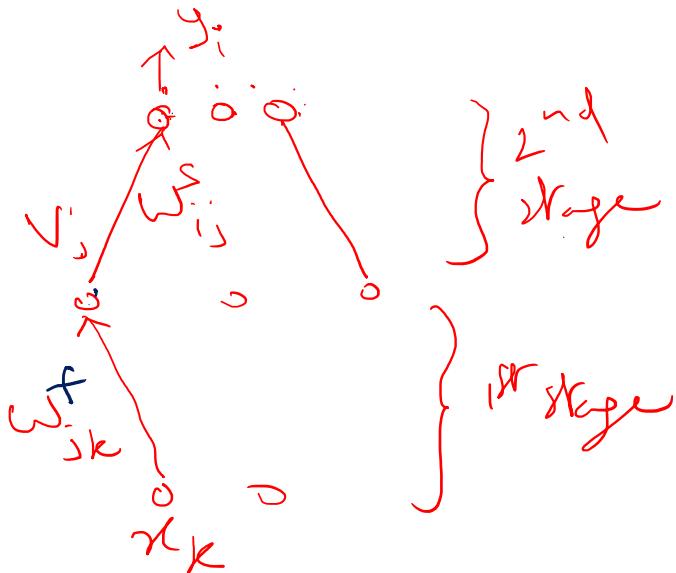
Output error for the p'th pattern: $e_i = d_i - y_i$

Network output: $y_i = g(h_i^s)$

Net input to the output layer: $h_i^s = \sum_j w_{ij}^s V_j - \theta_i^s$

Output of the hidden layer: $V_j^f = g(h_j^f)$

Net input of the hidden layer: $h_j^f = \sum_k w_{jk}^f x_k - \theta_j^f$



$$E = \sum_p E_p$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{1}{2} \sum_i^n (d_i - y_i)^2$$

$$\Delta w_{ij}^s = ?$$

$$\Delta w_{jk}^f = ?$$

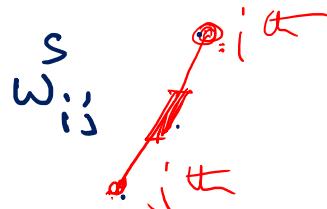
G.D

$$\Delta w = -\eta \frac{\partial E}{\partial w}$$

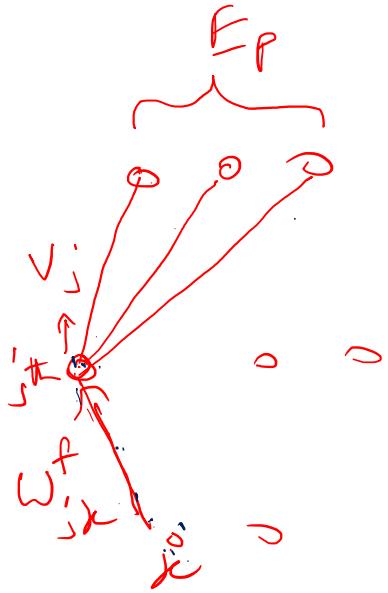
$$\Delta w_{ij}^s = -\eta \frac{\partial E_p}{\partial w_{ij}^s}$$

$$\frac{\partial E_p}{\partial w_{ij}^s} = \frac{1}{2} \cdot 2 \cdot (d_i - y_i) (-1) g'(h_i^s) v_j$$

$$\Delta w_{ij}^s = \eta (d_i - y_i) g'(h_i^s) v_j \quad \checkmark$$



locality principle



$$\Delta w_{jk}^f = -\eta \frac{\partial E_p}{\partial w_{jk}^f}$$

$$\frac{\partial E_p}{\partial w_{jk}^f} = \frac{1}{2} (2) \sum_i (d_i - y_i) (+) g'(h_i^f) w_{i,j}^s \frac{\partial v_j}{\partial w_{jk}^f}$$

$$\left(\frac{\partial v_j}{\partial w_{jk}^f} = g'(h_j^f) x_k \right)$$

$$= \sum_i (d_i - y_i) (-) g'(h_i^f) w_{i,j}^s \\ g'(h_j^f) x_k$$

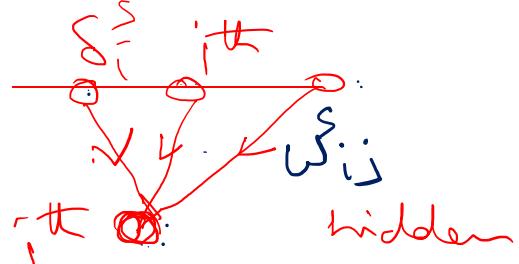
violates locality principle

Define δ 's (error terms)

~~Output layer~~

$$\boxed{\Delta w_{ij}^s = \eta \delta_i^s y_j} \quad \checkmark$$

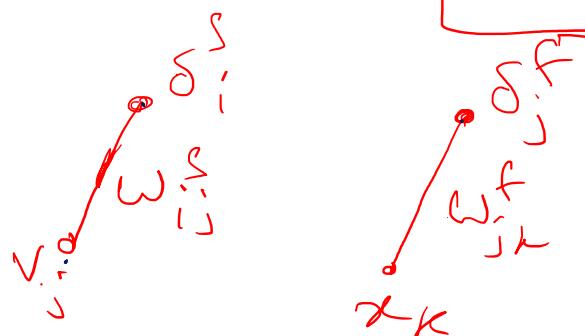
$$\delta_i^s = (d_i - y_i) g'(h_i^s) \quad \longrightarrow \quad \Delta w_{jk}^f = \eta \sum_i (d_i - y_i) g'(h_i^s)$$



$$w_{ij}^s g'(h_j^s) x_k$$

$$= \eta \left(\sum_i \delta_i^s w_{ij}^s \right) g'(h_j^s) x_k$$

$$\boxed{= \eta \delta_j^f x_k} \quad \checkmark$$



$$\delta_j^f = \left(\sum_i \delta_i^s w_{ij}^s \right) g'(h_j^f)$$

BPA

- Update rule for the weights using gradient descent:

$$\Delta w_{ij}^s = -\eta \frac{\partial E_p}{\partial w_{ij}^s}; \quad \Delta \theta_i^s = -\eta \frac{\partial E_p}{\partial \theta_i^s}$$

$$\Delta w_{jk}^f = -\eta \frac{\partial E_p}{\partial w_{jk}^f}; \quad \Delta \theta_j^f = -\eta \frac{\partial E_p}{\partial \theta_j^f}$$

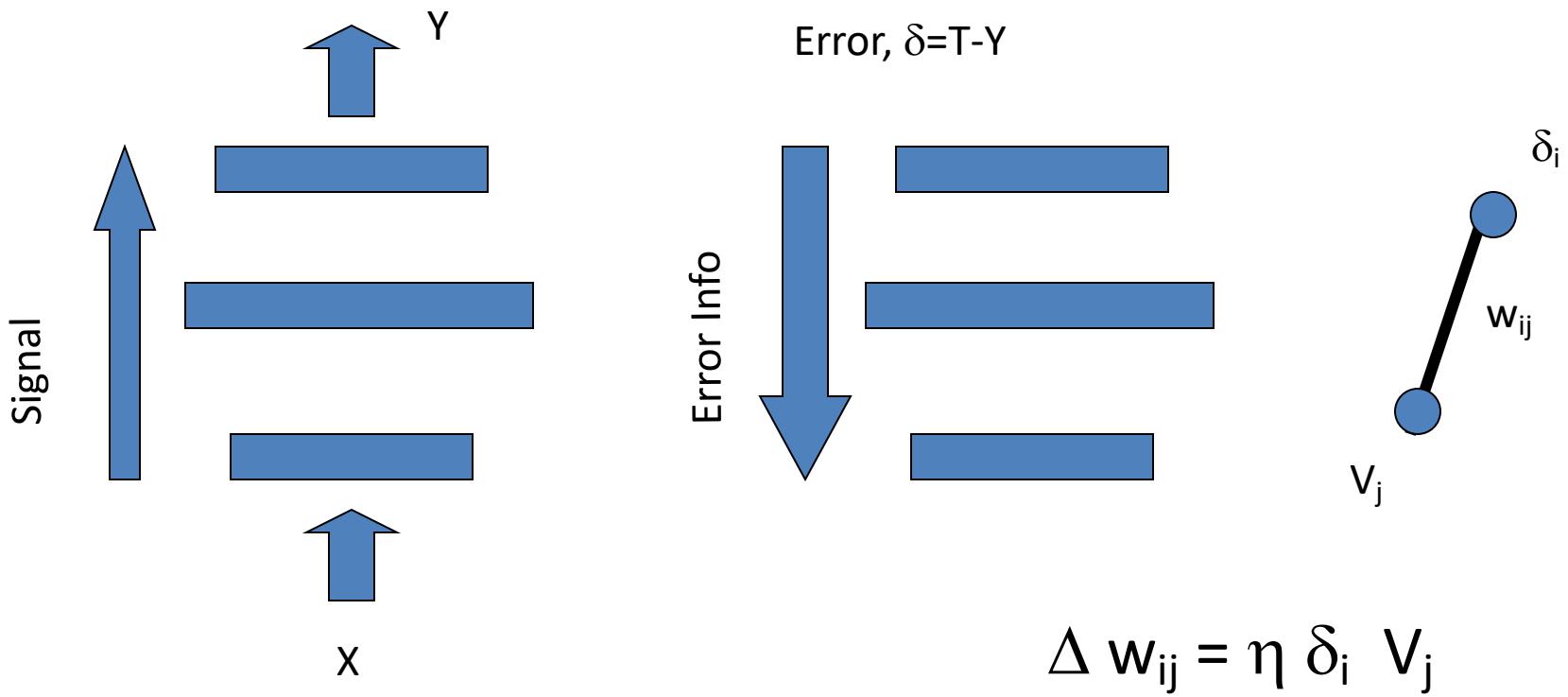
Updating w_{ij}^s :

$$\begin{aligned}\frac{\partial E_p}{\partial w_{ij}^s} &= \frac{\partial E_p}{\partial e_i} \frac{\partial e_i}{\partial y_i} \frac{\partial y_i}{\partial h_i^s} \frac{\partial h_i^s}{\partial w_{ij}^s} \\ &= e(-1) g'(h_i^s) V_j\end{aligned}$$

- **weight correction = (learning rate) * (local δ from ‘top’) * (activation from ‘bottom’)**

Training MLP

- Backpropagation



Training and Testing a MLP

Traning:

- Randomly initialize weights.
- Train network using backprop eqns.
- Stop training when error is sufficiently low and freeze the weights.

Testing

- Start using the network.

Merits of MLP

Merits of MLP trained by BP:

- A general solution to a large class of problems.
- With sufficient number of hidden layer nodes, MLP can approximate arbitrary target functions.
- Backprop applies for arbitrary number of layers, partial connectivity (no loops).
- Training is local both in time and space – parallel implementation made easy.
- Hidden units act as “feature detectors.”
- Good when no model is available

Demerits of MLP by BPA

Problems with MLP trained by BP:

- Blackbox approach
- Limits of generalization not clear
- Hard to incorporate prior knowledge of the model into the network
- slow training
- local minima

Development of an NN Application

- The Stages
 - Feasibility study
 - Data collection
 - Designing/Training the Network
 - Testing the Network
 - Deployment...

Feasibility Study

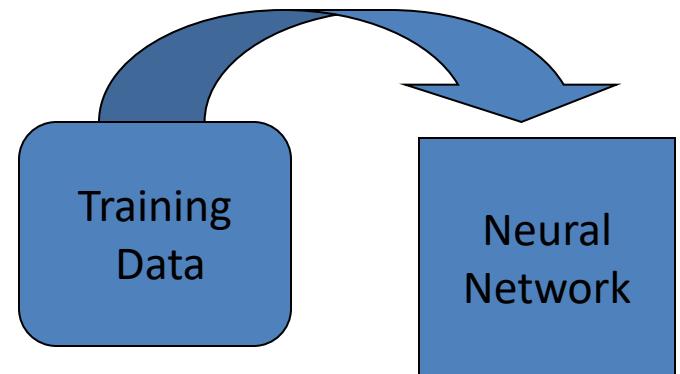
- Do you really need a NN?
- Is it an I/O problem?
- What are the inputs and the outputs?
- Are inputs and outputs uniquely related?
- Do I have enough domain knowledge?
- Do I have enough Data?

Domain Knowledge

- Can never overemphasize the value of Domain Knowledge
- Domain Knowledge is used to -
 - Breaking up the problem
 - Map problem onto NN(s)
 - Data collection
 - Data preprocessing

Data

- Knowledge in Data ==> Knowledge in NN
- Data must be-
 - Abundant
 - real data
 - simulated data
 - Comprehensive
 - include all possible cases
 - Reliable
 - **Right representation** is the key



Data Preprocessing

- Representation
 - Ex: 1) odd/even discr., 2) linkage problem
 - Remove irrelevant or redundant information
 - Normalization, scaling,...
 - Reduce input dimension
 - Principal component analysis
 - Feature extraction

Designing/Training the Network

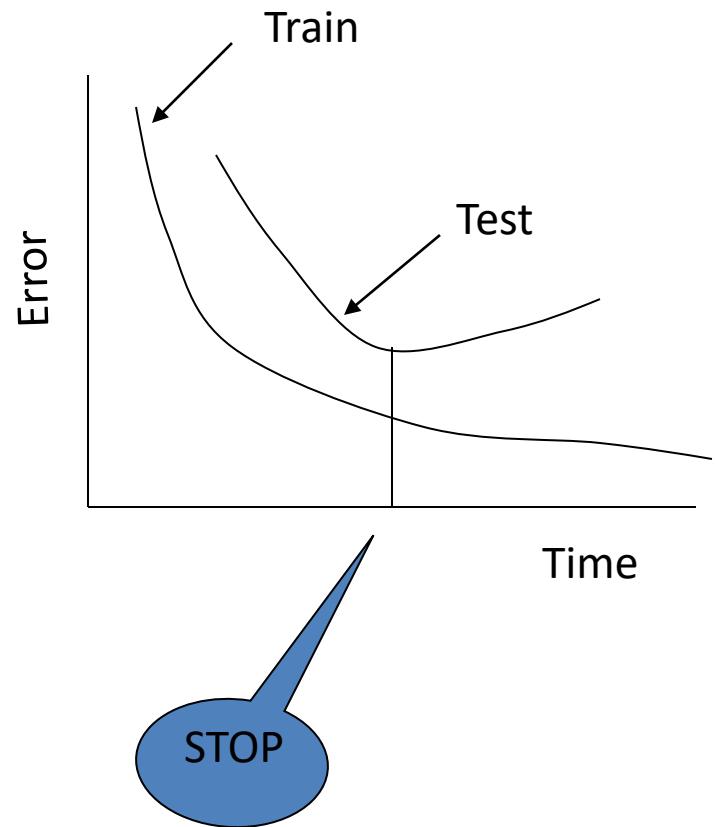
- Architecture
- Initialization
- Training

Architecture

- Feedforward Network: MLP, RBF Network, Hybrid,...?
- MLP:
 - 1 hidden layer?
 - 2 hidden layers?
 - hidden layers > 2?
 - Customized architecture?

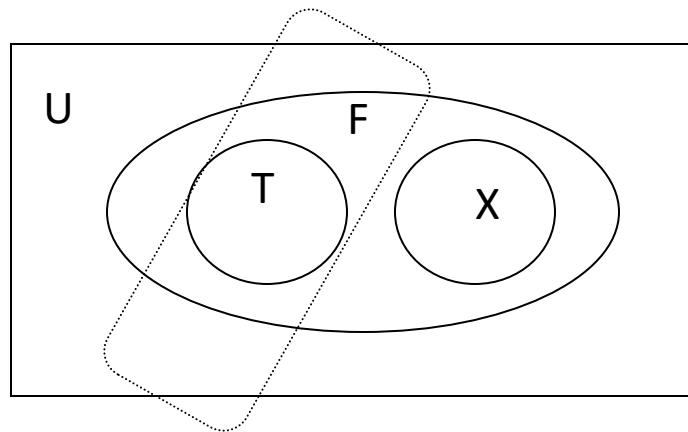
Training & Pruning

- Training procedure
 - Backpropagation
 - adaptive learning rate
 - momentum,...
 - Higher-order methods
- When to stop training?



Generalization & Testing

- The problem of generalization



- Cross-validation

Caution: Specify explicitly the conditions under which the network was trained

Pruning

- weight decay
- Optimal Brain Damage (OBD)

APPLICATIONS

Past tense learning

Ref: Rumelhart and McClelland, On learning
the past tenses of English verbs, Parallel
Distributed Processing, Vol. II, 1986

Three stages of past tense learning is seen in children.

Stage 1:- Only a small number of words used correctly in past tense

- High frequency words, majority are irregular
- Children tend to get the past tense quickly.
- Typical examples: Came, got, gave, looked, needed, took, went.

regular: → ed ending

Stage 2:-

Children use much larger number of words

Many verbs are used with correct past tense forms

Majority are regular

Eg. Wiped, pulled

Children now incorrectly apply regular past tense endings for words which they used in stage 1

Eg: come comed or camed

Stage 3:-

Regular or irregular forms exist.

Children have acquired the use of correct irregular form of past tense

But continue to apply the regular form to new words they learn.

Training

10- high frequency verbs (8 irregular + 2 regular): Eg: Came, get, give,, take,
go, have, feel, look, live

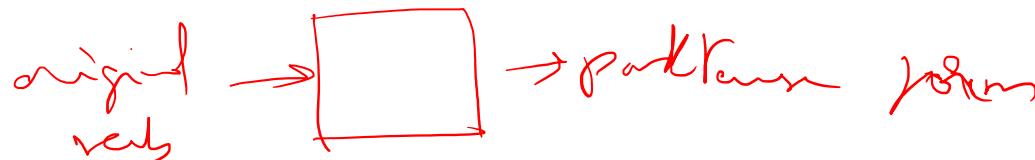
410- medium frequency verbs, 334 regular, 76 irregular

86-low frequency words, 72 regular, 14 irregular

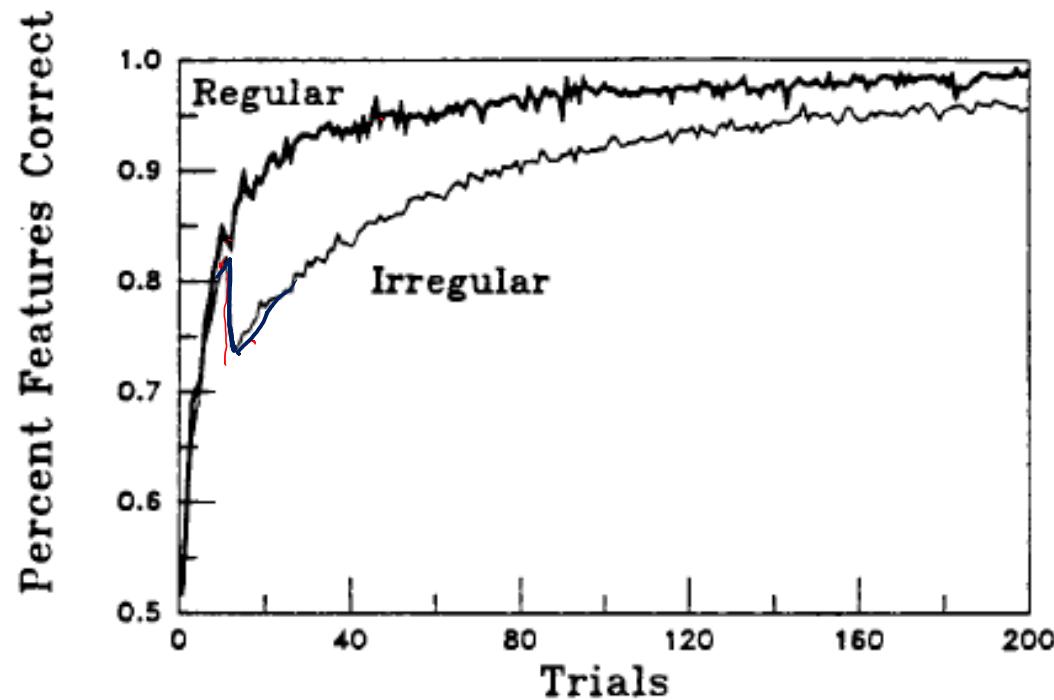
Stage 1:- 10 epochs of high frequency verbs. (Enough for good performance)

*Stage 2:- 410 medium frequency verbs were added to 10 verbs (trained for
190 more epochs after phase 2). In this stage errors suddenly start
creeping in. Most of the errors are due to regularization of irregular verbs*

*Stage 3:- 86 low frequency verbs are tested without training. Beyond state 2,
the performance over regulars and irregulars is nearly the same, each
touching 100%.*



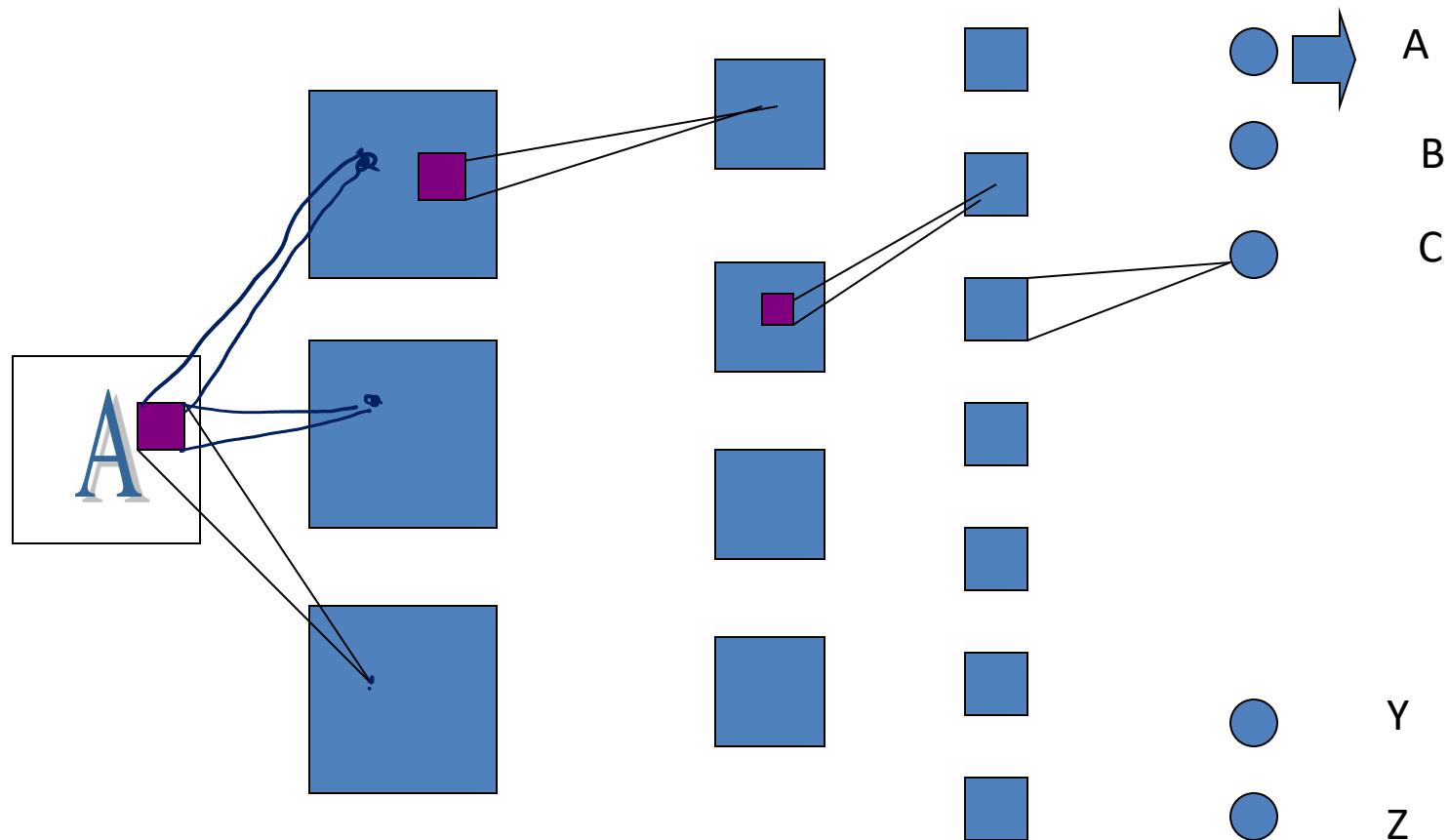
Performance



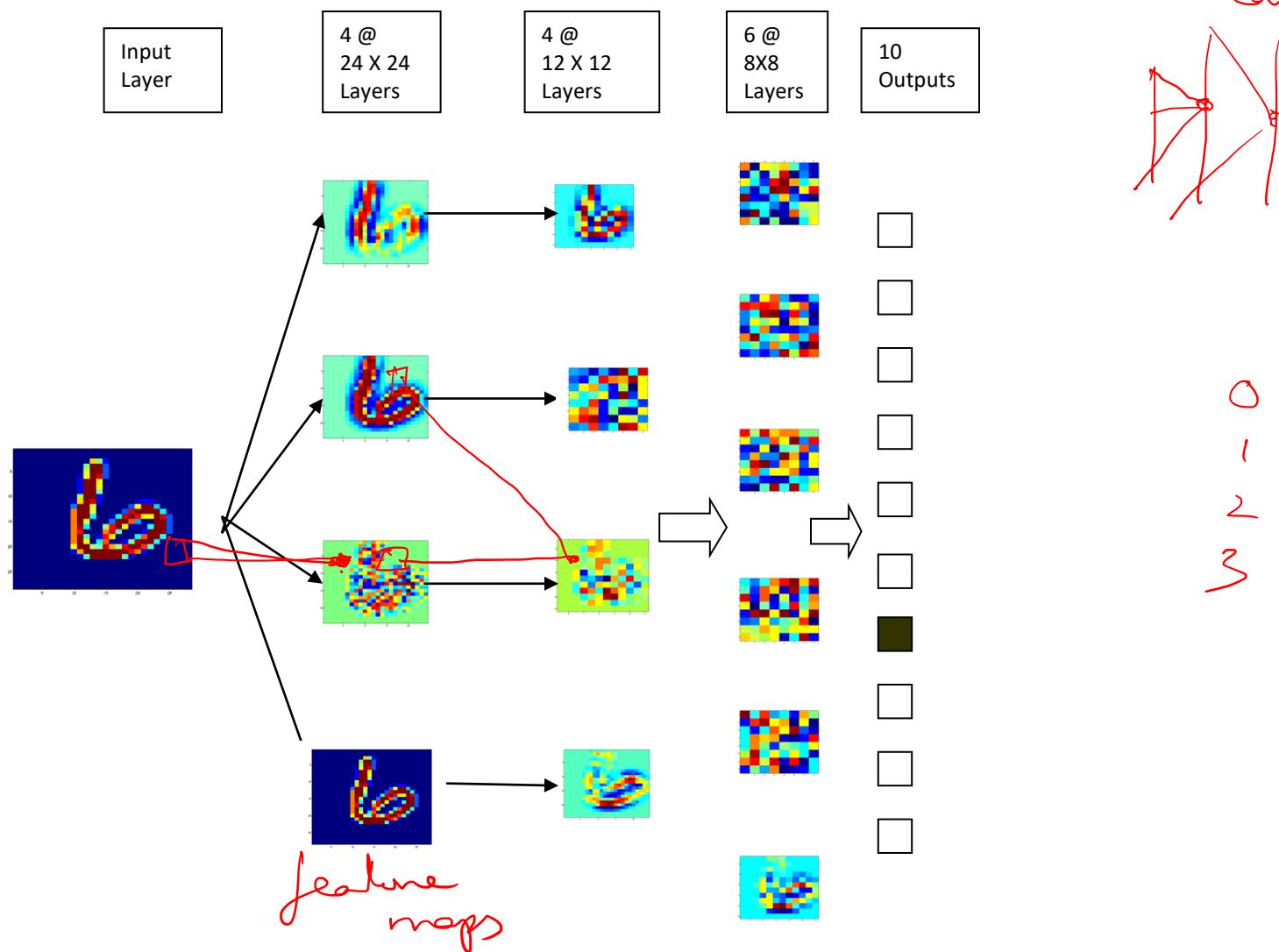
Handwritten Numeral Recognition using Convolutional Networks

- Lenet: Developed at AT & T to read zip codes on US postal envelopes
- Inspired by biological visual system
- Speed: 10 chars/sec to 1000 char/sec
- 99% accuracy on test data at a rejection rate of 12%.

Convolutional Neural Network for Character Recognition



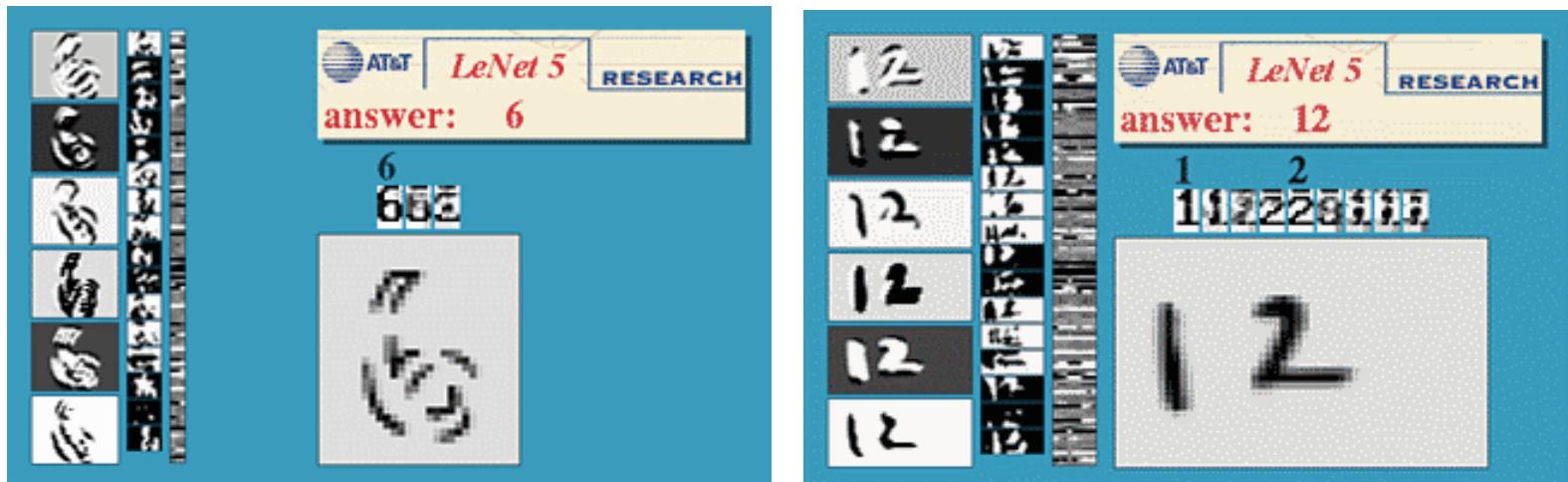
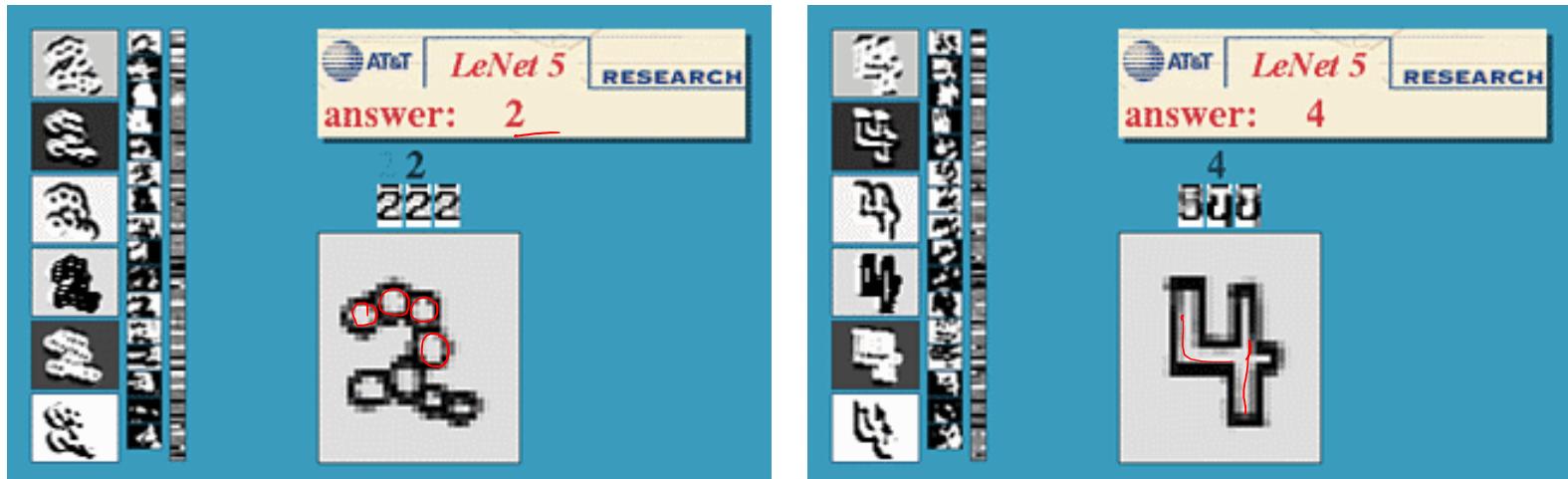
NEURAL NETWORK RECOGNIZES NUMERALS

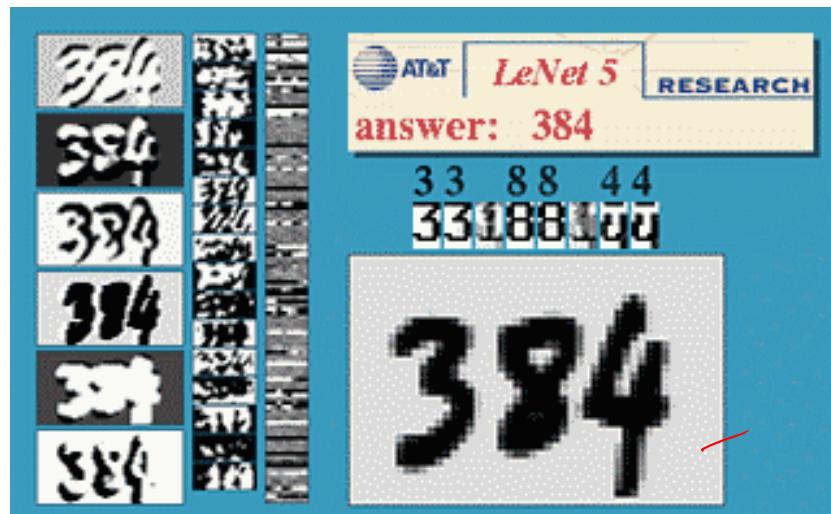
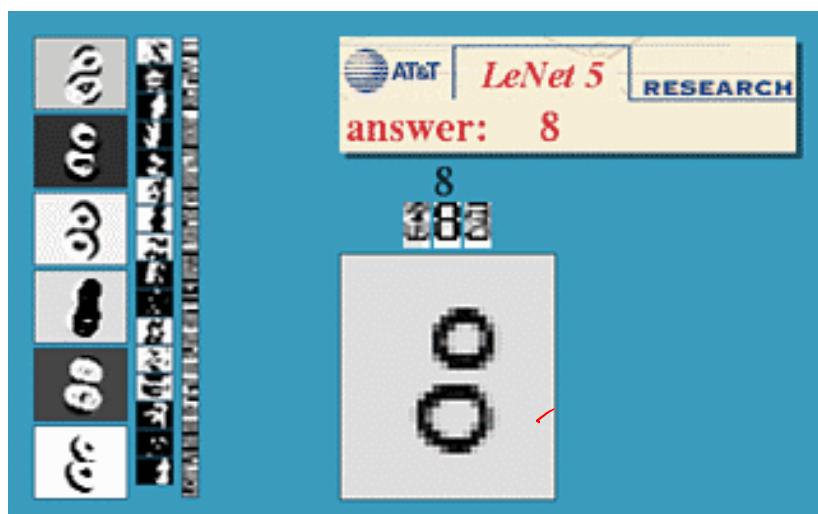
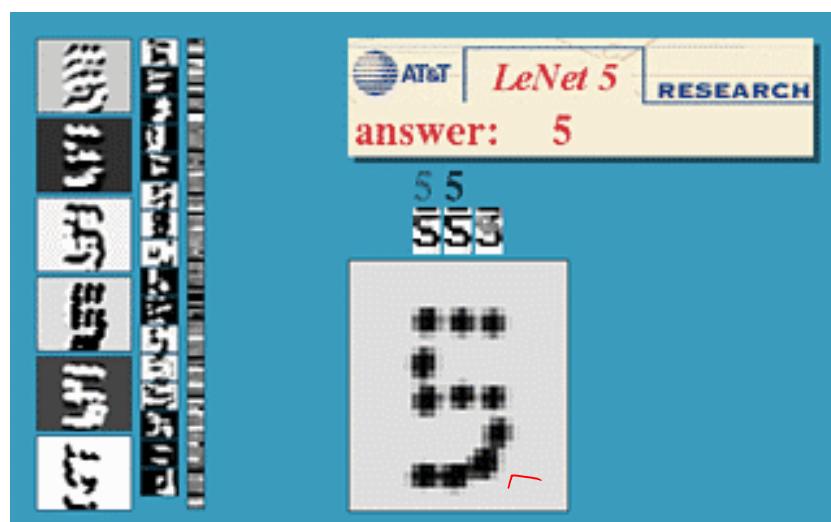
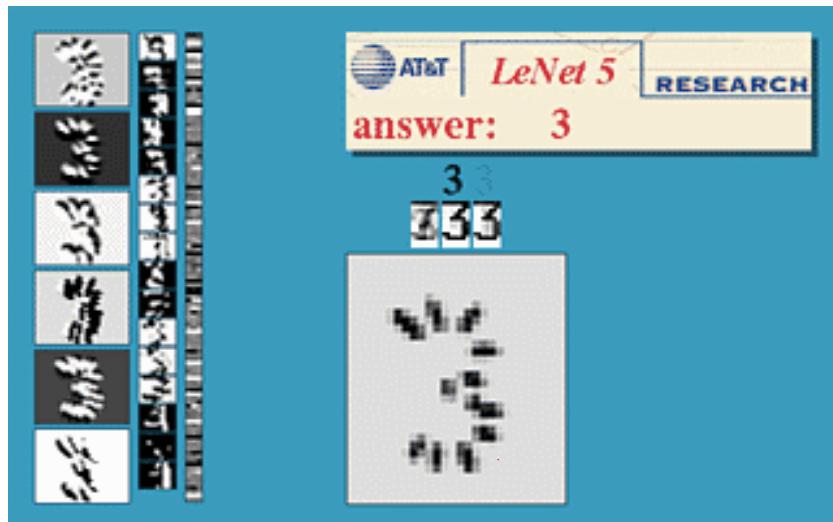


fully-connected

mnist dataset

2
4





NetTalk: A neural network that can read text aloud

Ref: Sejnowski, TJ Rosenberg, CR (1986).
NETtalk: A parallel network that learns to read aloud, Tech. Rep. No. JHU/EECS-86/Q1

Nettalk is a system that can read English text aloud and can pronounce letters accurately based on context (Sejnowski and Rosenberg 1986).

Background:

English is not a phonetic language.

Char → Sound mapping is not unique (different depending on context).

The character ‘c’ is pronounced as /k/ in “cat” and as /s/ in “façade.”

The letters “-ave” form a long vowel in “gave” and “brave” but not in “have.”

Similarly, the letters “-ea-“ are pronounced as /ii/ (long vowel) in “read” (present tense) (pronounced as “reed”) and as /i/ (short vowel) as in “read” (past tense) (pronounced as “red”).

Network

It uses a three layer MLP.

Input representation:

26 alphabets and three punctuation marks (comma, fullstop and blank space) are supported

Each character is presented along with a context which consists of three additional characters on either side of the character.

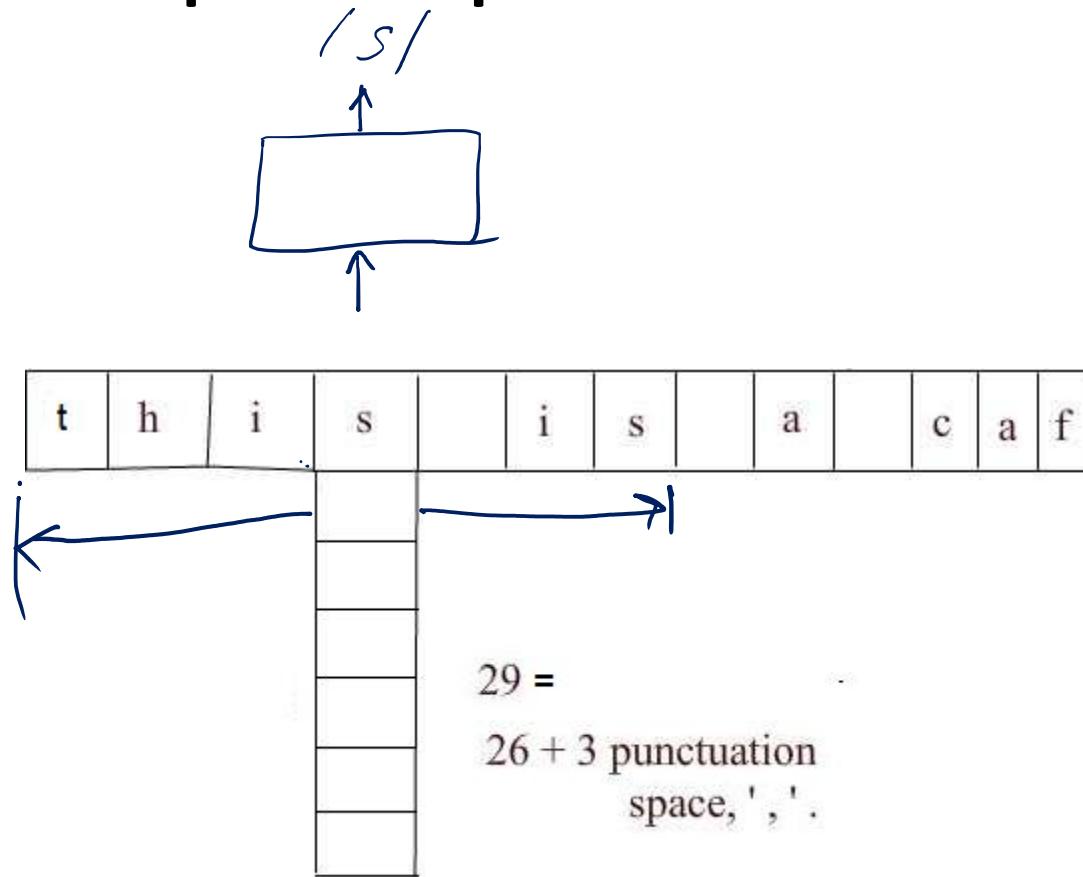
Thus text is presented as windows of 7 symbols

Hence, $(26 + 3) * 7$ input neurons

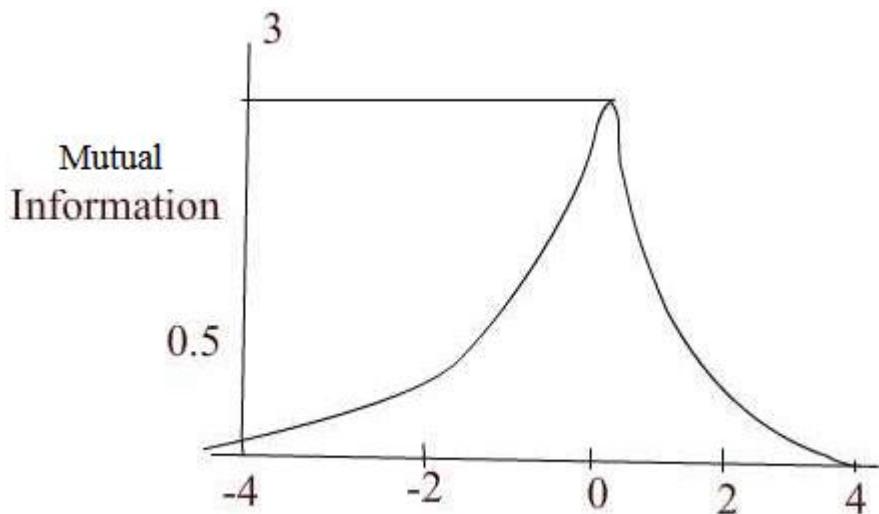
Hidden layer : 80 hidden neurons

And Output neurons: 26 (encoding phonemes)

Input representation



Network architecture



- By a window of 7 characters, mutual information from a central character falls off rapidly in both directions as shown in the above figure.
- Beyond 3 neighbors mutual information falls to less than 0.1.

Ref: Hetz et al, 1991

Training data

Phonetic transcription from informal continuous speech of a child

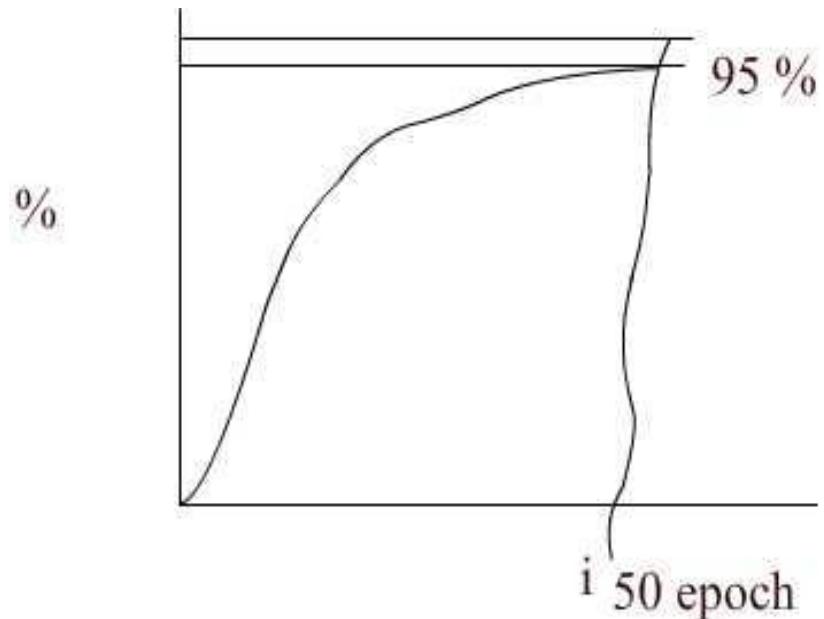
20,000 words from a pocket dictionary

Eg: phone f – o n –

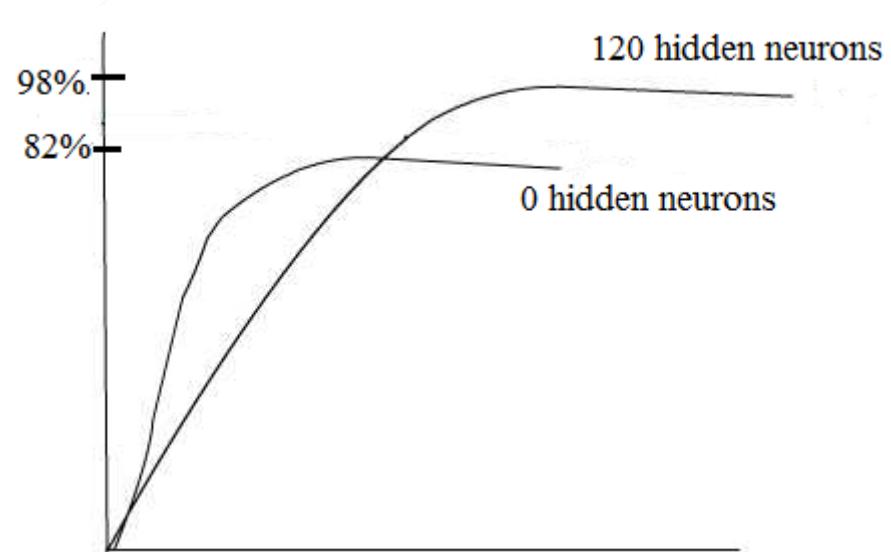
A set of 1000 words were chosen from this dictionary; these are selected from the Brown corpus of the most frequent words in English.

Learning curve

- Found in human skill learning
- Performance reaching about 95% correct after training the network with about 50,000 words.



- The presence of a hidden layer is found to be crucial to achieve this high level of performance. When a two-layer network (perceptron) was used for the same problem, performance quickly rose to 82% and saturated at that level.



Stages of learning

- One of the first features learnt by the network is distinction between vowels and consonants
- Next the network learnt to pause at word boundaries. Output resembled pseudo words with pauses between them.
- After 10 passes text was understandable

Error patterns

Errors were meaningful. e.g thesis, these

The network rarely confused between vowels & consonants.
Some errors actually were due to errors in annotation.

Test Performance

- 439 words from the regular speech. These words were not present in the training set.
- Performance was at 78%. Thus the network demonstrated that it can generalize from one set of words to another.

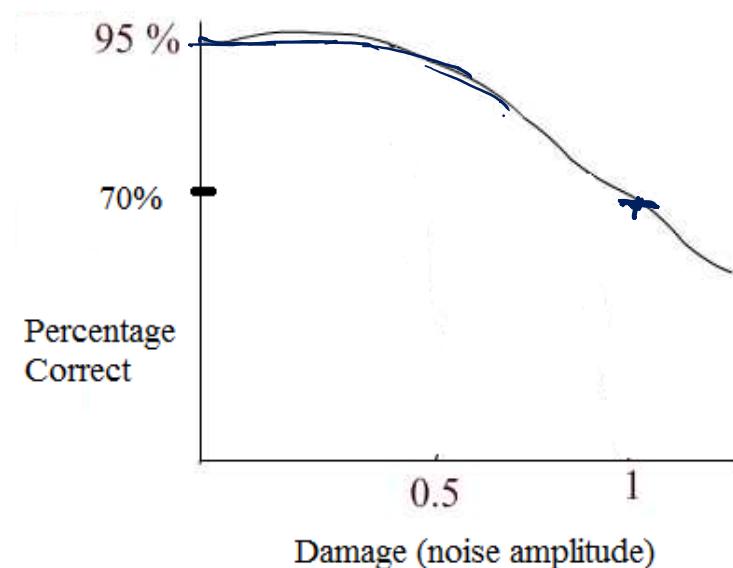
Damage testing

- In order to test the robustness of the network in face of damage, random noise, n , uniformly distributed between -0.5 and 0.5 was added to each weight in the trained network.

$$\omega_i \leftarrow \omega_i \pm v$$

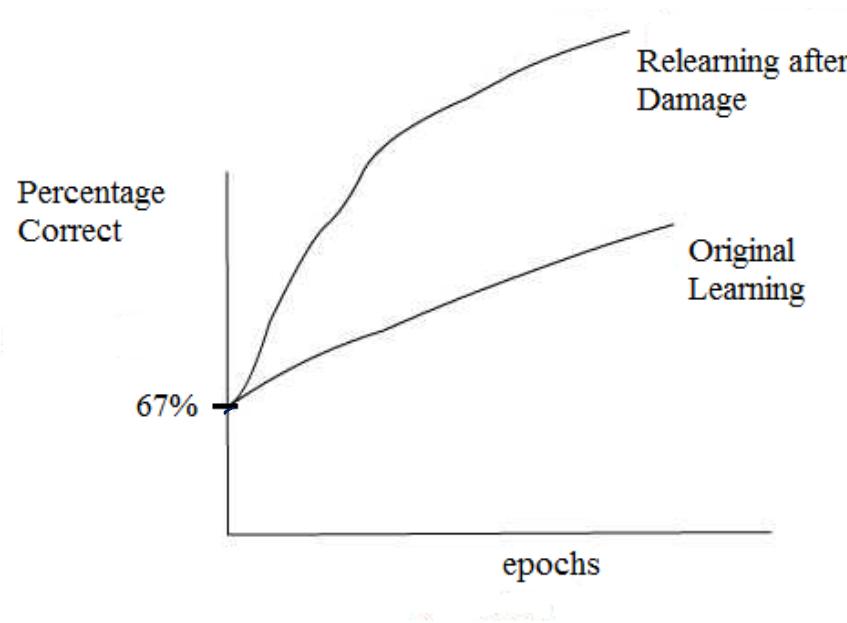
$$v \in [-0.5, 0.5]$$

- The damaged network showed 67% performance.

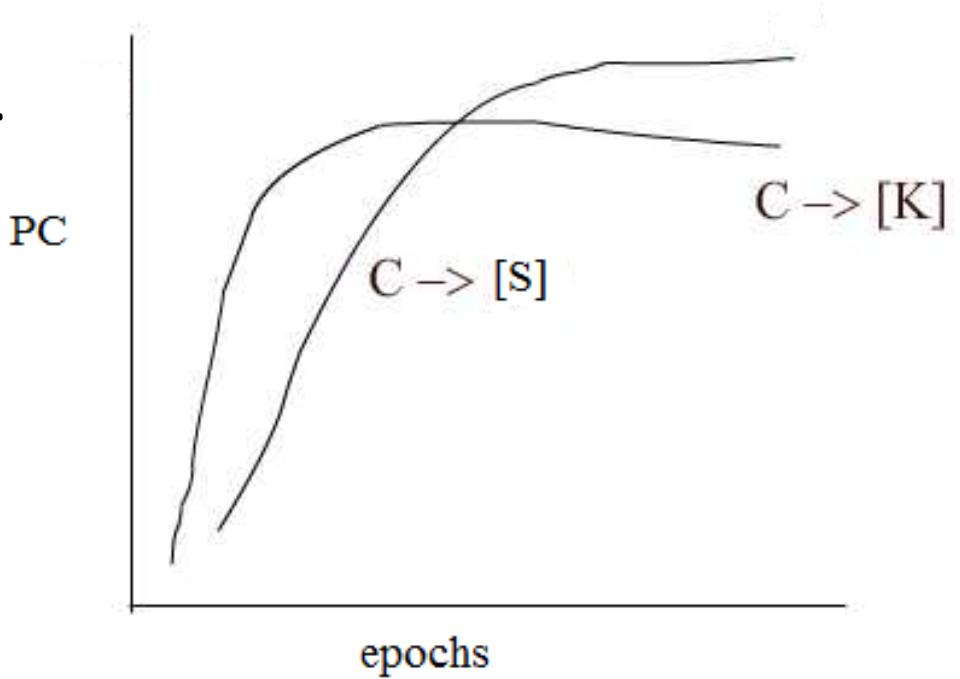


Relearning after Damage

- Although the damaged network begins with the same performance level as the original network, it relearns the task much faster than the original network.

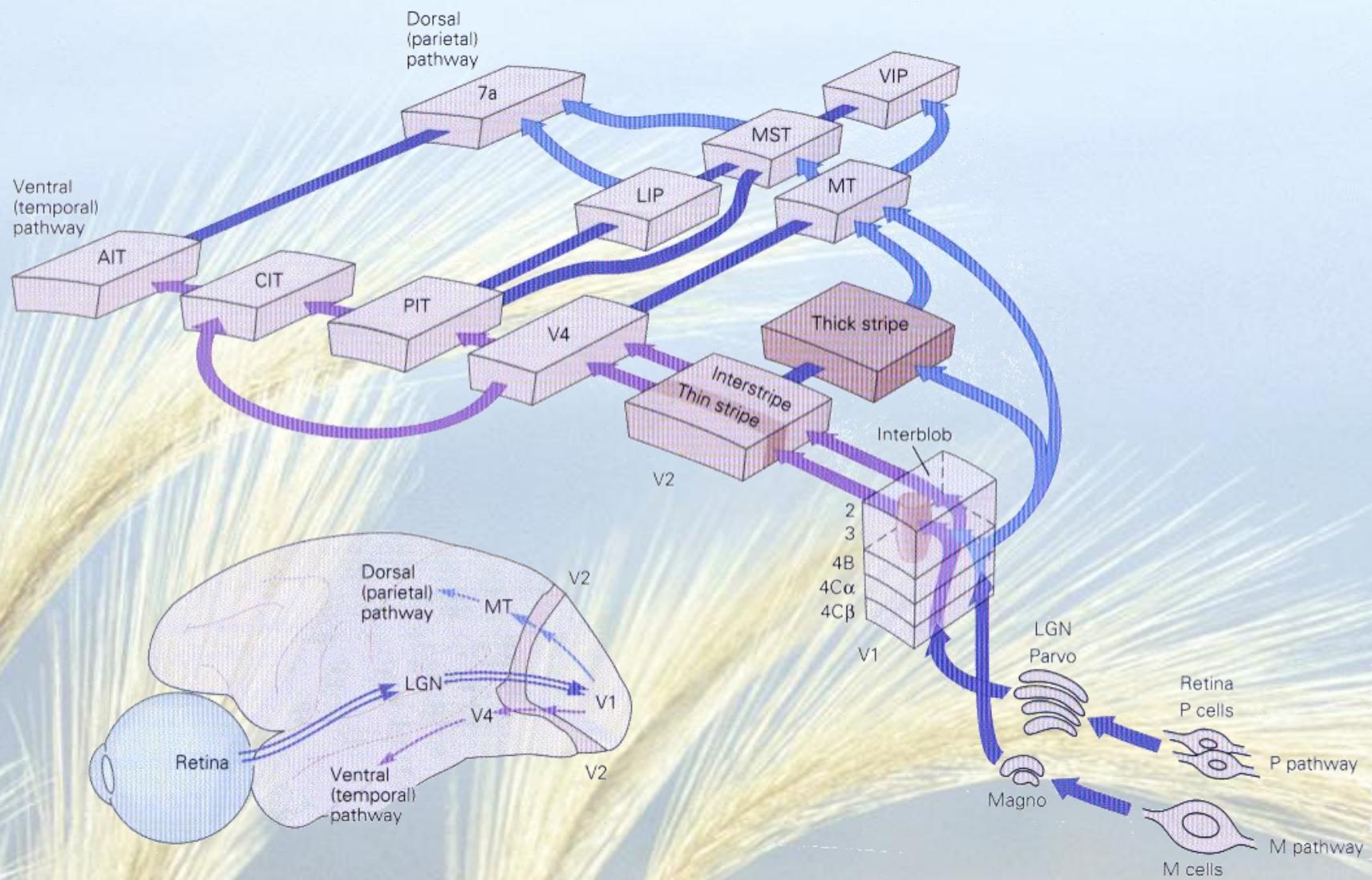


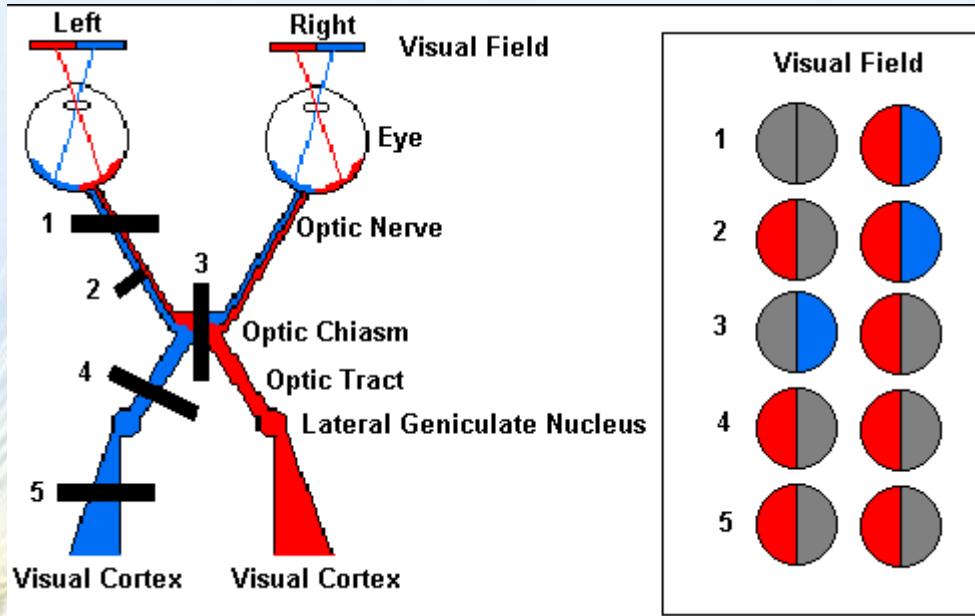
- Specific letter-to-phoneme correspondences showed that different correspondences were learnt at different rates.
- Soft 'c' (as in 'nice') takes longer to learn than hard 'c' (as in 'cat'). This is probably because hard c occurs about twice as often as soft c in the corpus.



Interestingly, children show a similar difficulty in learning.

Primary Visual Cortex V1-Striate Cortex





Damage at site #1: this would be like losing sight in the left eye.

The entire left optic nerve would be cut and there would be a total loss of vision from the left eye.

Damage at site #2: partial damage to the left optic nerve.

Here, information from the nasal visual field of the left eye (temporal part of the left retina) is lost.

Damage at site #3: the optic chiasm would be damaged.

In this case, the temporal (lateral) portions of the visual field would be lost. The crossing fibers are cut in this example.

Damage at site #4 and #5: damage to the optic tract (#4) or the fiber tract from the LGN to the cortex (#5) can cause identical visual loss.

In this case, loss of vision of the right side.

V1

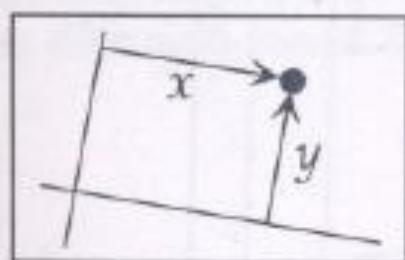
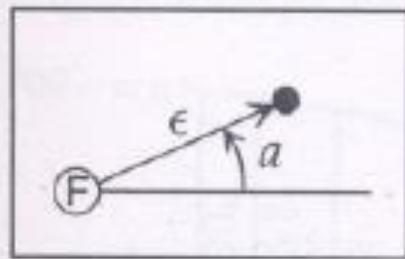
- First stopover of visual information in cortex
- Retinotopic map:
 - Retinal information is mapped onto V1 such that nearby points are mapped onto nearby neurons in V1

Visual Field

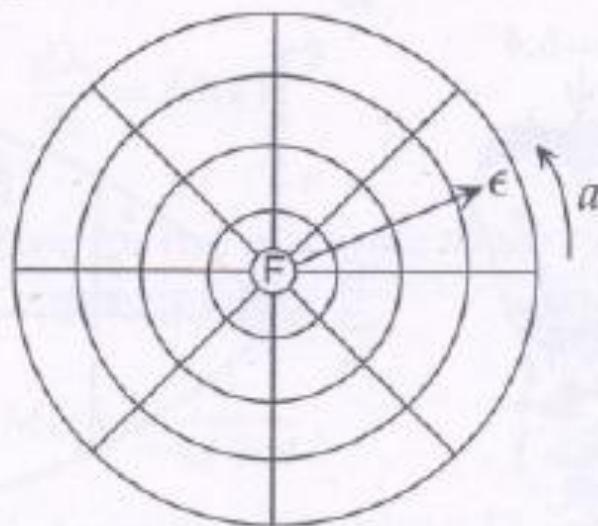
- Consider the visual field as a sphere with the viewer at the center
- “North pole” of the sphere is the fixation point; this point is mapped onto fovea
- Latitude is called “eccentricity”, ϵ ($0\text{--}70^\circ$)
- Longitude is called “azimuth,” α (-90° to 90°)

Visual field

A



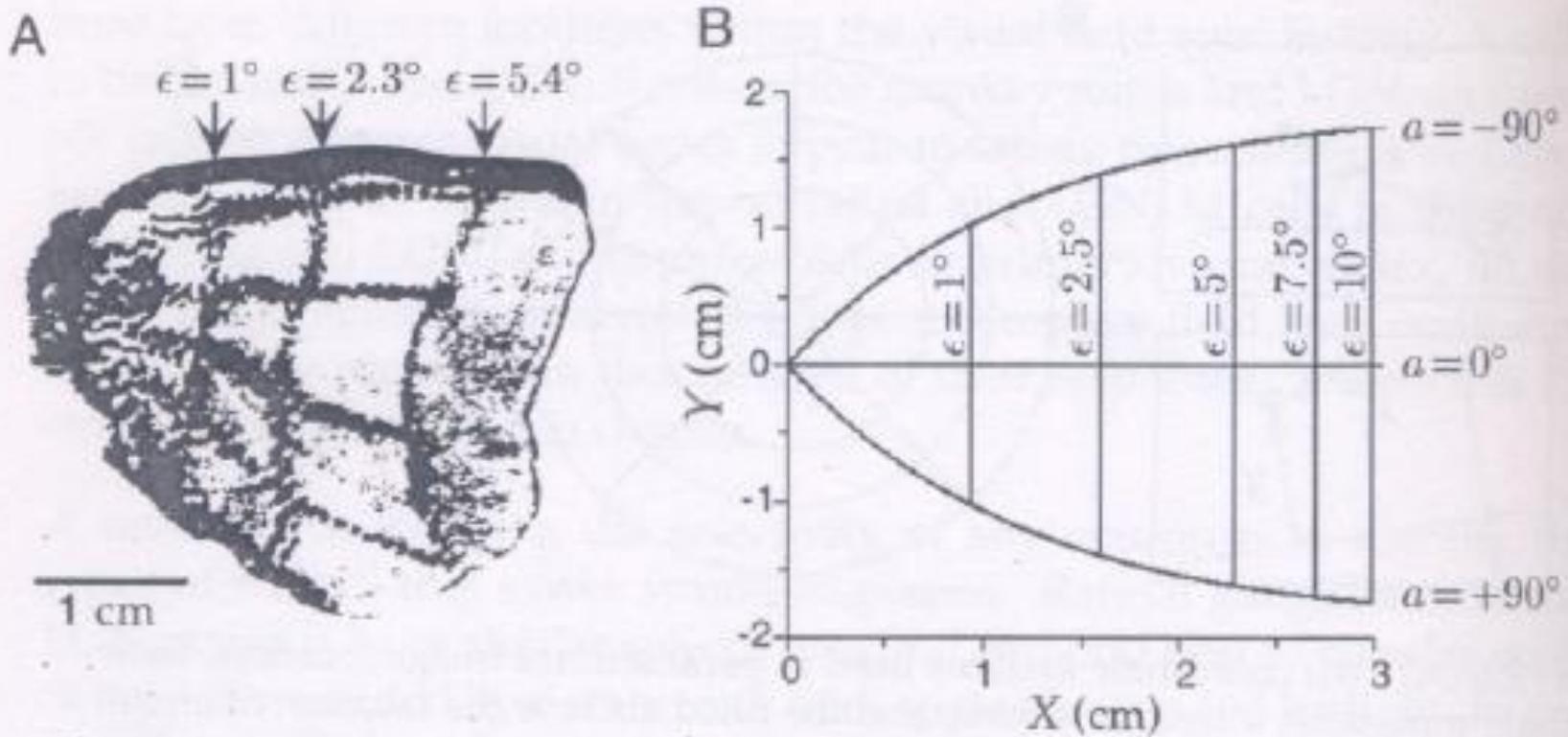
B



Experiment to find Retinotopic Map

- “Bull’s-eye” pattern is displayed on a screen
- Pattern of activity in V1 is imaged by using a radioactive glucose; imaging reveals which neurons are active (taking up glucose)
- Experiment performed on monkey

Retinotopic Map



Retinotopic Map Features

- Vertical lines correspond to circles in the image
- Roughly horizontal lines correspond to radial lines
- Fovea is represented at the leftmost pole
- Azimuthal angles are positive in lower half, and negative in the upper half

Responses of neurons in V1

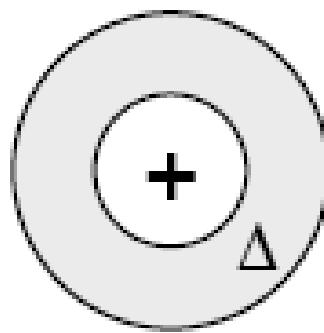
- Work of Hubel & Wiesel at Harvard in '60s.
 - Neurons in V1 respond to oriented bars and edges and not to spots.
 - Some neurons also respond to moving bars.
 - Different neurons respond to different orientations. Within a dia of 1mm, all orientations are represented.

Simple and Complex Cells

- Simple cells: respond to an oriented line present in the center of their RF
- Complex cells: respond to an oriented line present almost anywhere in their RF
- End-stopping: some cells respond best when the line is not too long.

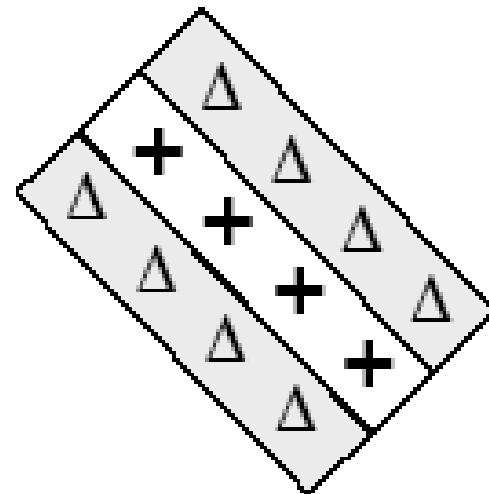
Transformation of Receptive Field Properties from LGN to Primary Visual Cortex

LGN
Neuron

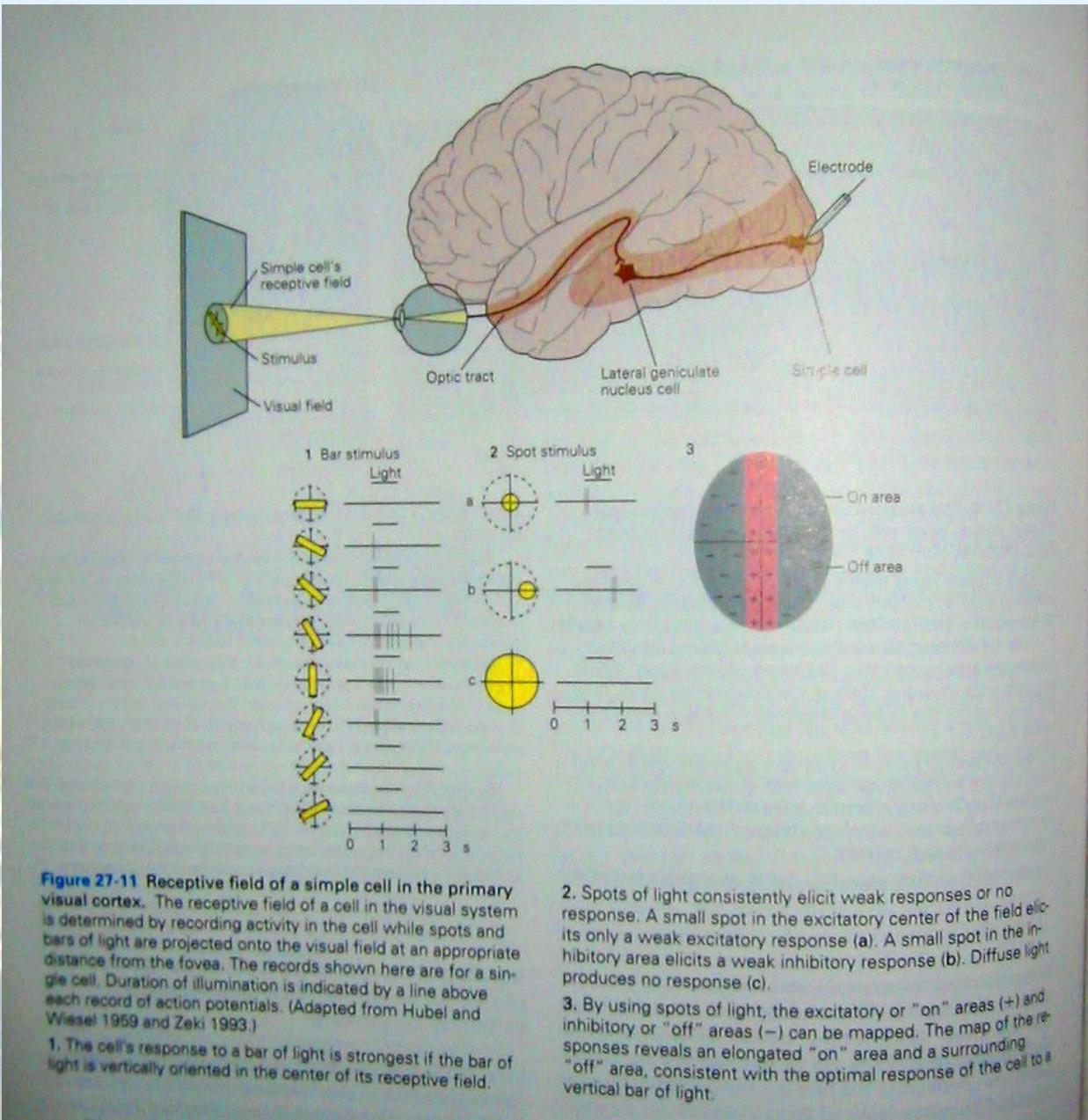


Circular Receptive Field
(e.g., on-center, off-surround)

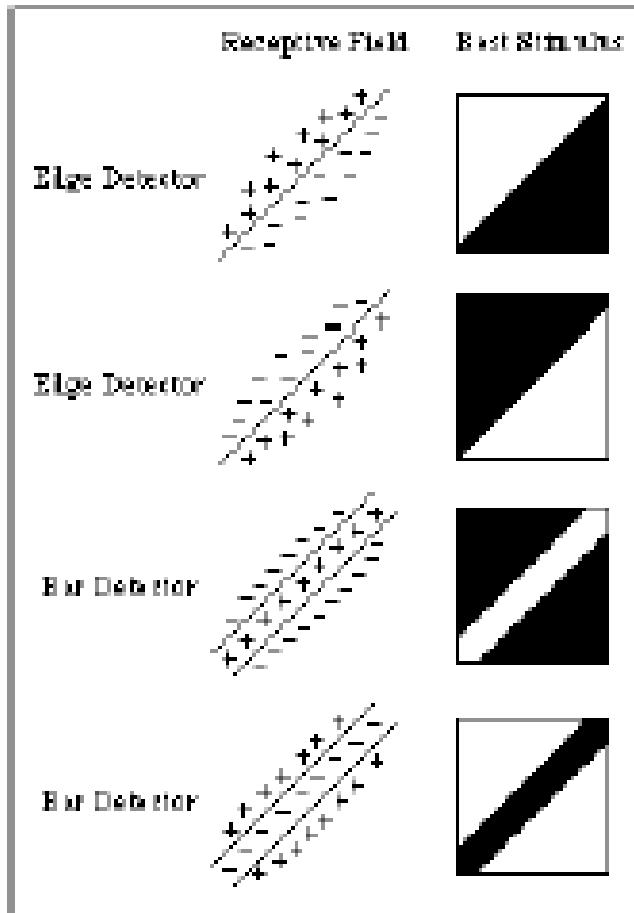
“Simple Cell”
in Cortex



Rectangular Receptive Field
(e.g., on-center, off-flanks)



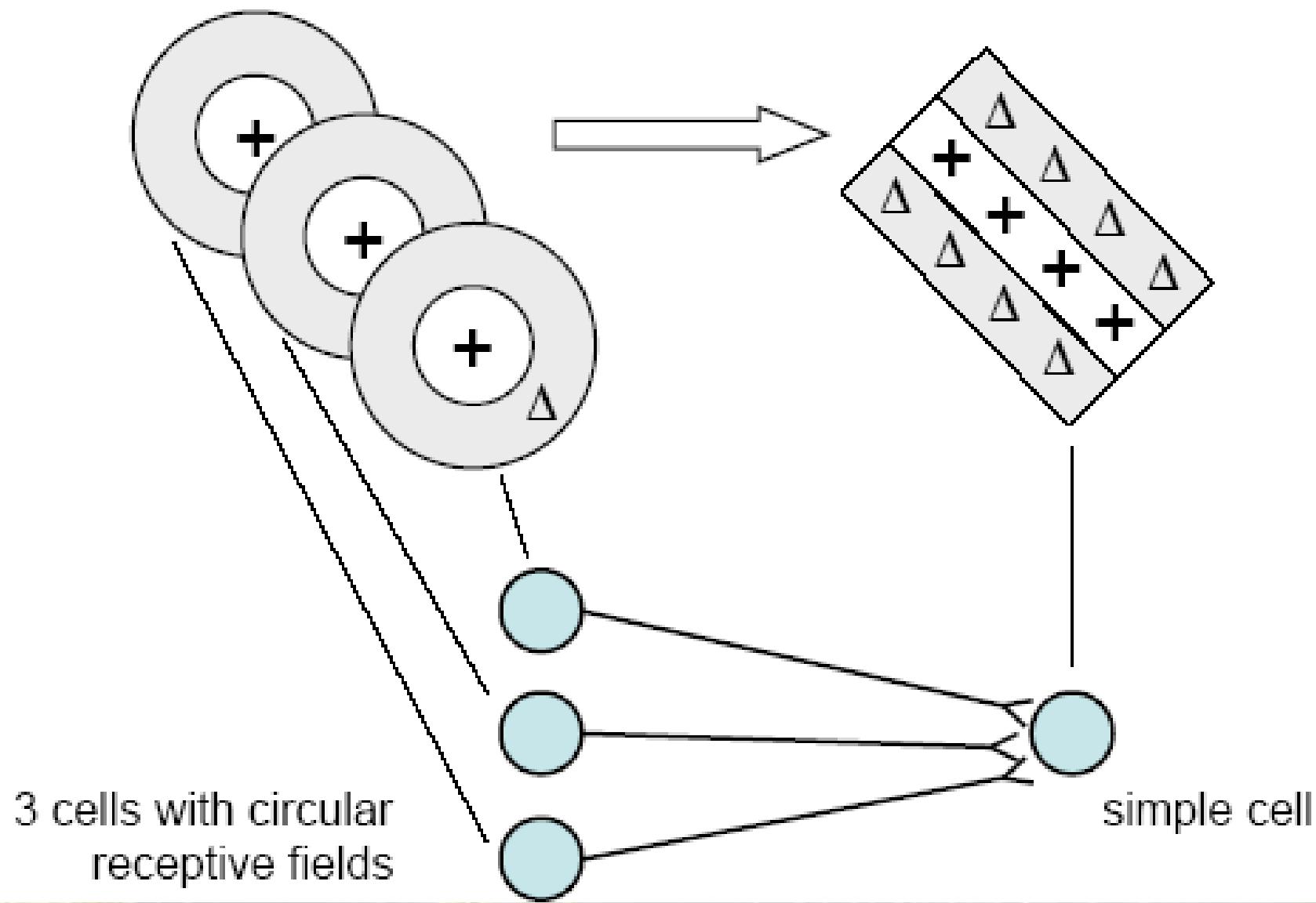
Examples of Simple Cell Receptive Fields



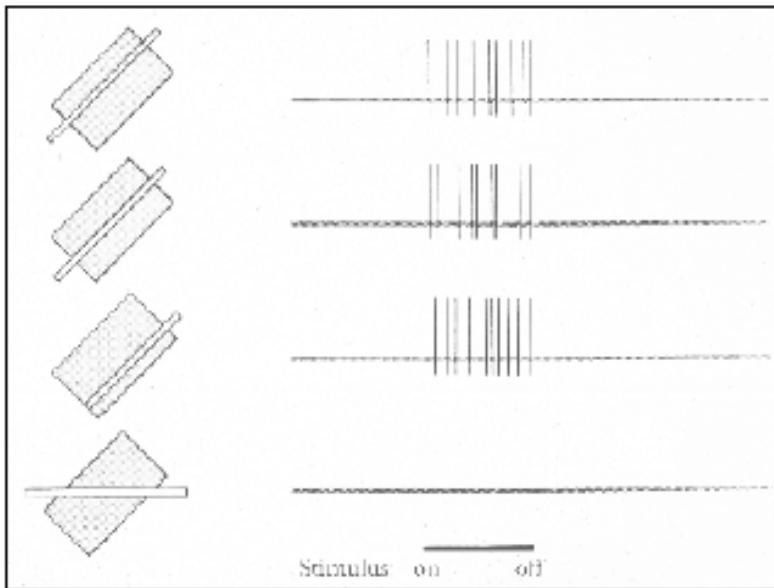
<http://www.cs.queens.uontario.ca/~psych/psych2800/ch4/orientSelac.html>

Simple cells can be used to detect edges and contours

Hypothetical Wiring Diagram for Generating a Simple Cell Receptive Field



Complex Cell Receptive Field Properties



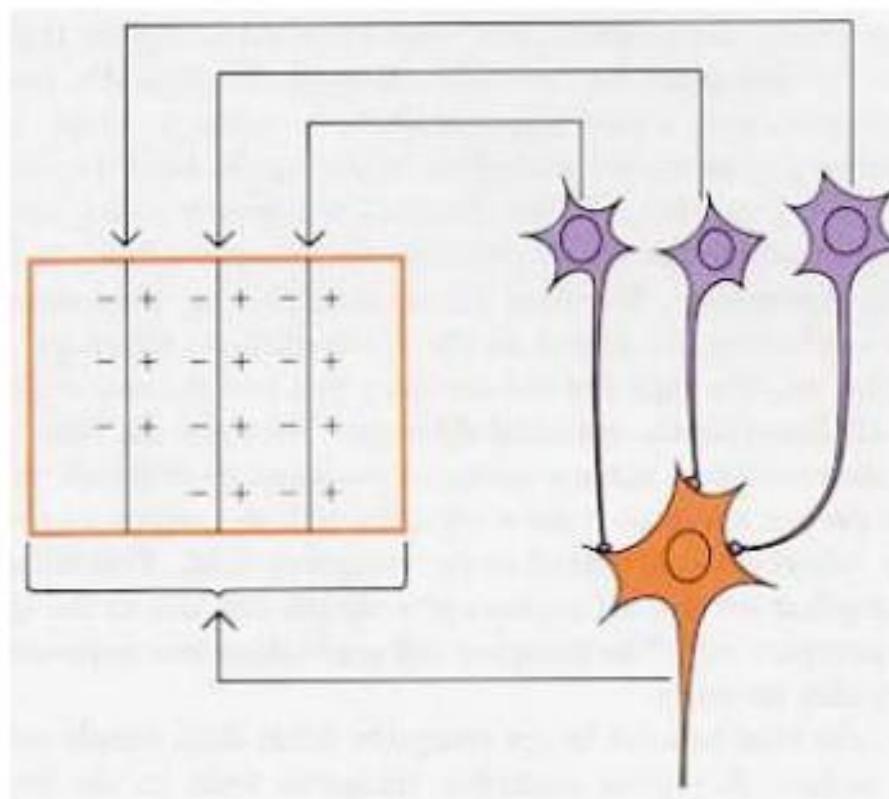
<http://www-psych.stanford.edu/~lera/psych115s/notes/lecture3/figures.html>

	Receptive Field	Best Orientation
Orientation Detector	$\begin{matrix} - & + & - & + & - \\ - & - & - & + & - \\ + & + & + & + & - \\ + & - & - & + & - \\ + & - & + & + & + \end{matrix}$	

<http://www.cquest.utoronto.ca/psych/psy280f/ch4/orientSelc.html>

- “On” and “off” regions throughout receptive field
- Orientation selective

Hypothetical Wiring Diagram for Generating a Complex Cell's Receptive Field



<http://neuro.med.harvard.edu/site/dh/b18.htm>

Construction of complex cell receptive field via
input from multiple simple cells

Schematic of Orientation Selectivity in the Primary Visual Cortex

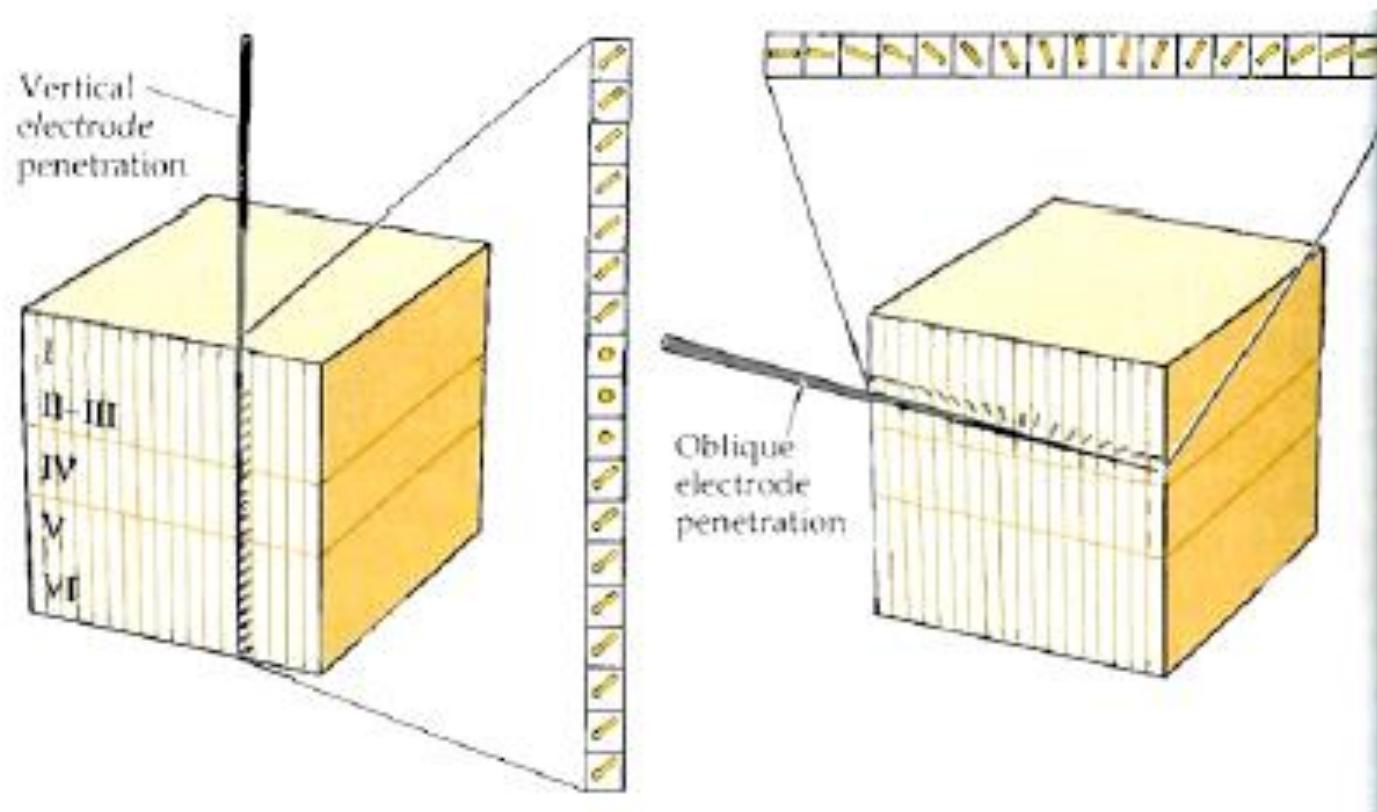
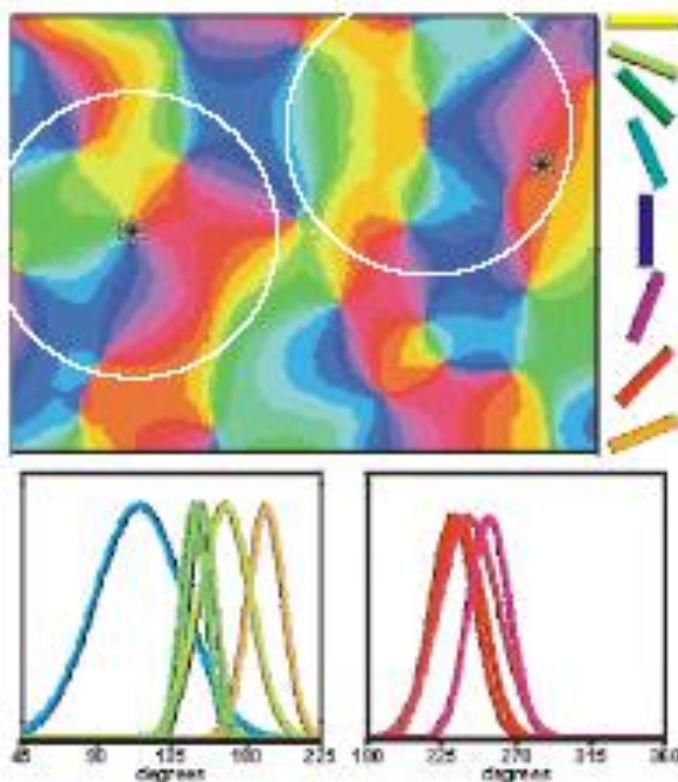


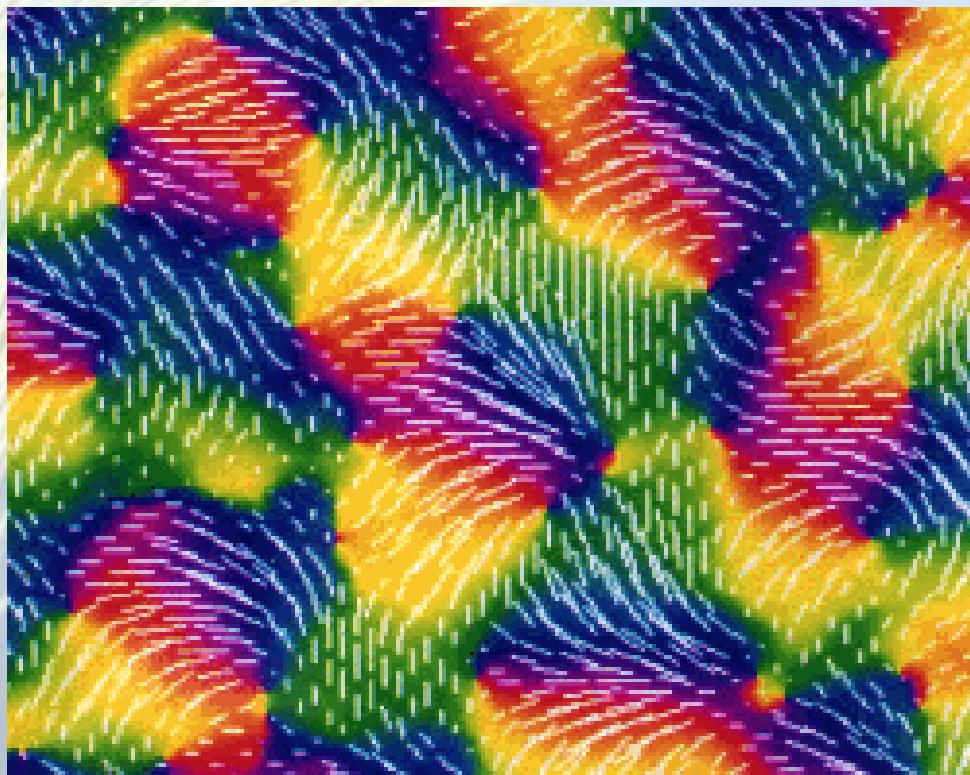
Fig. 11.12, Purves et al., Neuroscience, 3rd edition

- Oblique/tangential penetration reveals progressive change in orientation selectivity
- Vertical penetration reveals columnar organization of orientation selectivity

Pinwheel Arrangement of Orientation Columns Revealed by Optical Imaging of Intrinsic Signals



Orientation Columns



Schematic of Ocular Dominance Columns

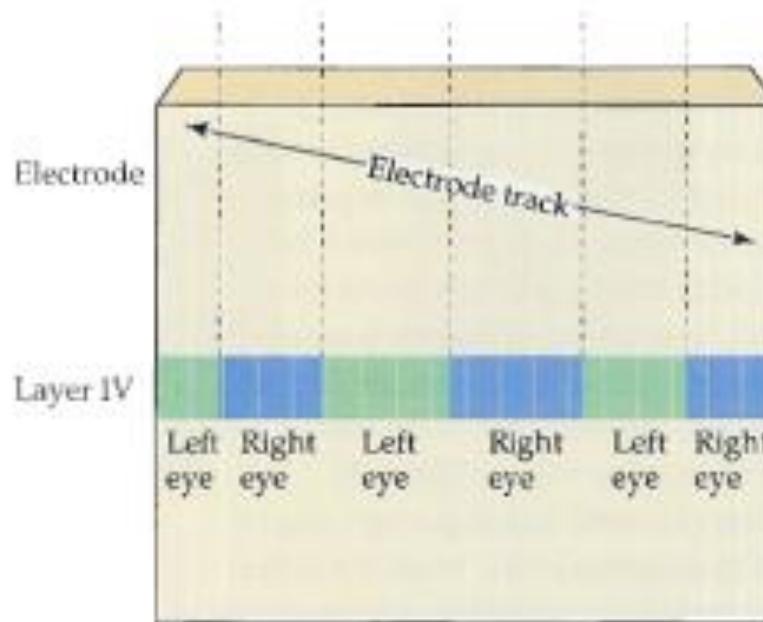


Fig. 11.13, Purves et al., Neuroscience, 3rd edition

- Alternating columns of cells showing preferential responses to right eye or left eye input
 - Monocular cells in layer 4
 - Binocular cells are found in other layers

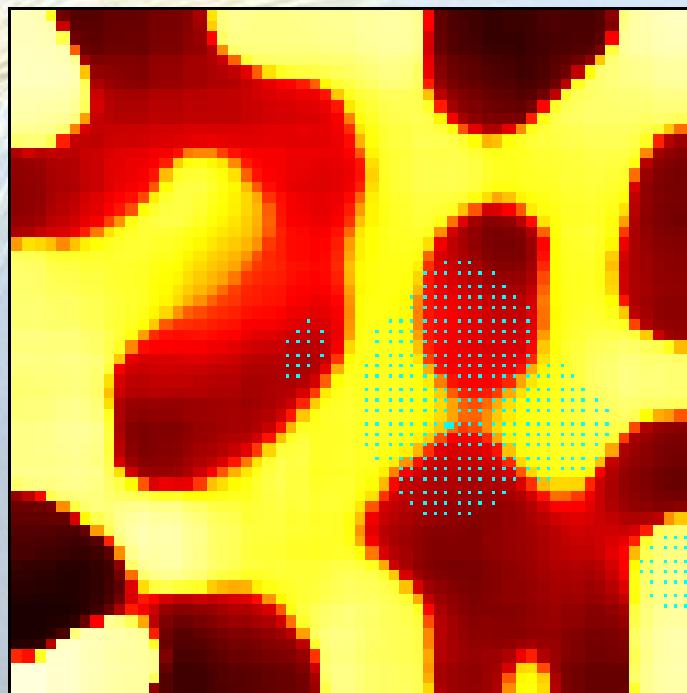
Ocular Dominance Columns in Primary Visual Cortex Revealed by Trans-Synaptic Labeling

- Injection of ^3H amino acid tracer into one eye
- Tracer is transferred trans-synaptically from retina to LGN to cortex
- Autoradiography of flattened cortical sheet reveals interdigitating regions of left eye vs. right eye inputs

Autoradiogram of V1



Ocular Dominance Columns in Simulations



Orientation sensitivity AND Ocular Dominance

Properties of OrientationMaps:

- The maps of O.S. and O.D. are highly repetitive
- Orientation changes continuously as a function of cortical location except at isolated points.
- Orientation changes by 180 deg around singularities
- Both types of singularities appear in equal numbers
- There exist line-like regions (fractures), across which orientation preferences change rapidly with distance.
- (Obermeyer, Blasdel & Schulten 1992)

Orientation sensitivity AND Ocular Dominance

- Properties of Ocular Dominance Maps
 - Ocular dominance changes continuously as a function of cortical location.
 - The ocular dominance pattern is locally organized into parallel strips, which sometimes branch and terminate.
 - **Iso-orientation slabs often cross the borders of ocular dominance bands at approximately right angles.**
 - **The singularities tend to align with the center of the ocular dominance bands.**

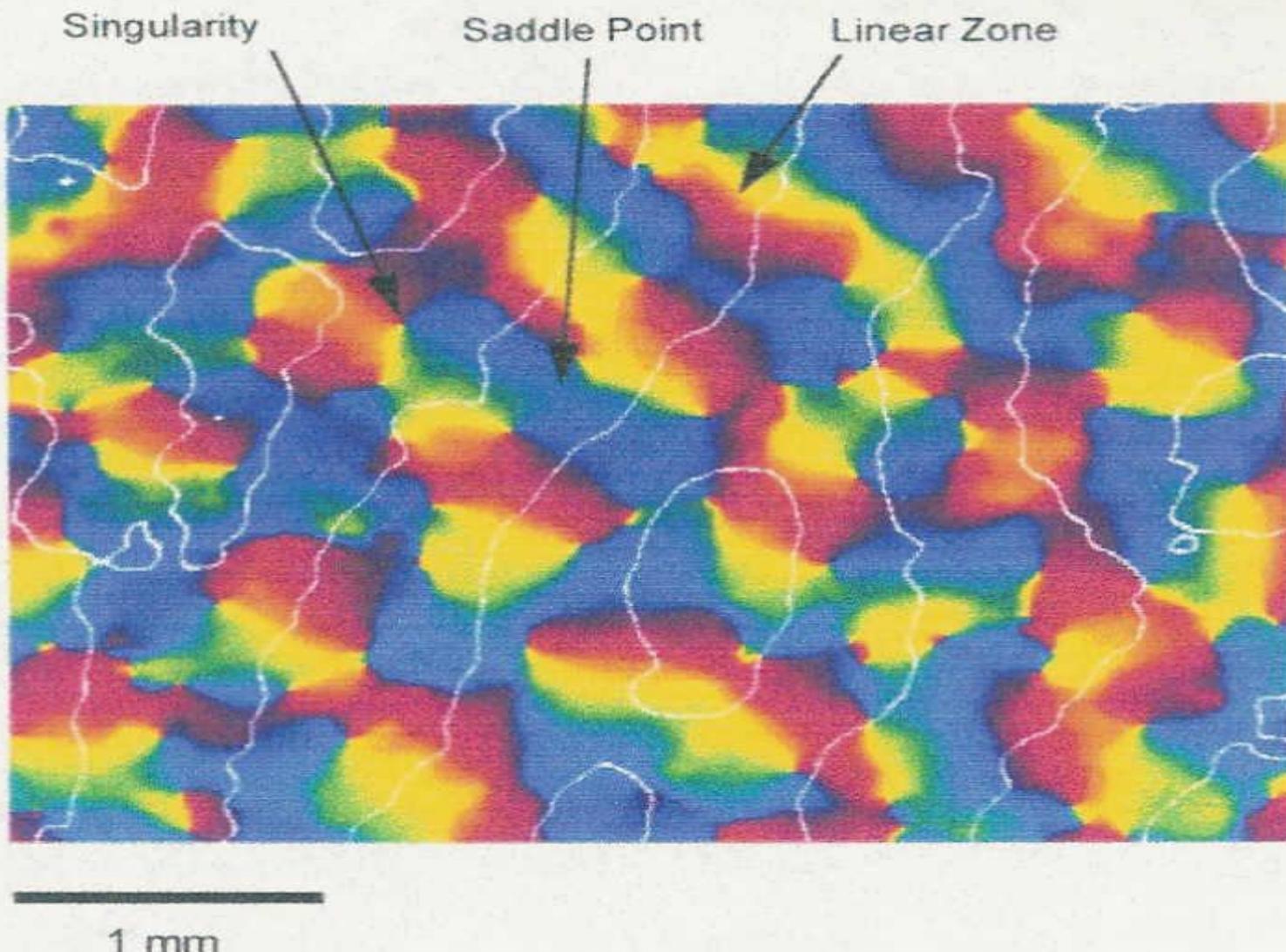
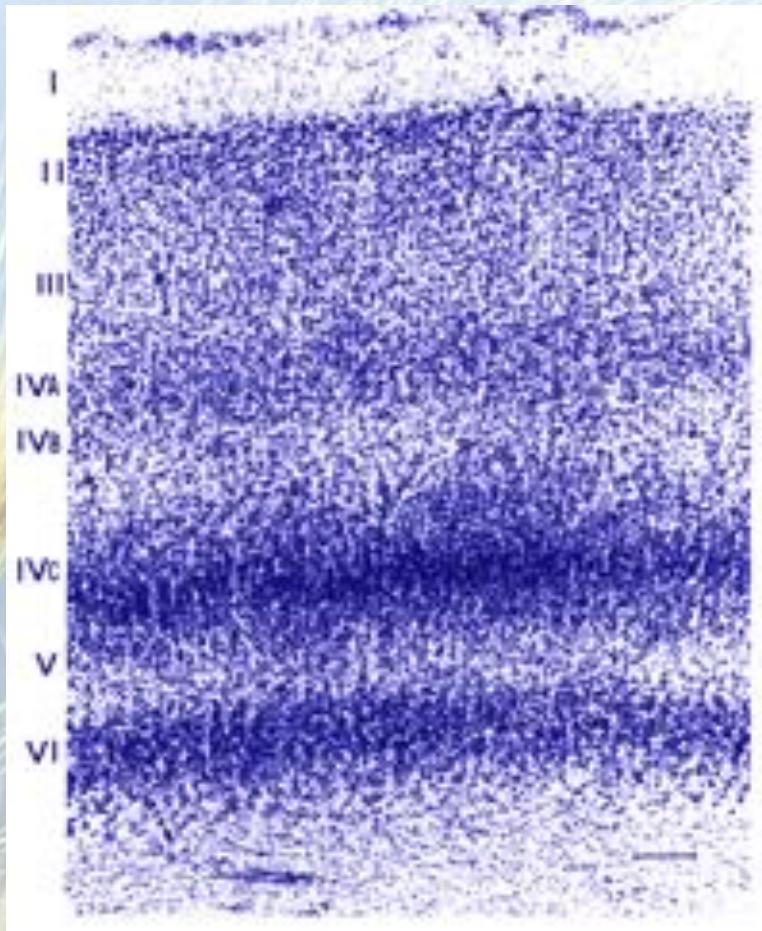


Figure 3. Composite figure showing the arrangement of orientation domains (a single colour represents a unique range of orientation preferences) and their relationship with ocular dominance column boundaries (white lines). The images were obtained by optical recording in macaque monkey striate cortex. Note that the iso-orientation domains tend to intersect ocular dominance column borders at right angles. (Figure supplied by K Obermayer, from data presented in Blasdel (1992b).)

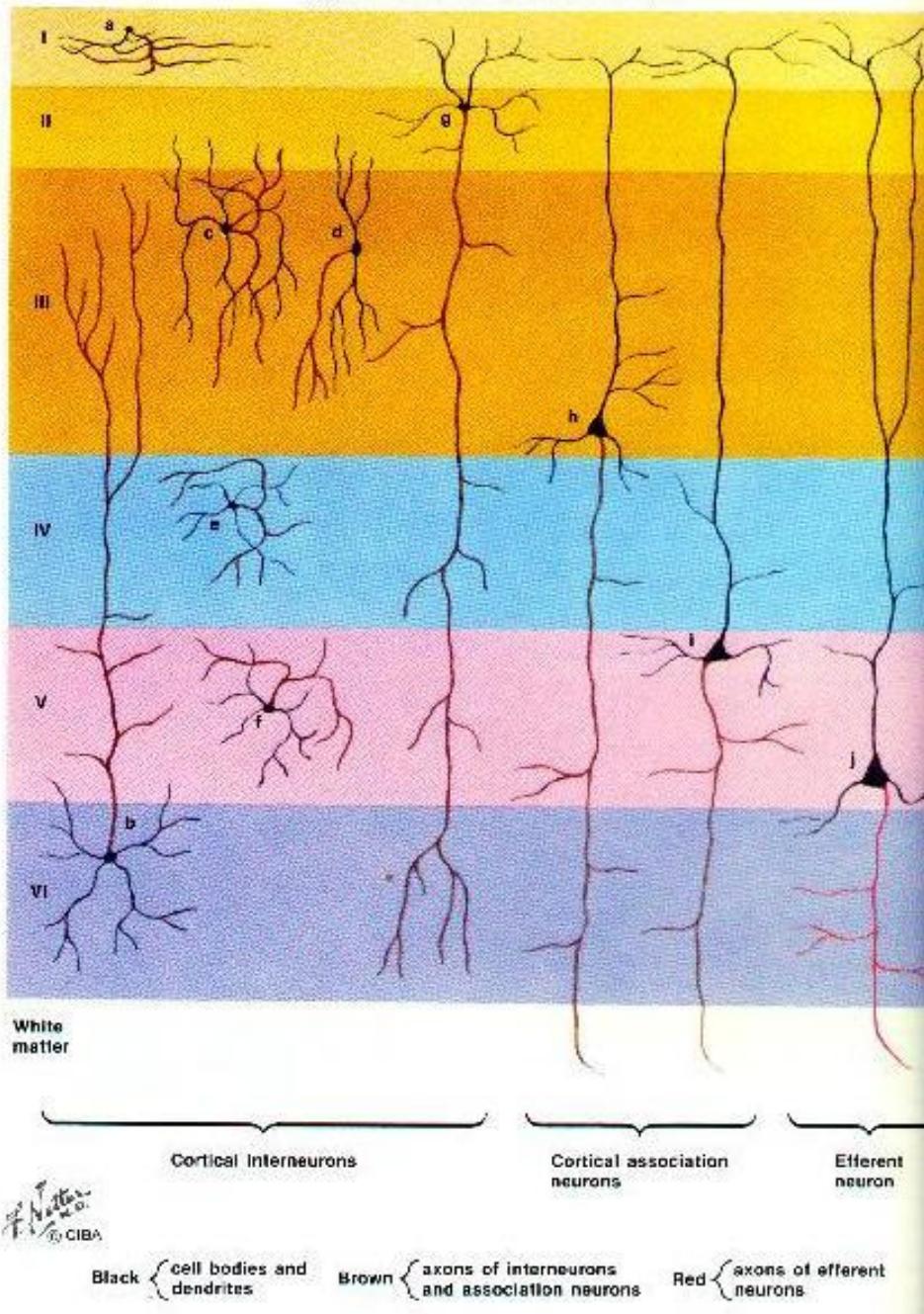
Blobs

- Peg-shaped regions of cells
- In layers 2 and 3 of V1
- These cells respond to color, not orientation
- About 0.2 mm dia

Cortical Layers



Types of Neurons in Cerebral Cortex



Blob regions

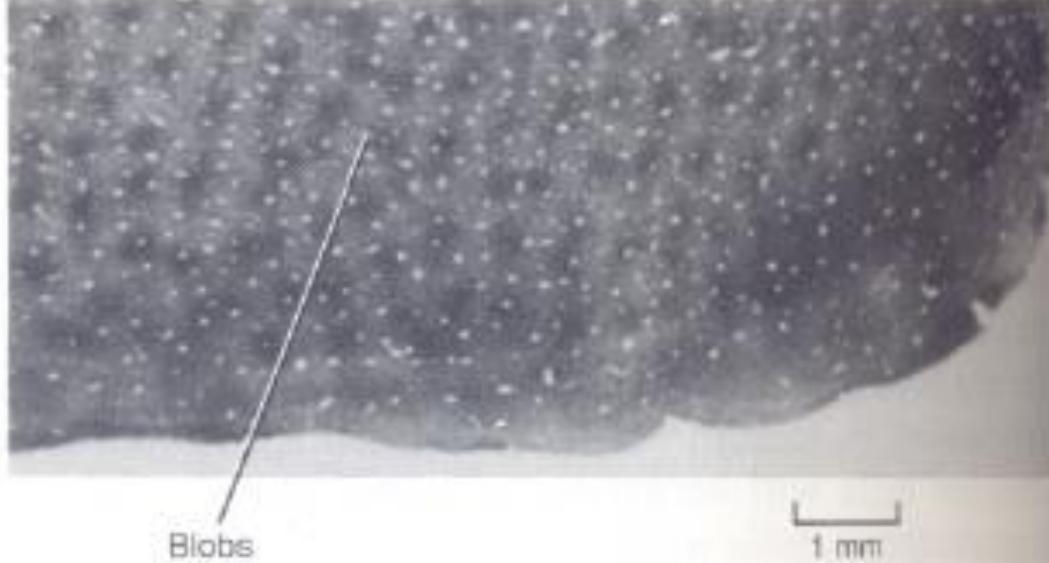


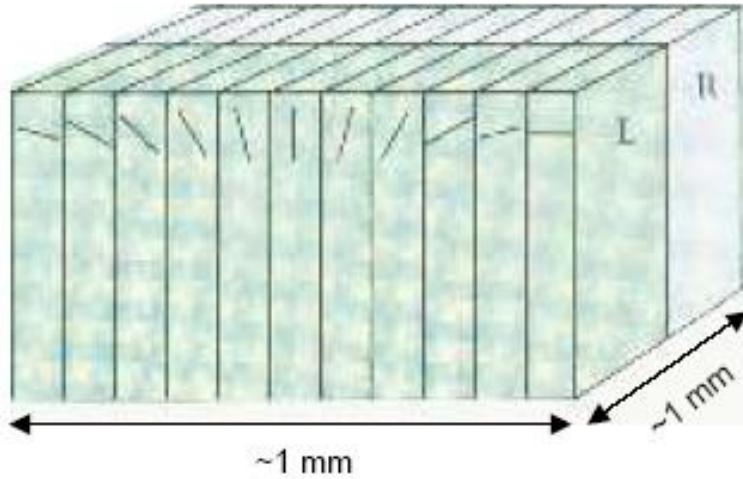
FIGURE 29–13

The distribution of the mitochondrial enzyme cytochrome oxidase in the superficial layers of the visual cortex, as seen in tangential sections of area 17 of the macaque monkey. The rows of dark patches or *blobs* represent areas of heightened enzymatic activity. This is thought to represent heightened neural activity in the blobs because of the lower response selectivity of these cells. (Courtesy of D. Ts'o, C. Gilbert, and T. Wiesel.)

Hypercolumn

- Smallest unit in V1 necessary to analyze all aspects of a region of visual field.
- Area = 1 sqmm
- Complete set of orientation columns (180°)
- Inputs from both eyes.
- Several blobs

Schematic of a Cortical Hypercolumn: A Unit of Information

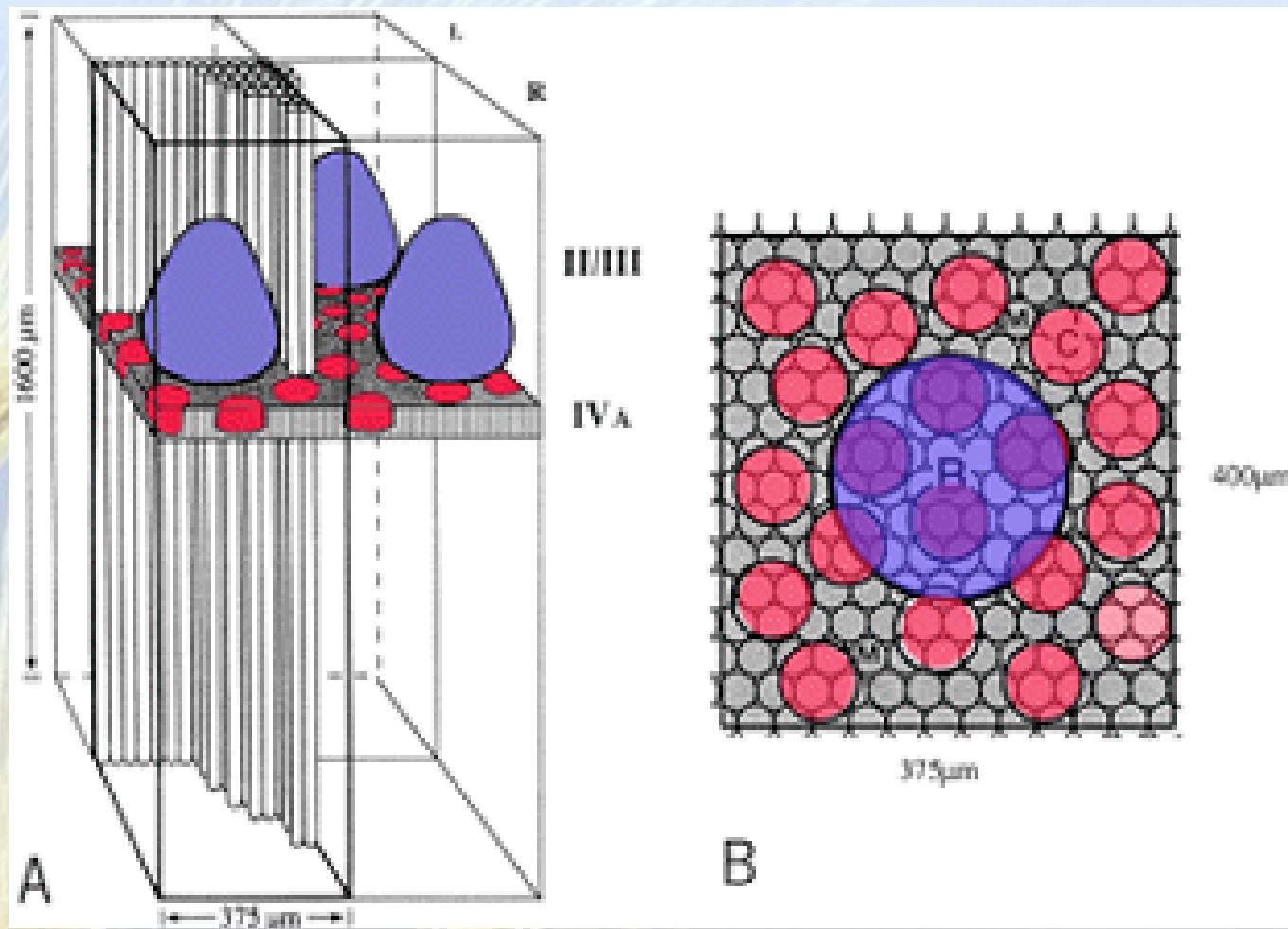


<http://www-psych.stanford.edu/~lera/psych115s/notes/lecture3/figures.html>

Each column
= 30 – 100 μ m

~1 mm x ~1 mm
180° orientation
one L + R pair

Hypercolumns



Horizontal connections among hypercolumns

- Axon collaterals of pyramidal cells in layers 3 and 5 run long distances parallel with layers
- They give rise to clusters of axon terminals at regular intervals that approximate the width of hypercolumn

Blob regions form a horizontal network

- (Lund and Rockland)
- Injected horseradish peroxidase into restricted regions within cortical layers 2 and 3
- Found elaborate honeycomb-like lattice of labeled cells and axons that formed walls around unlabeled patches about 500 μm dia

Orientation sensitive cells form a horizontal network

- Tso, Gilbert, Wiesel
- Recorded cells that respond to a given orientation
- Many cells fire simultaneously
- Injected fluorescently labeled microbeads which were taken up axon terminals at the site of injected and transported to cell bodies.
- Injected radio-labeled 2-deoxyglucose, which revealed active cells
- Anatomical and metabolic patterns coincide well

Other visual areas

- One in area 17 – V1 (striate cortex)
- Two in area 18 – V2, V3
- Three in area 19 – V3a, V4, V5 (Middle Temporal area)
- Parietal cortex – V5a (Medial Superior Temporal area)

Functions of visual areas

- V1 – primary visual analysis
- V2 – more visual analysis
- V3 – dynamic form
- V4 – color and form
- V5 - motion

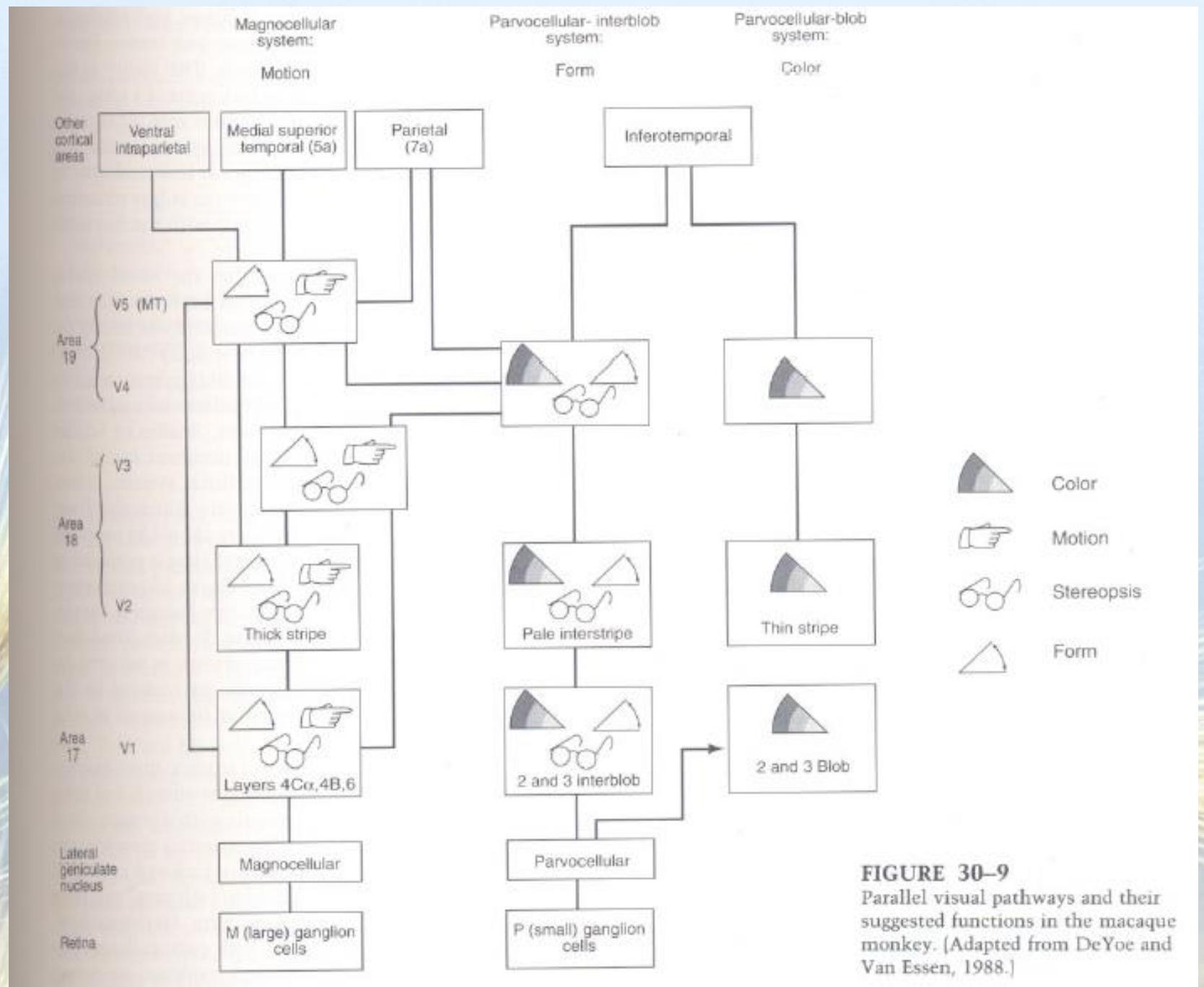


FIGURE 30–9
Parallel visual pathways and their suggested functions in the macaque monkey. [Adapted from DeYoe and Van Essen, 1988.]

Visual streams at V1

- Beginning of segregation of visual streams in V1
 - Magnocellular system
 - Parvocellular-interblob system
 - Parvocellular-blob system

Magnocellular system

- Specialized for: motion, spatial analysis, stereo vision etc
- Relatively insensitive to color
- Neurons respond rapidly but transiently
- Layers 4Ca, 4B, 6 in V1

Parvocellular-interblob system

- Specialized for form (and to some extent color)
- Area 2 and 3 interblob
- Neurons here respond to orientation of edges

Parvocellular-blob system

- Specialized for color
- 2 and 3 layer in blob regions

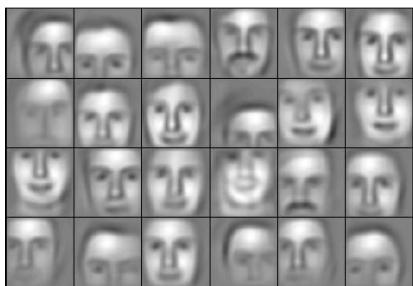
Structure of V2

- Located in area 18
- Staining revealed 3 patches:
 - Thick stripes
 - Thin stripes
 - Pale interstripes

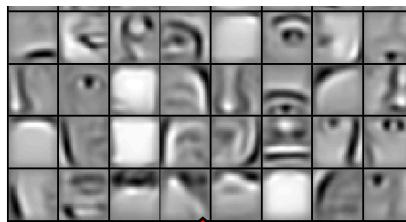
The Visual System: Retina-LGN-V1-V2-V3-V4- V5

How does a CNN learn to recognize faces?

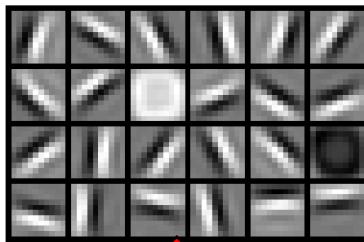
Last layer learns nearly the entire face



Middle layer learns parts of the face



Lowest layer learns edges



Hierarchical organization in the visual system

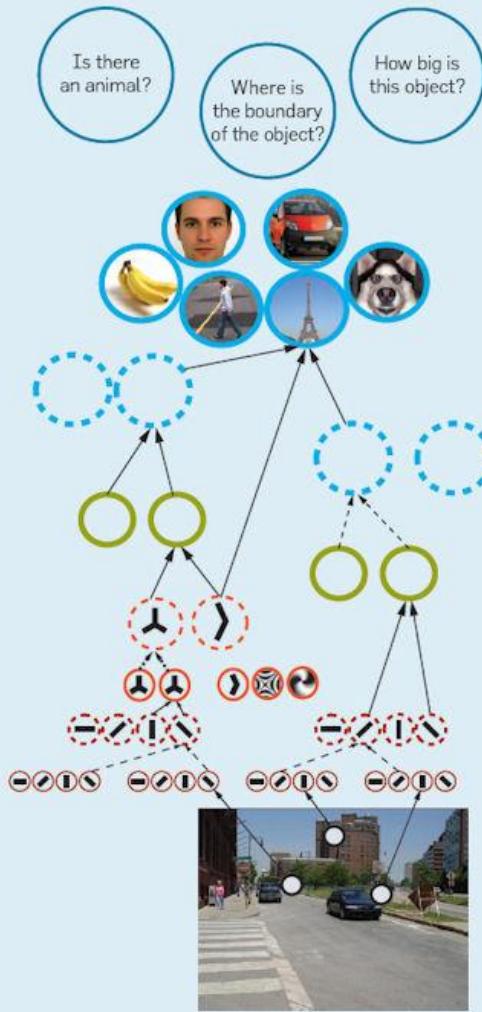
visual routines

AIT

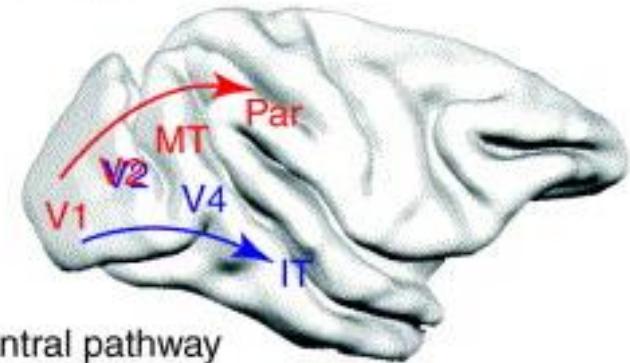
PIT

V2-V4

V1



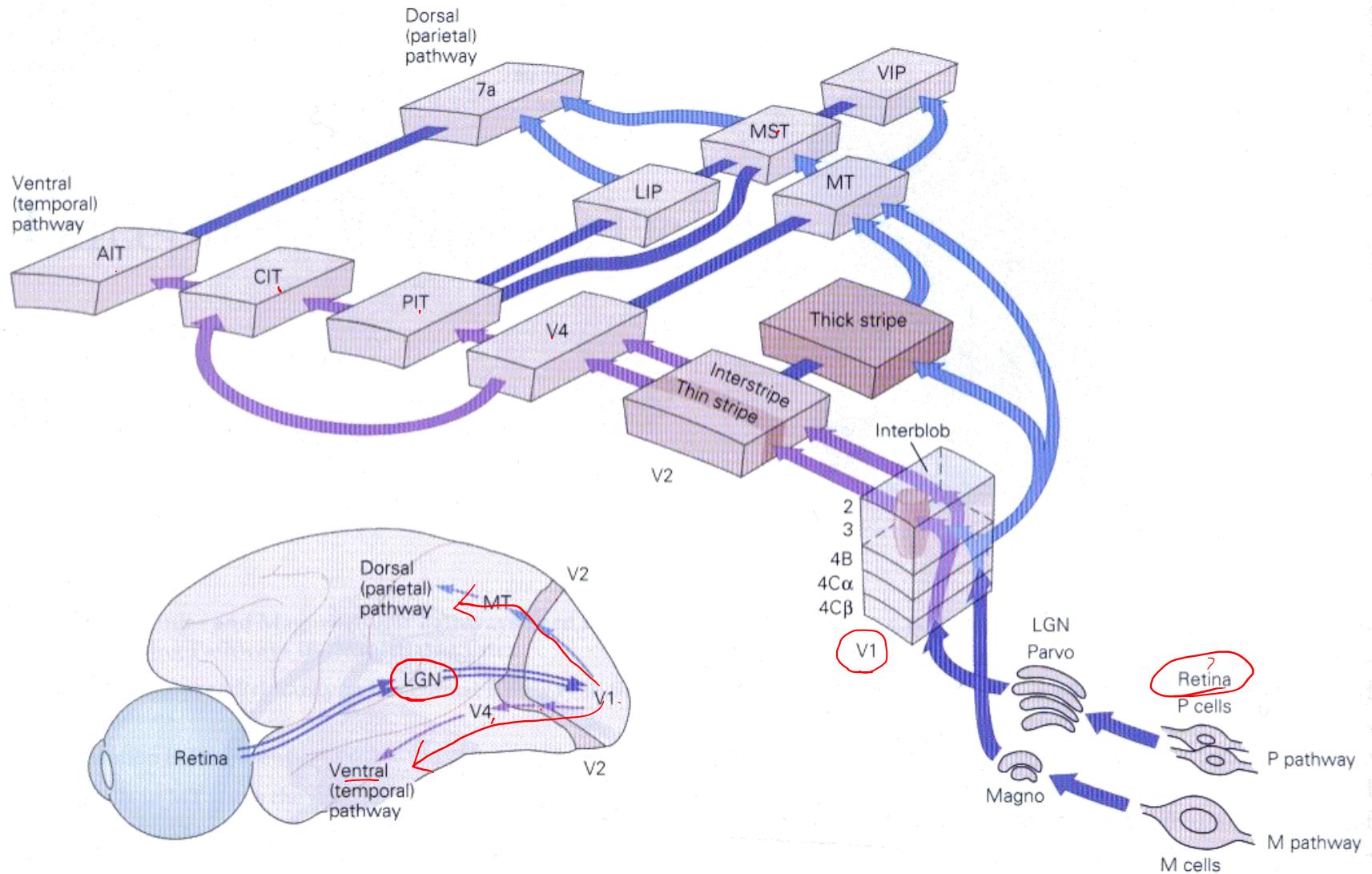
Dorsal pathway



Ventral pathway

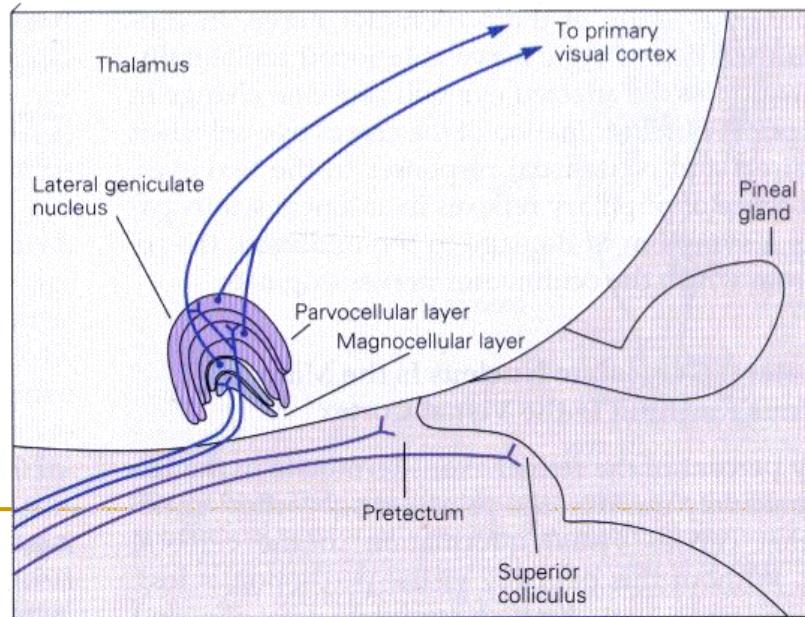
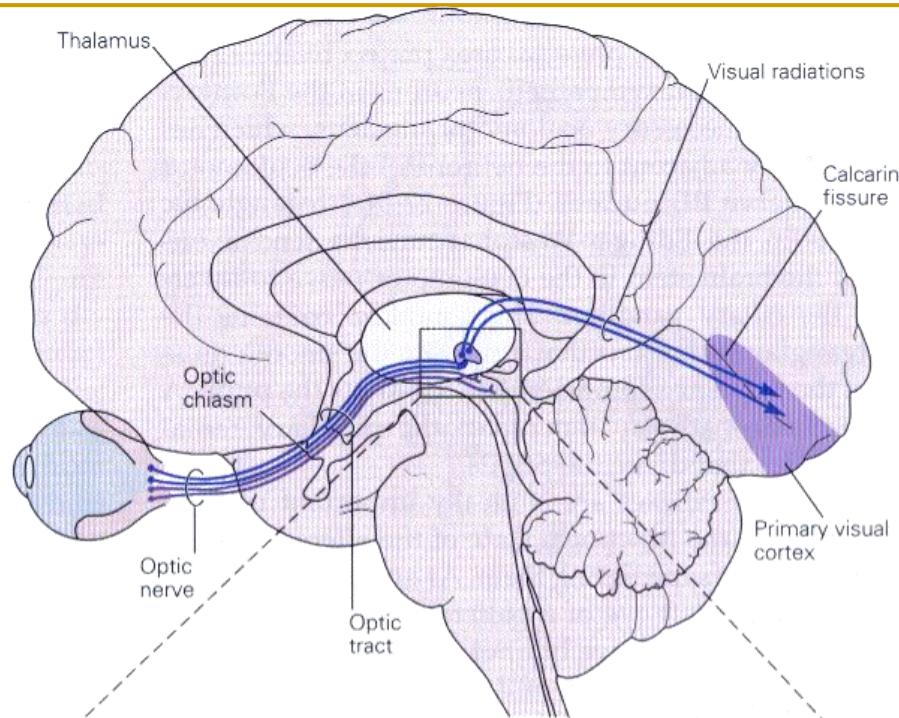
**LET'S LEARN ABOUT THE
VISUAL SYSTEM...**

Visual pathway



Visual Areas

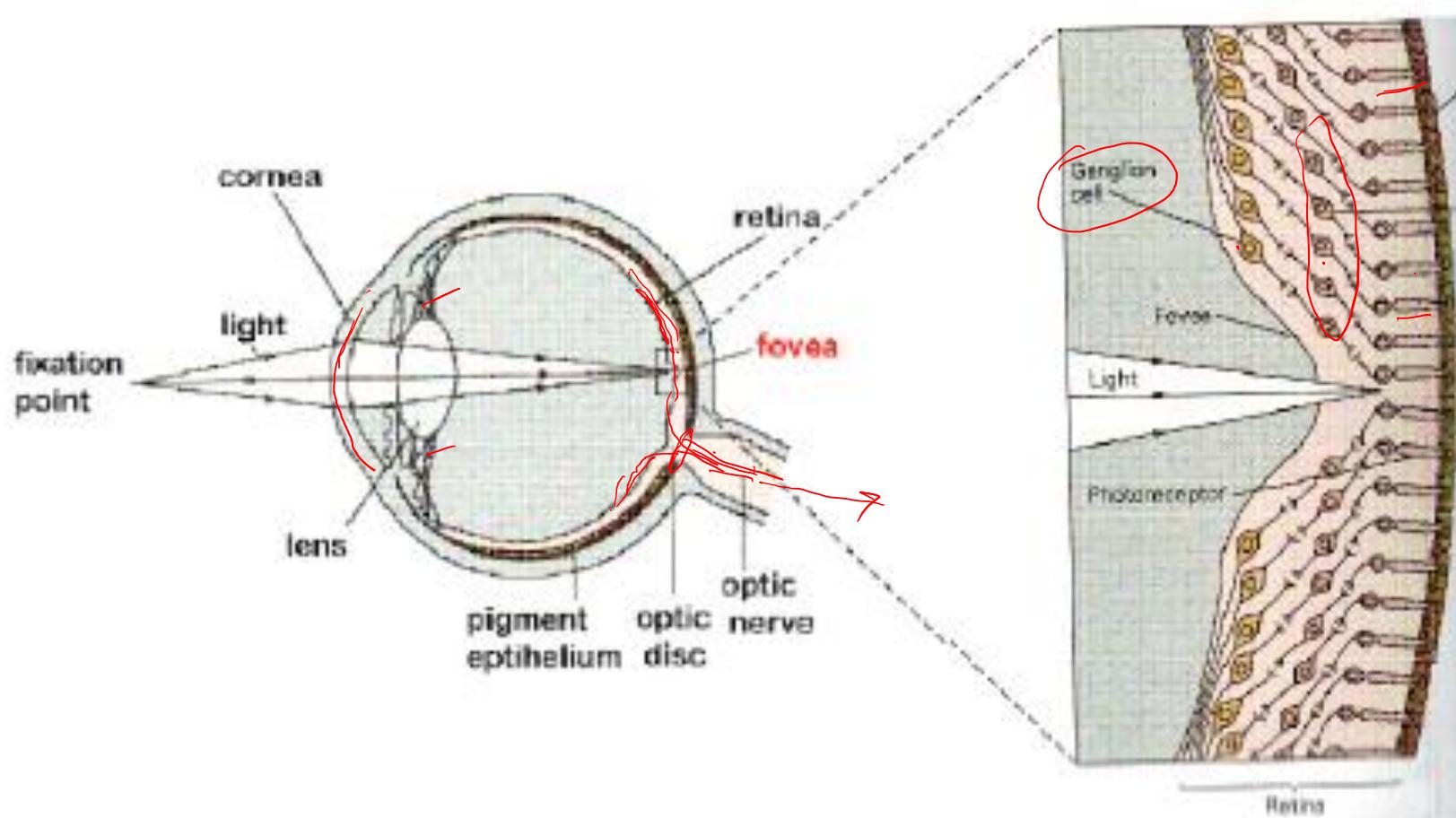
- Area 7a
- LGN: lateral geniculate nucleus
- V1: striate cortex
- AIT: anterior inferior temporal area
- CIT: central inferior temporal area
- LIP: lateral intraparietal area
- MST: medial superior temporal area
- MT: middle temporal area
- PIT: posterior inferior temporal area
- VIP: ventral intraparietal area



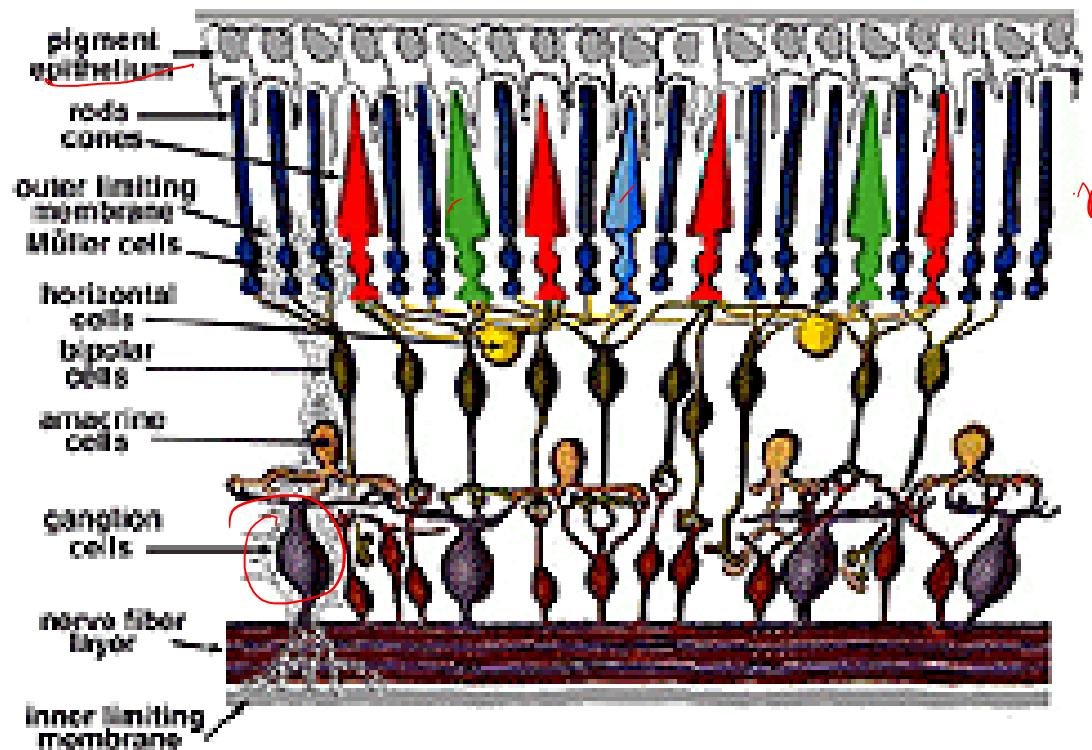
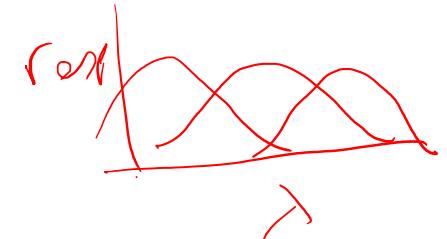
[http://www.inma.ucl.ac.be/
EYELAB/neurophysio/
light_perception/LGN.html](http://www.inma.ucl.ac.be/EYELAB/neurophysio/light_perception/LGN.html)

The Fovea

- Packed with cones - high density of sensors, divergent connections
- Minimal light scattering (depleted of other cell types)
=> High acuity (high resolution) vision



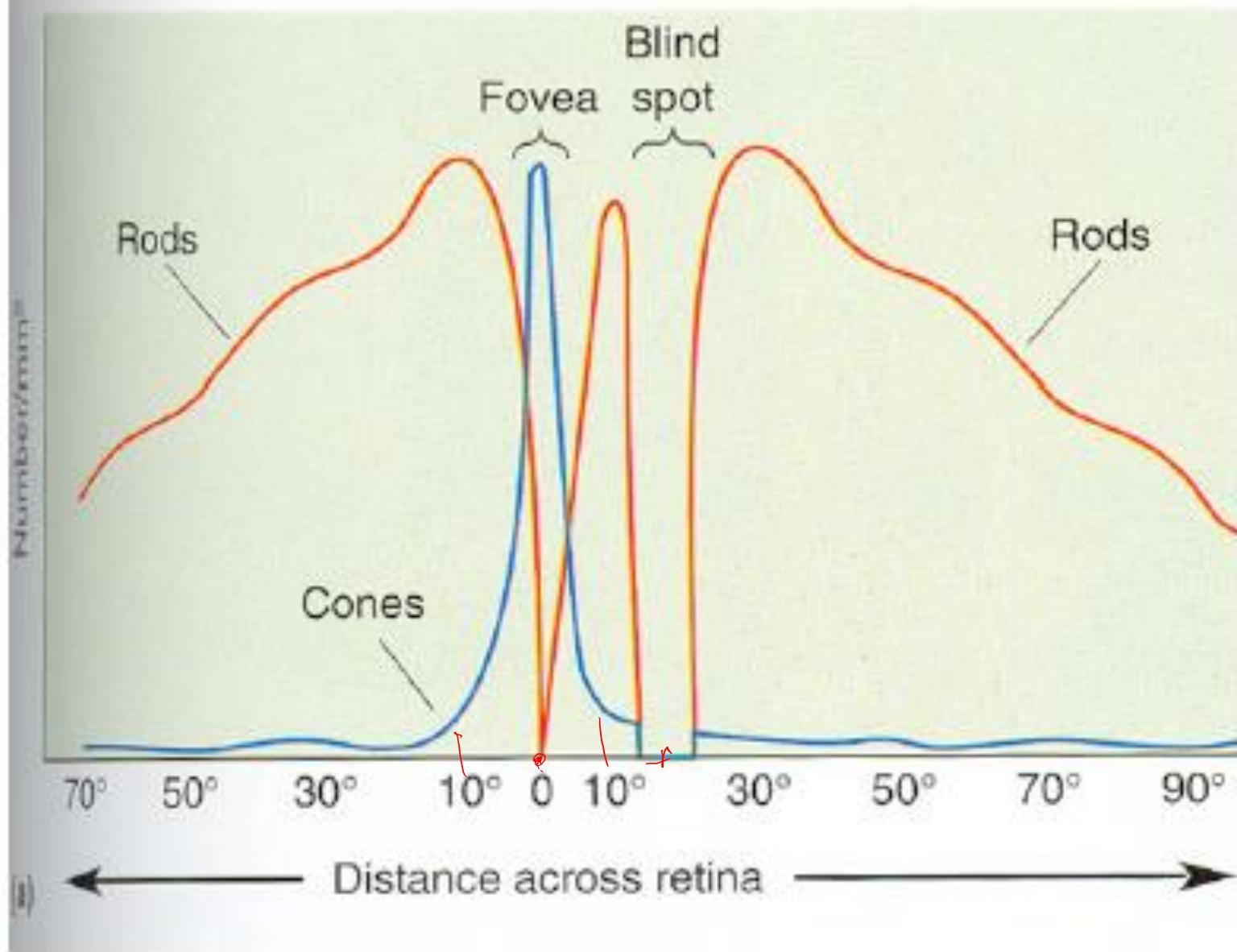
Retinal Layers



Transducers: Rods and Cones

- Convert light into electrical energy.
- In one eye:
 - 120 million rods
 - 7 million cones
- **Fovea:** a central region with a high density of cones. It has no rods.
- **Cones:** color sensitive (R, G & B)
- **Rods:** not color sensitive

Distribution of Rods and Cones



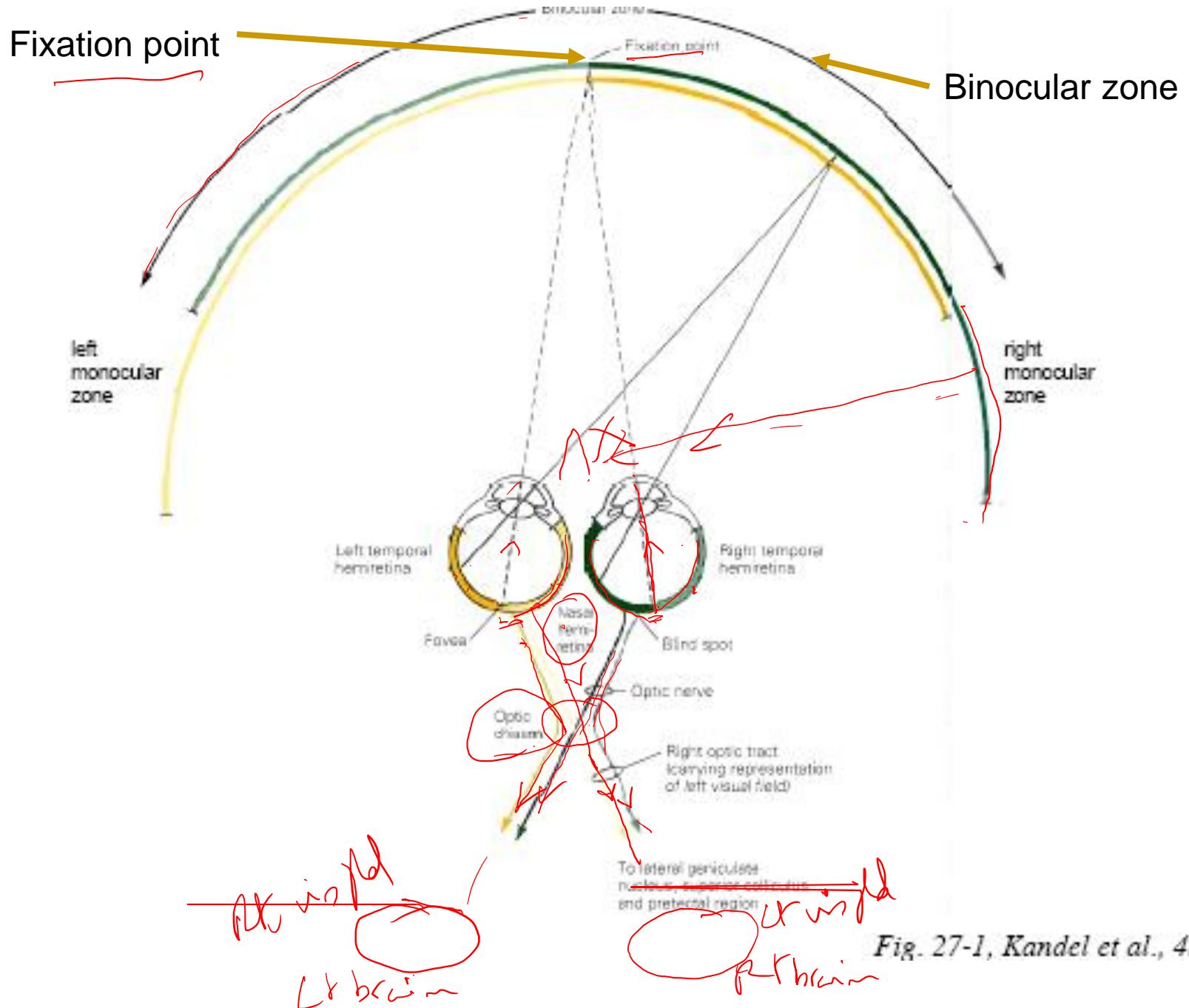
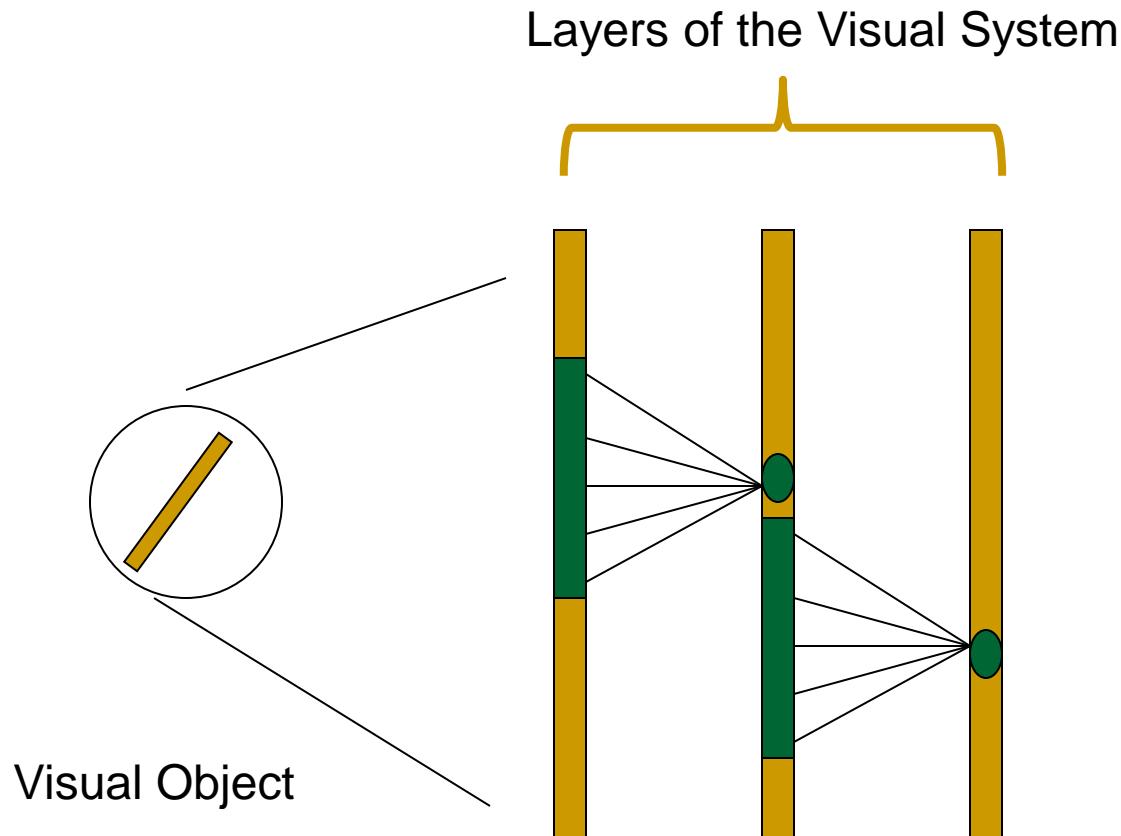
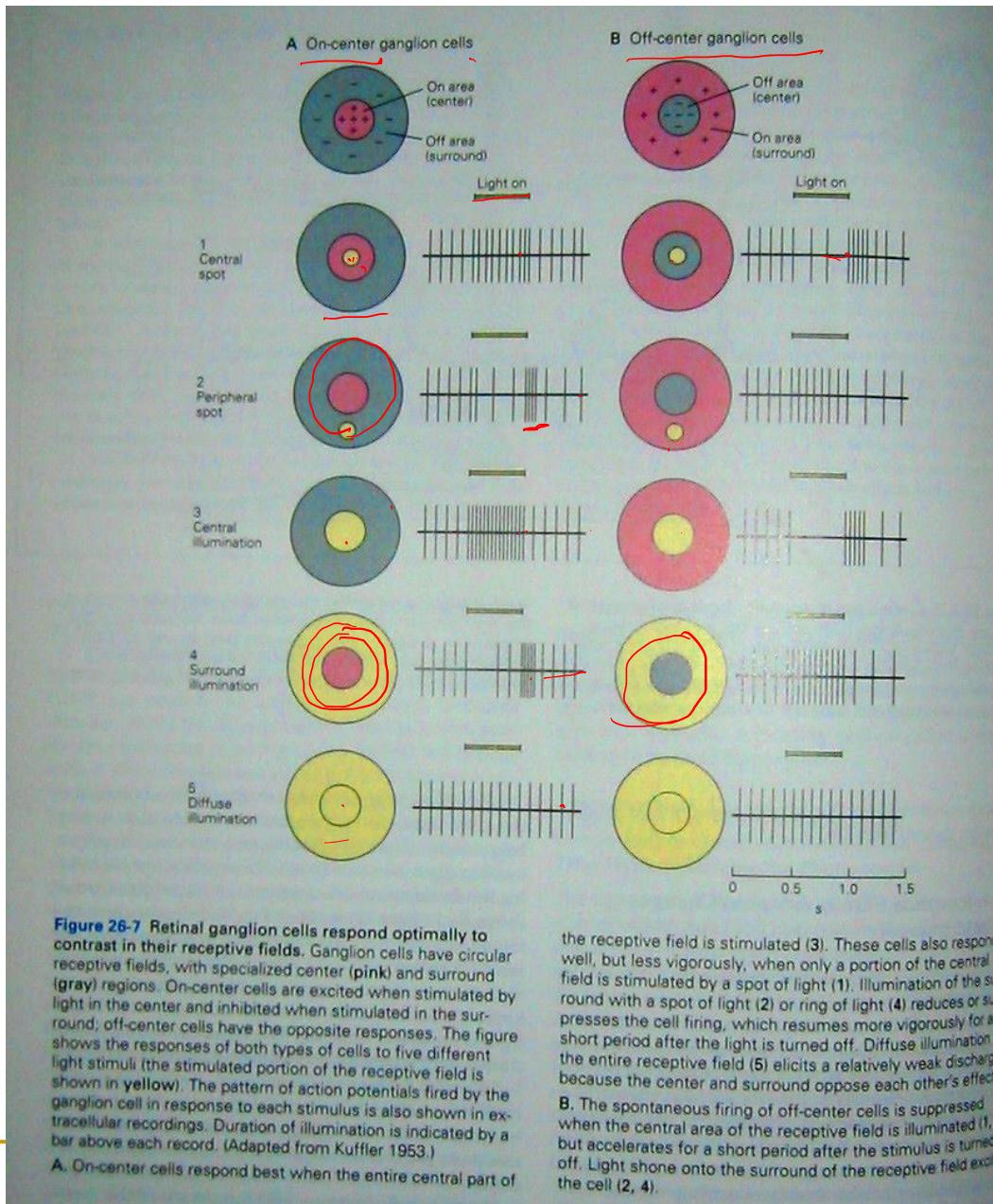
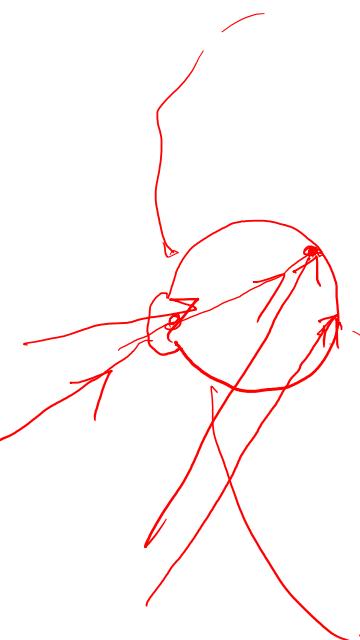


Fig. 27-1, Kandel et al., 4th edition

The Notion of a Receptive Field

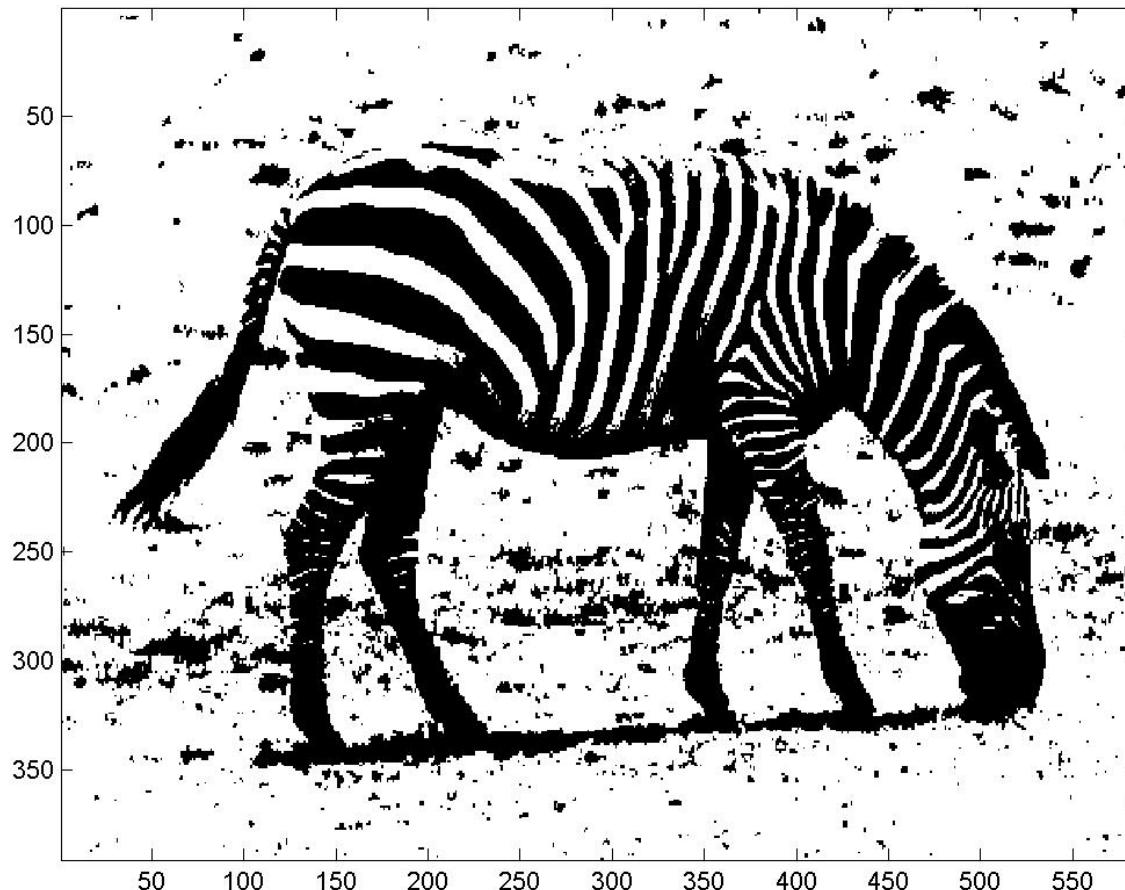


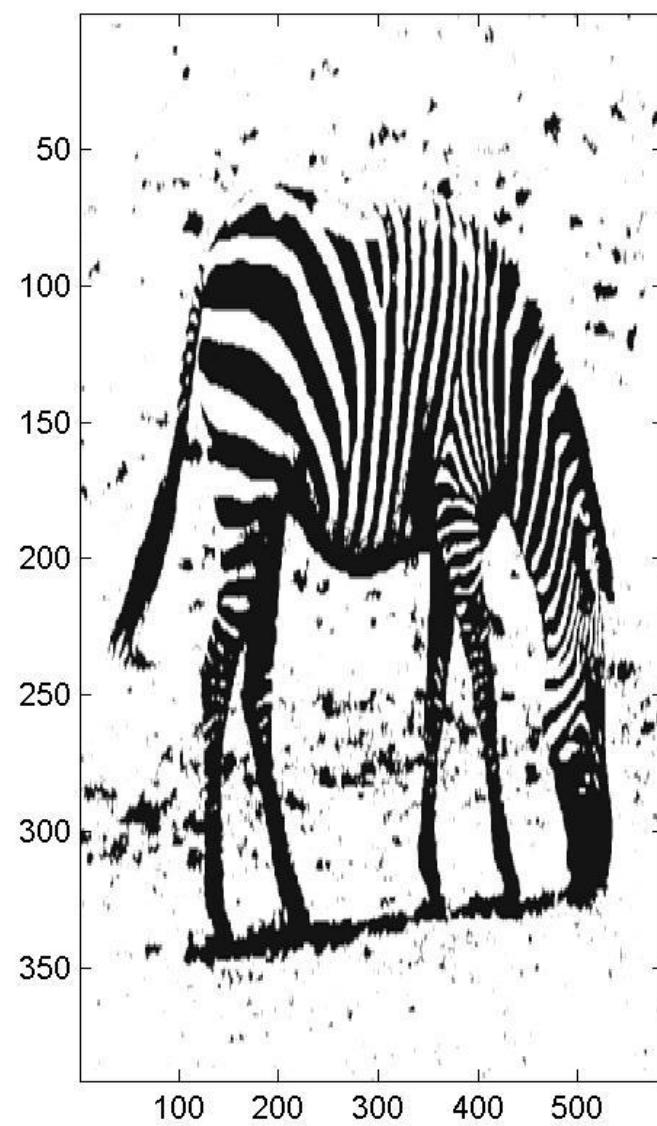


What do the Ganglion Cells do?

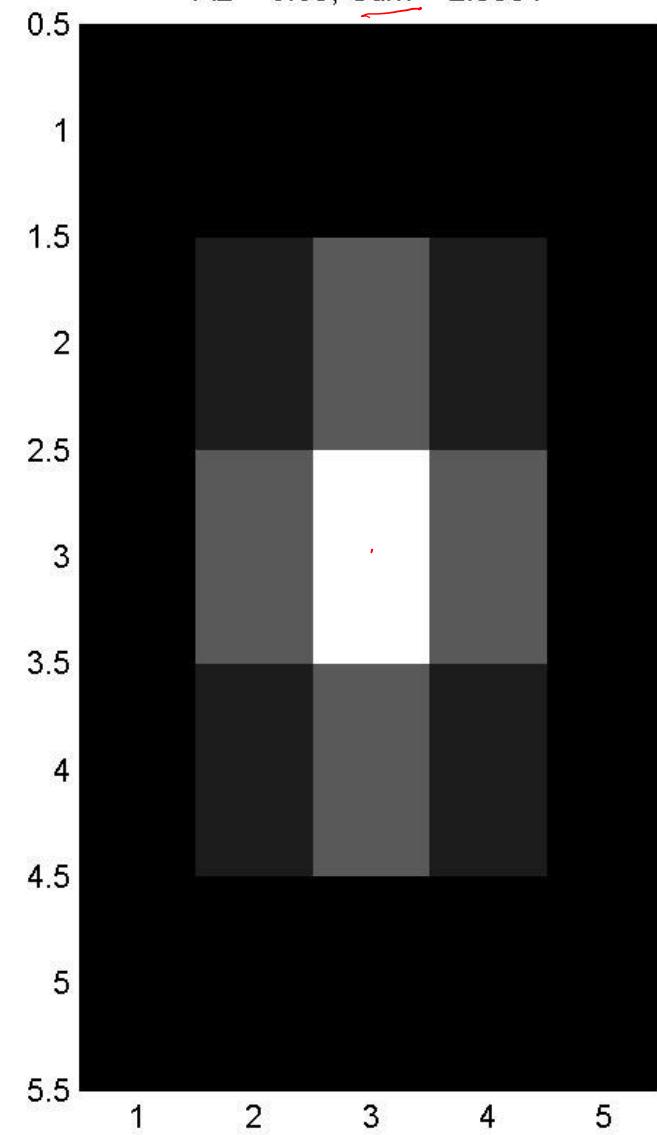
- Response of the ganglion cell layer as a convolution with a center-surround kernel
- Evaluate Response for various kernel parameters

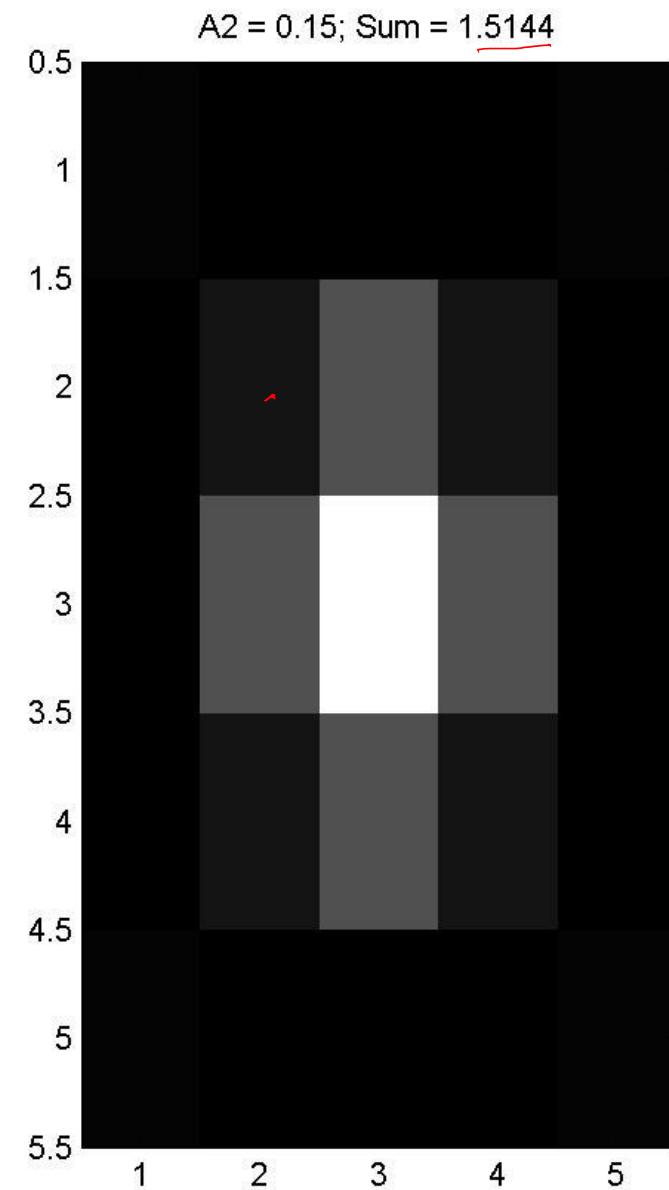
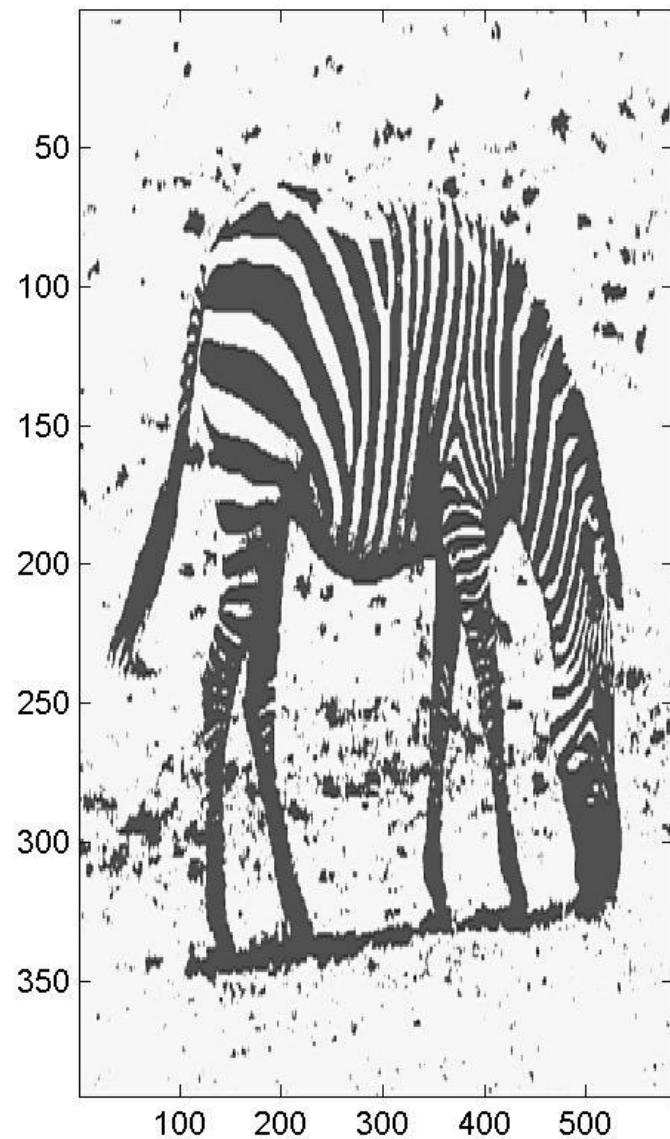
Original Image

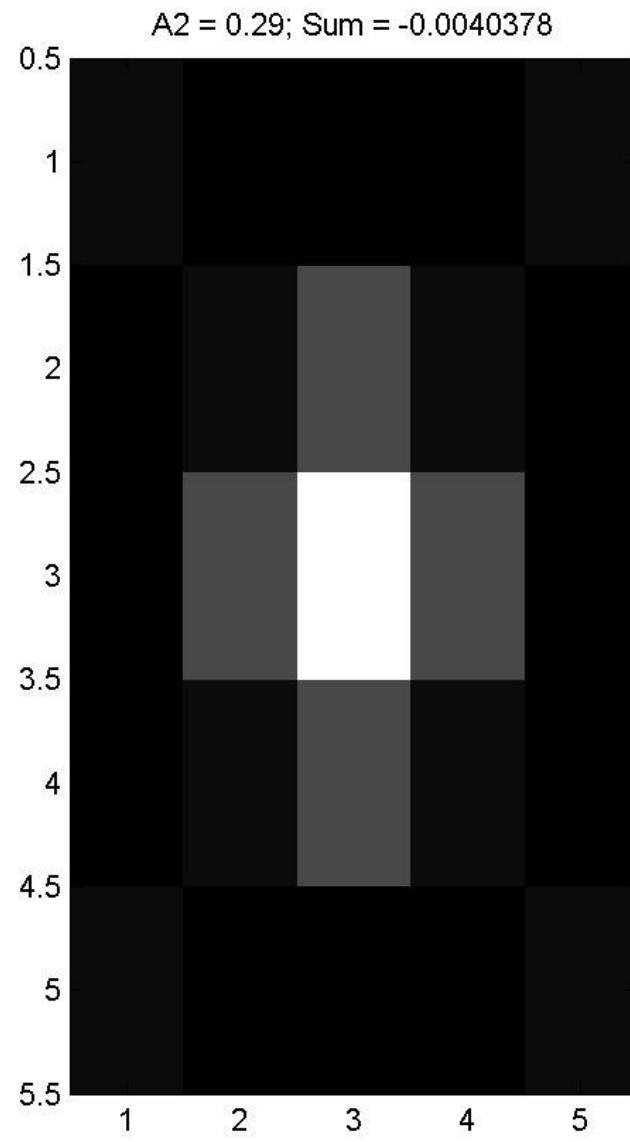
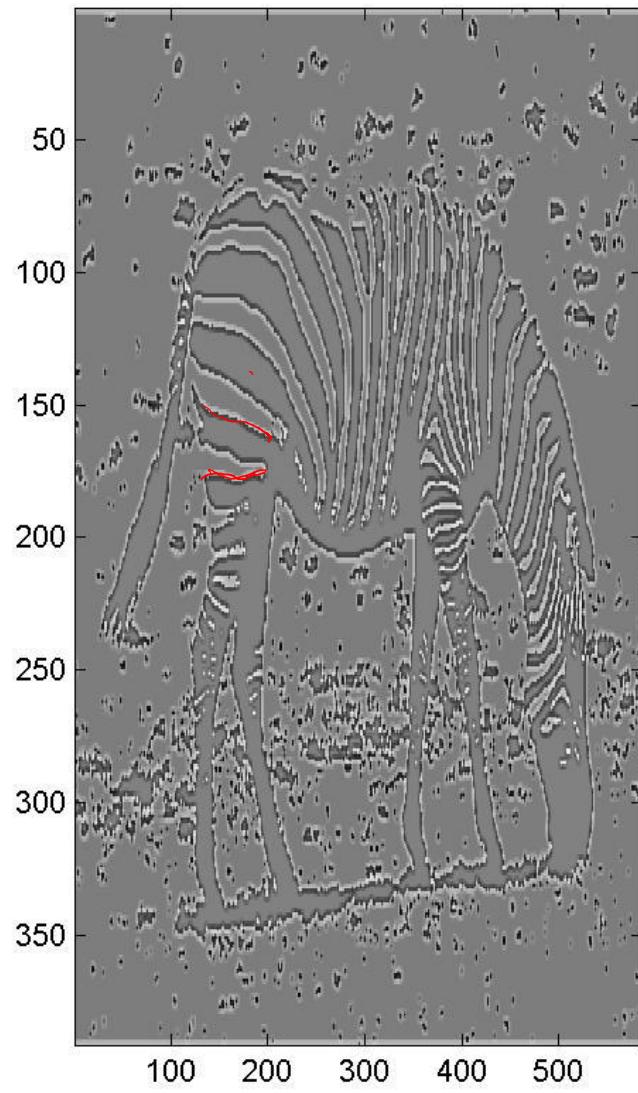


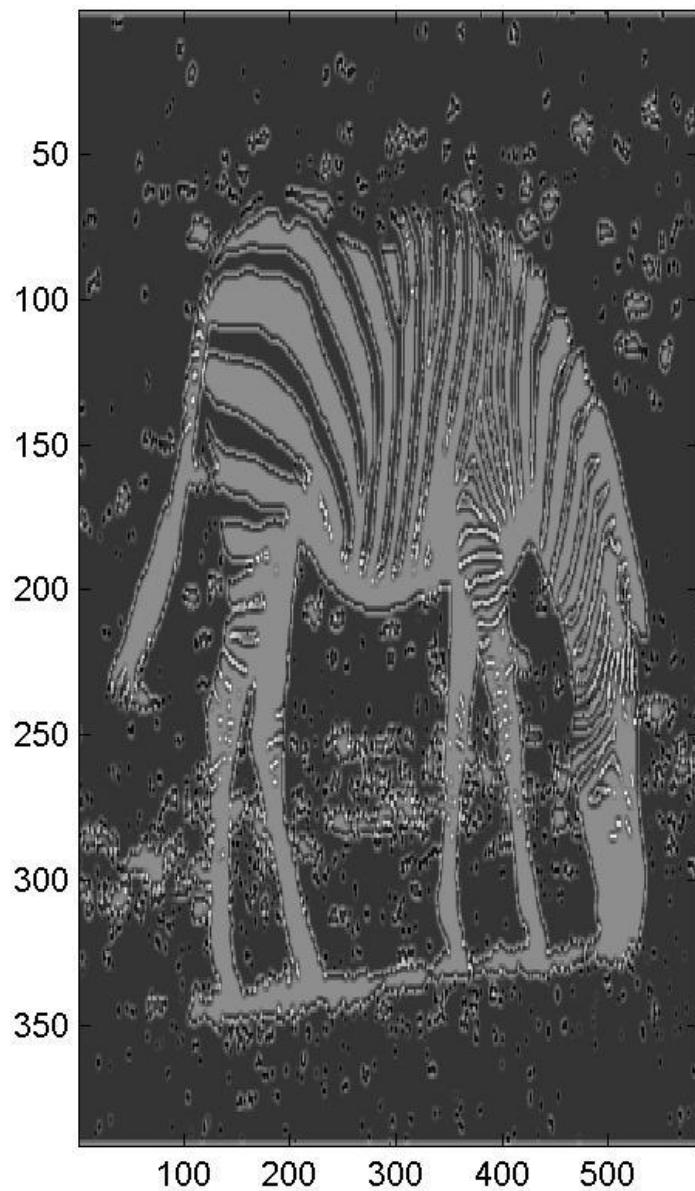


A2 = 0.05; Sum = 2.5991

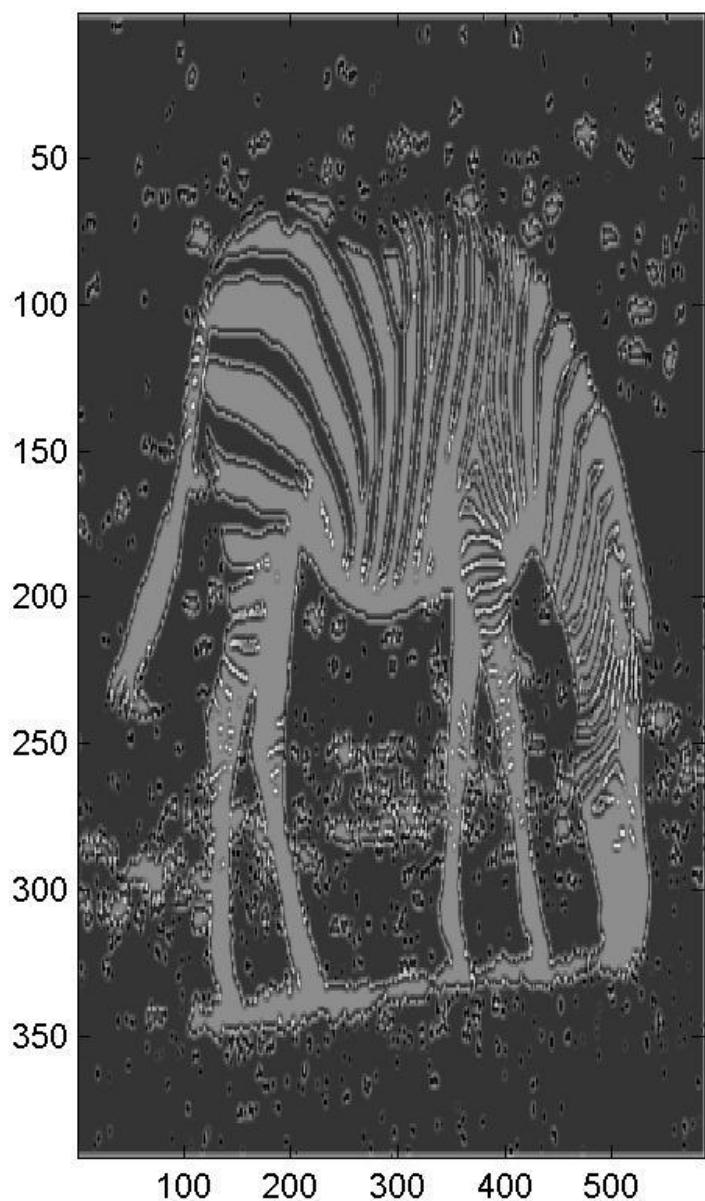


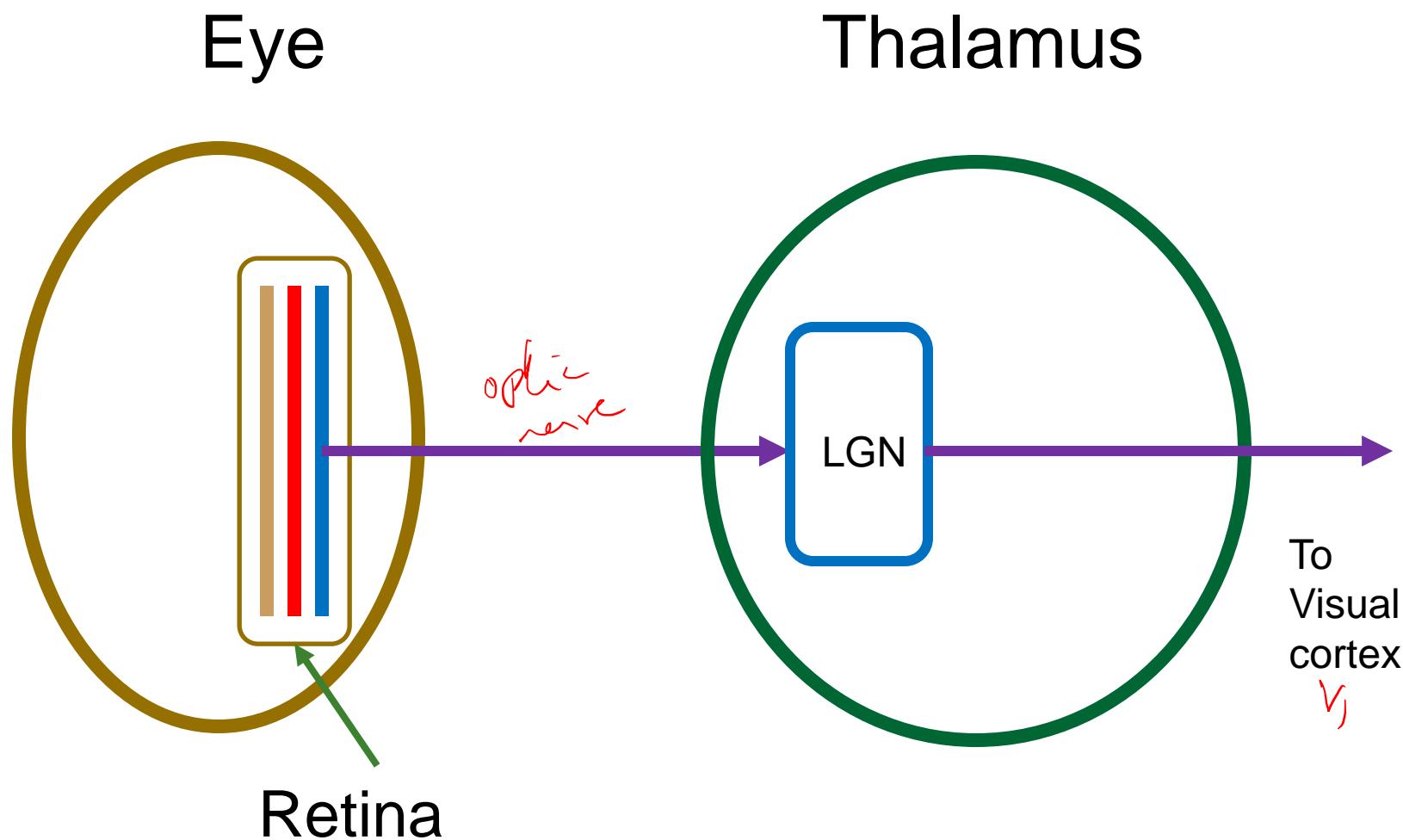






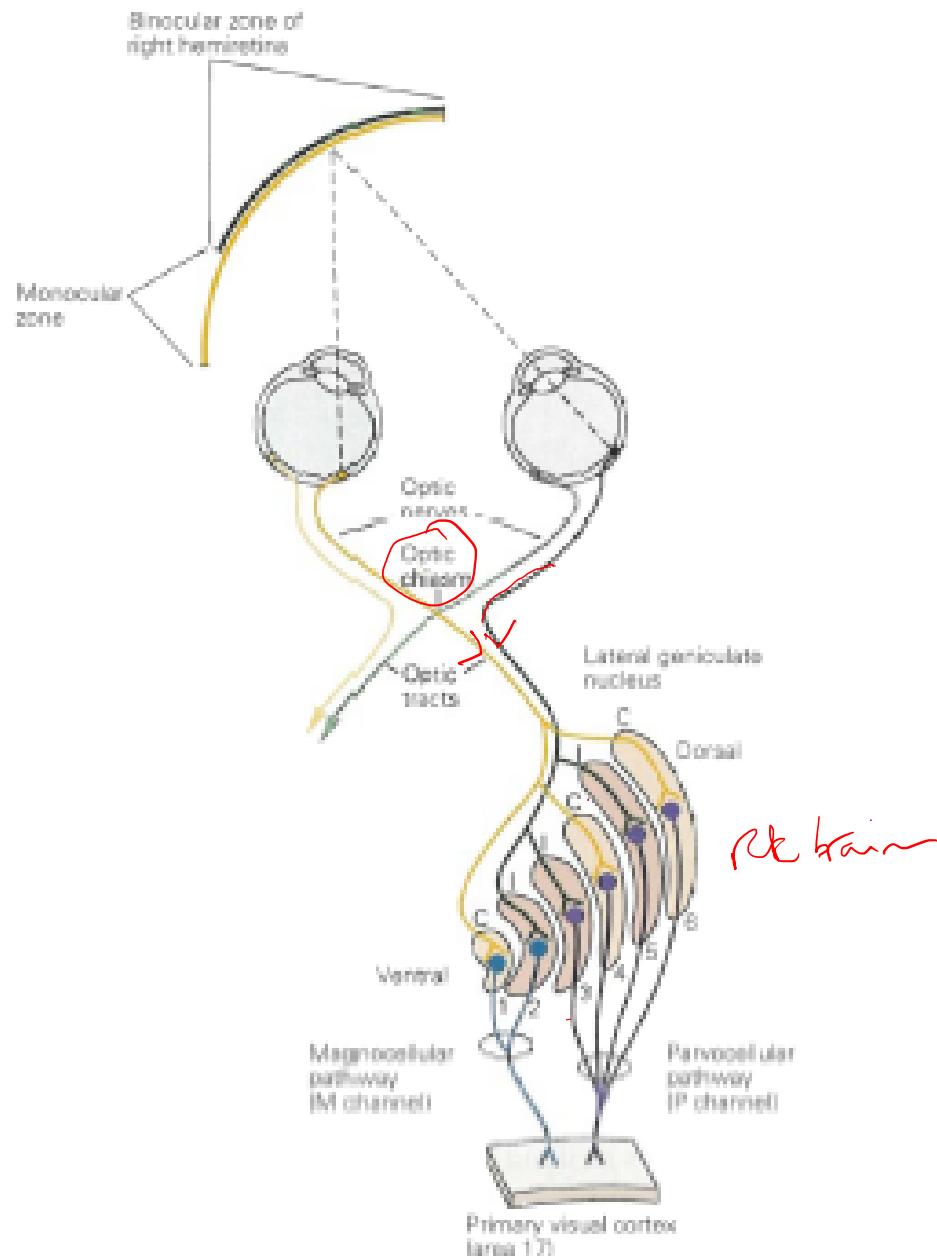
$A_2 = 0.35$; Sum = -0.65481



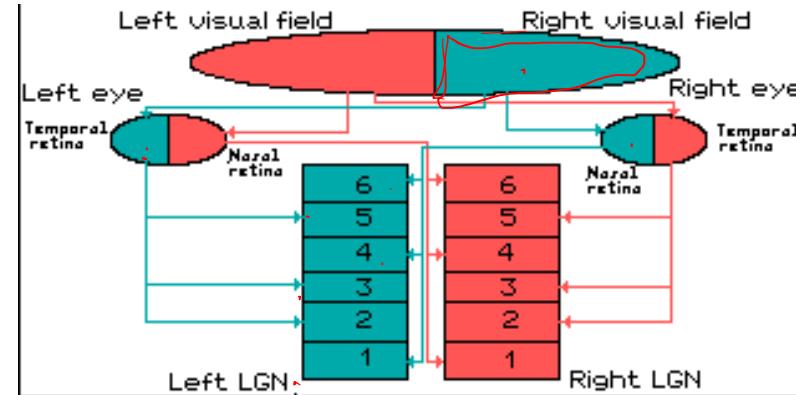
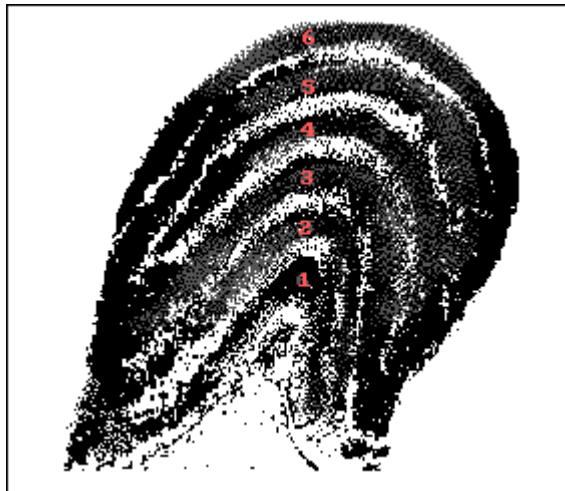


Segregation of Inputs into the LGN

- Magnocellular pathway
Layers 1 & 2
- Parvocellular pathway
Layers 3 - 6
- Contralateral input
Layers 1, 4, 6
- Ipsilateral input
Layers 2, 3, 5
- Point-to-point
topography of visual
field is preserved
between retina and LGN



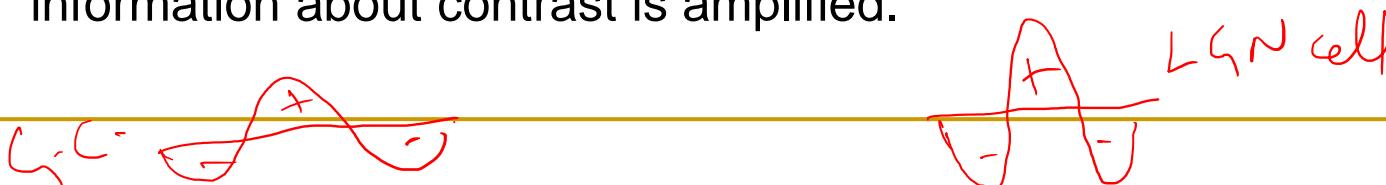
Inputs to LGN



- 1-2: Magnocellular layers; M ganglion cells project here.
- 3-6: parvocellular layers; P ganglion cells project here.
- Individual layer in LGN receives input from one eye only
- Fibers from contralateral nasal hemiretina project to layers 1, 4, 6
- Fibers from ipsilateral temporal hemiretina project to layers 2, 3, 5
- Each layer contains the representation of the contralateral hemifield
- Greater representation to Fovea than periphery
 - Nearly half the LGN cell mass represents the fovea and surrounding region
 - Magnification factor

Receptive Fields of LGN cells

- LGN cells have circular, center-surround receptive fields -- similar to those of retinal ganglion cells (1° dia)
- Like ganglion cells, both types of RFs (ON-Ctr/OFF-Surr & OFF-Ctr/ON-Surr) are present
- However, there is some segregation of the two types of RFs
 - magnocellular layers (1 and 2) each contain a mixture of cells with ON/OFF and OFF/ON receptive fields,
 - each of the parvocellular layers (3-6) contains only one or the other type of receptive field (two layers contain only ON/OFF and two contain only OFF/ON).
- The surrounds of LGN receptive fields also have stronger influences than ganglion cell surrounds. This means that the surrounds in LGN receptive fields are weighted more heavily, and, consequently, information about contrast is amplified.



Outputs from LGN

- M-pathways
 - Initial analysis of movement
 - Projections from M cells to V1
- P-pathways
 - Fine structure and color vision
 - Projections from P cells to V1
- Only 10-20% of inputs to LGN cells are from retina; the rest are from cortex (and retinal formation etc)

Function of LGN

- enhances information about contrast
- organizes information (e.g., eye of origin, colour, motion, form)
- modulates levels of processing with arousal (via the reticular activating system)
- receives feedback from higher areas (V1)

Reconstruction of Natural Scenes from Ensemble Responses in the Lateral Geniculate Nucleus

(Stanley, Li & Dan, 1999 – *J. Neuroscience*)

- Natural scenes were reconstructed from 177 LGN cells
- Quality of reconstruction at a given point:
 - depends on the number of cells
 - saturates at 6-8 pairs (ON/OFF) of cells

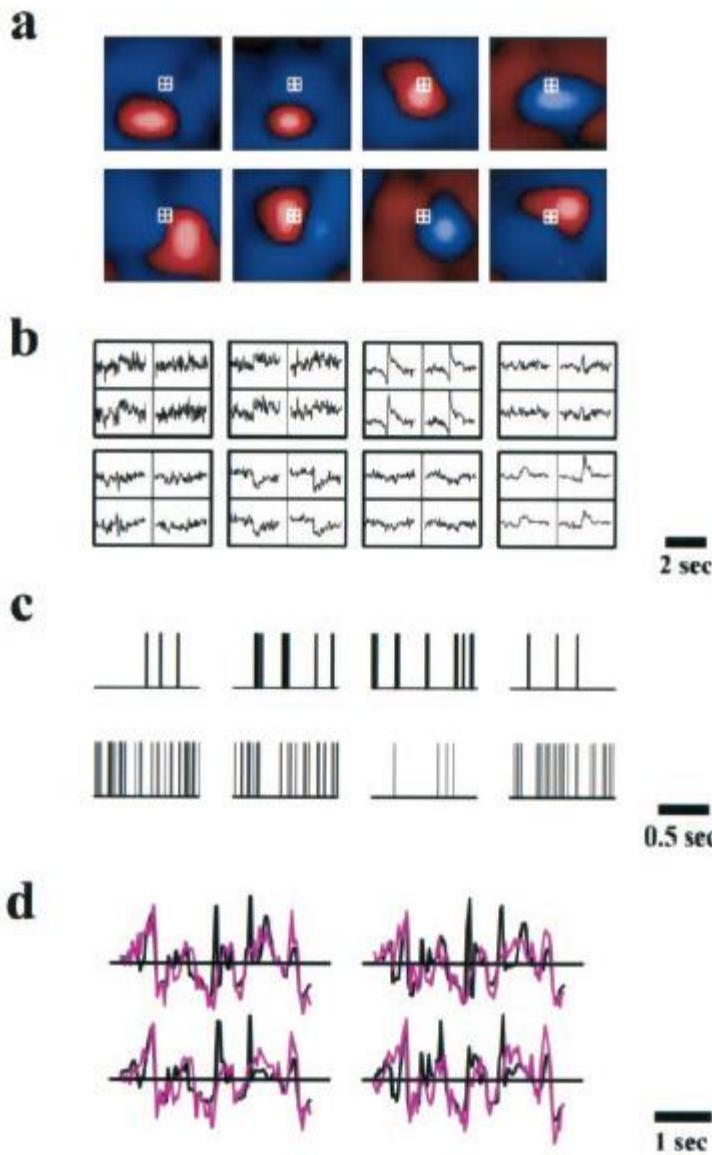


Figure 1. The procedure for reconstructing visual stimuli from the responses of multiple neurons. *a*, Receptive fields of eight neurons recorded simultaneously with multielectrodes. These receptive fields were mapped with white-noise stimuli and the reverse correlation method (Sutter, 1987; Reid et al., 1997). *Red*, On responses. *Blue*, Off responses. The brightest colors correspond to the strongest responses. The area shown is $3.6 \times 3.6^\circ$. The responses of these cells were used to reconstruct visual inputs at the four pixels (0.2% pixel) outlined with the *white squares*. *b*, Linear filters for input reconstruction. The eight blocks correspond to the eight cells shown in *a*. Shown in each block are the four filters from that cell to the four pixels outlined in *a*. They represent the linear estimates of the input signals at these pixels immediately preceding and following a spike of that cell. Each filter is 3.1-sec-long, with 1.55 sec before and 1.55 sec after the spike. *c*, Spike trains of the eight neurons in response to movie stimuli. *d*, The actual (black) and the reconstructed (magenta) movie signals at the four pixels outlined in *a*. Unlike white noise, natural visual signals exhibit more low-frequency, slow variations than high-frequency, fast variations. Such temporal features are well captured by the reconstruction.

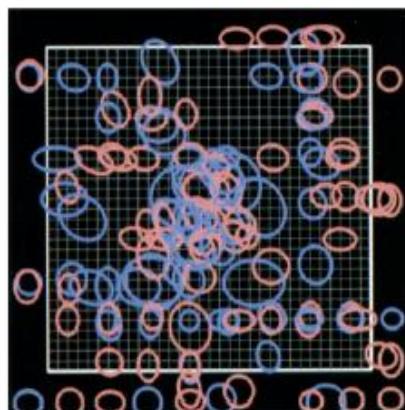
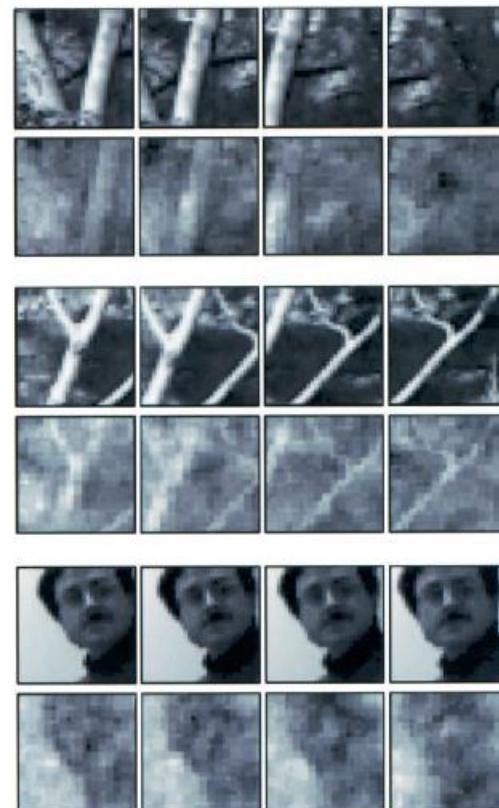
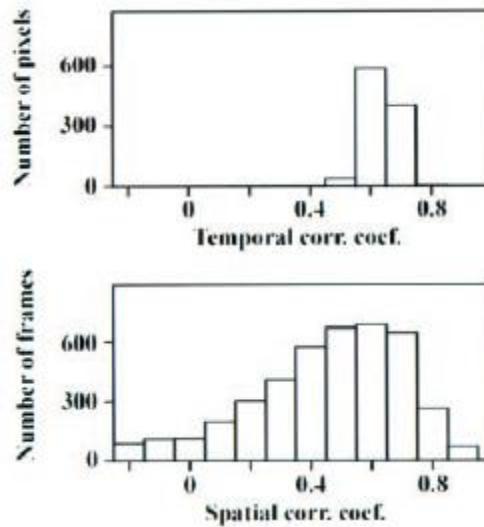
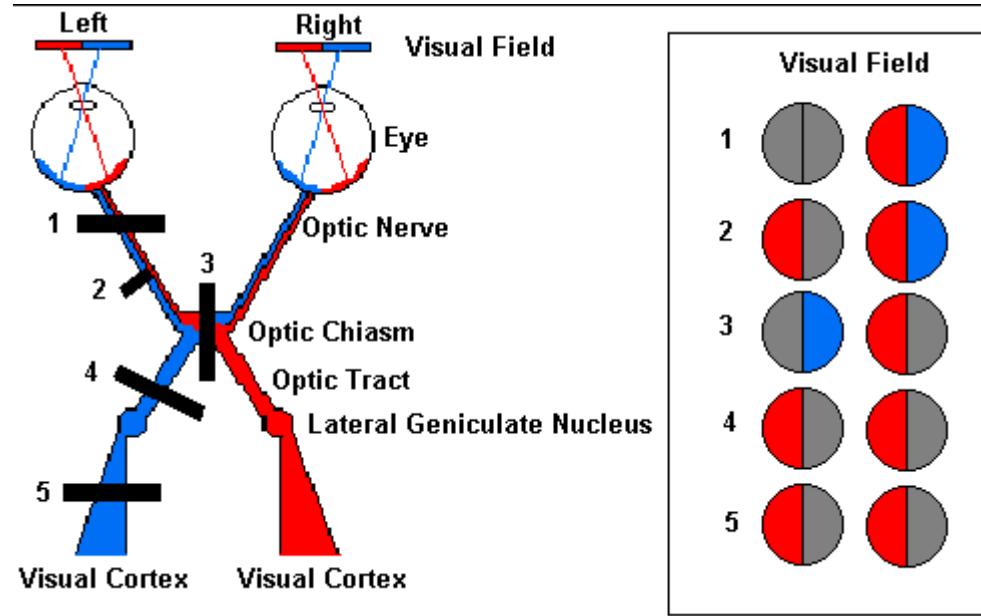
a**b****c**

Figure 2. Reconstruction of natural scenes from the responses of a population of neurons. *a*, Receptive fields of 177 cells used in the reconstruction. Each receptive field was fitted with a two-dimensional Gaussian function. Each ellipse represents the contour at one SD from the center of the Gaussian fit. Note that the actual receptive fields (including surround) are considerably larger than these ellipses. *Red*, On center. *Blue*, Off center. An area of 32×32 pixels (0.2°/pixel) where movie signals were reconstructed is outlined in white. The grid inside the white square delineates the pixels. *b*, Comparison between the actual and the reconstructed images in an area of $6.4 \times 6.4^\circ$ (*a*, white square). Each panel shows four consecutive frames (interframe interval, 31.1 msec) of the actual (*top*) and the reconstructed (*bottom*) movies. *Top panel*, Scenes in the woods, with two trunks of trees as the most prominent objects. *Middle panel*, Scenes in the woods, with smaller tree branches. *Bottom panel*, A face at slightly different displacements on the screen. *c*, Quantitative comparison between the reconstructed and the actual movie signals. *Top*, Histogram of temporal correlation coefficients between the actual and the reconstructed signals (both as functions of time) at each pixel. The histogram was generated from 1024 (32×32) pixels in the white square. *Bottom*, Histogram of spatial correlation coefficients between the actual and the reconstructed signals (both as functions of spatial position) at each frame. The histogram was generated from 4096 frames (512 frames per movie; 8 movies).



Damage at site #1: this would be like losing sight in the left eye.

The entire left optic nerve would be cut and there would be a total loss of vision from the left eye.

Damage at site #2: partial damage to the left optic nerve.

Here, information from the nasal visual field of the left eye (temporal part of the left retina) is lost.

Damage at site #3: the optic chiasm would be damaged.

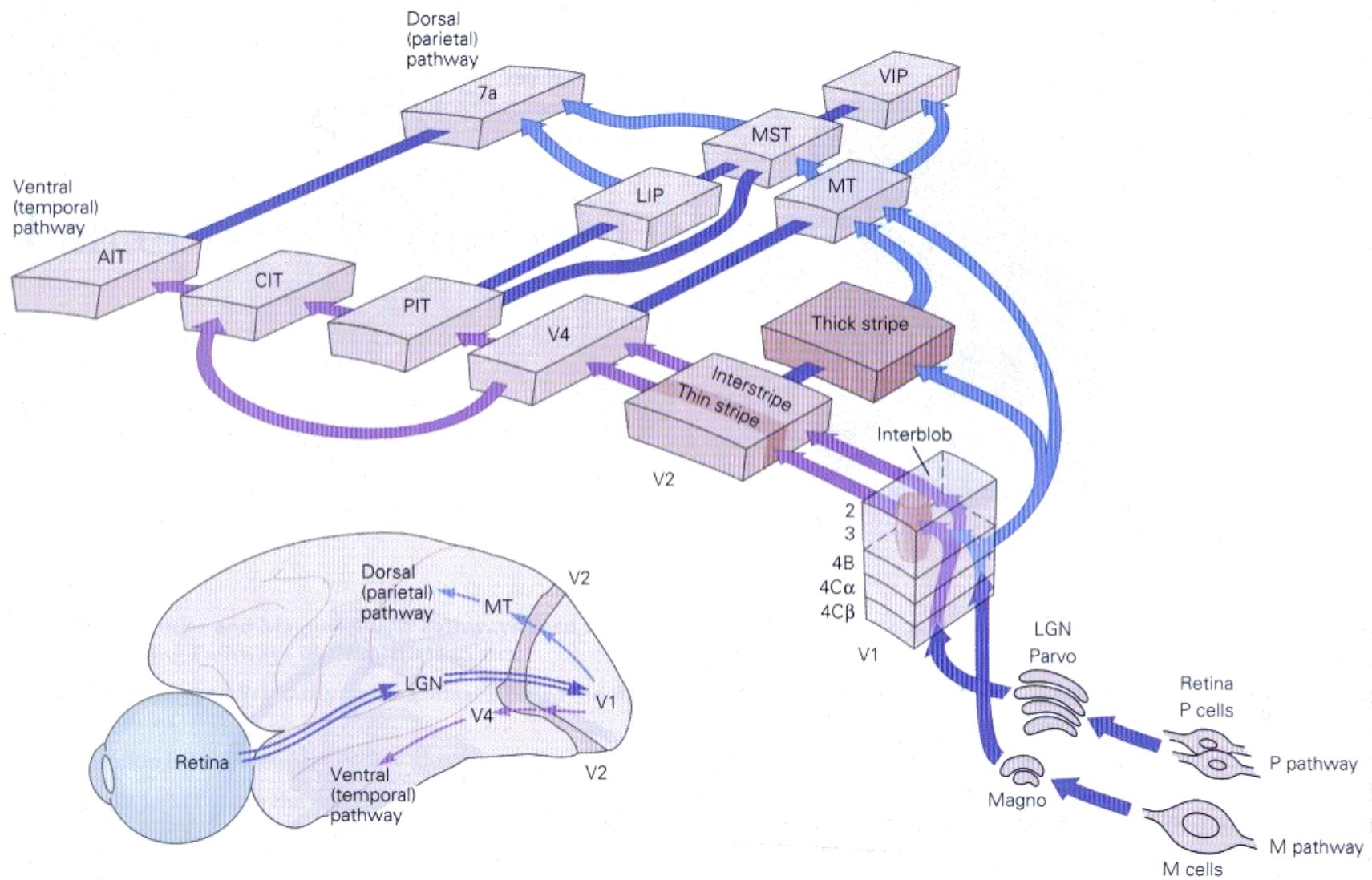
In this case, the temporal (lateral) portions of the visual field would be lost. The crossing fibers are cut in this example.

Damage at site #4 and #5: damage to the optic tract (#4) or the fiber tract from the LGN to the cortex (#5) can cause identical visual loss.

In this case, loss of vision of the right side.

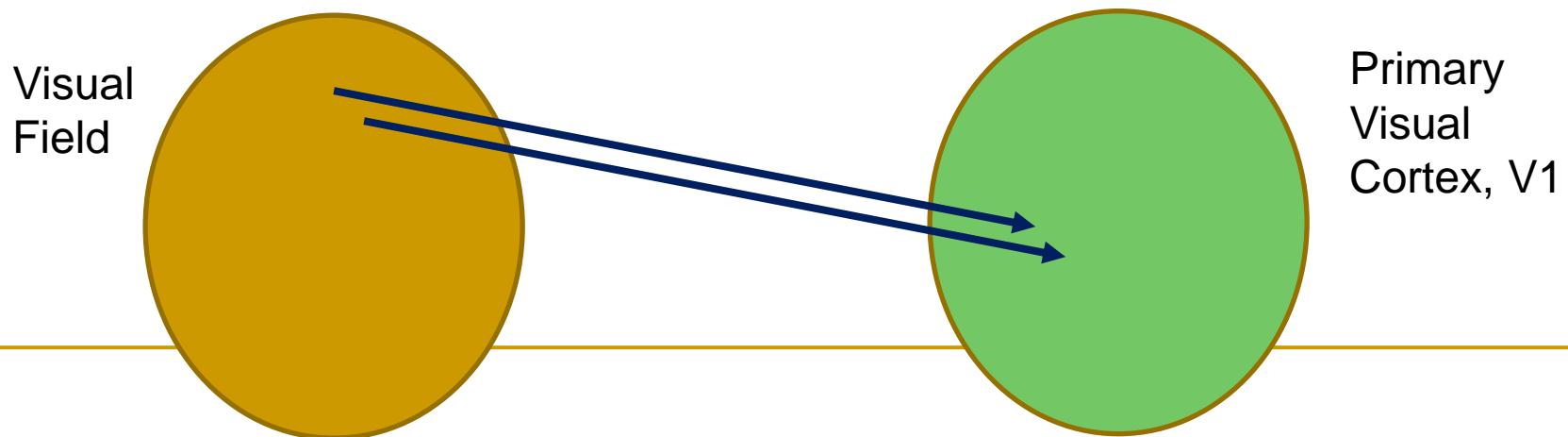
Primary Visual Cortex

V1-Striate Cortex



V1

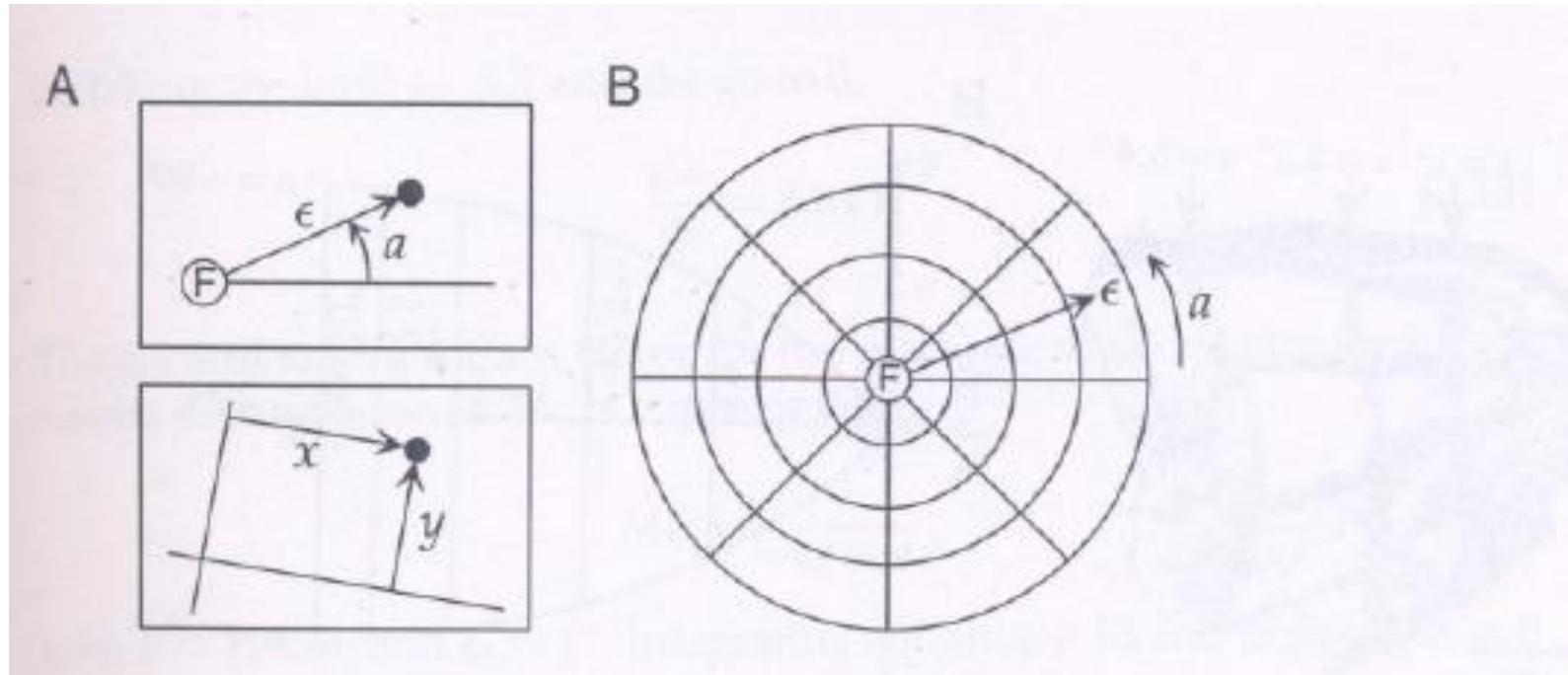
- First stopover of visual information in cortex
- Retinotopic map:
 - Retinal information is mapped onto V1 such that nearby points in the visual field are mapped onto nearby neurons in V1



Visual Field

- Consider the visual field as a sphere with the viewer at the center
- “North pole” of the sphere is the fixation point; this point is mapped onto fovea
- Latitude is called “eccentricity”, ε ($0\text{--}70^\circ$)
- Longitude is called “azimuth,” α (-90° to 90°)

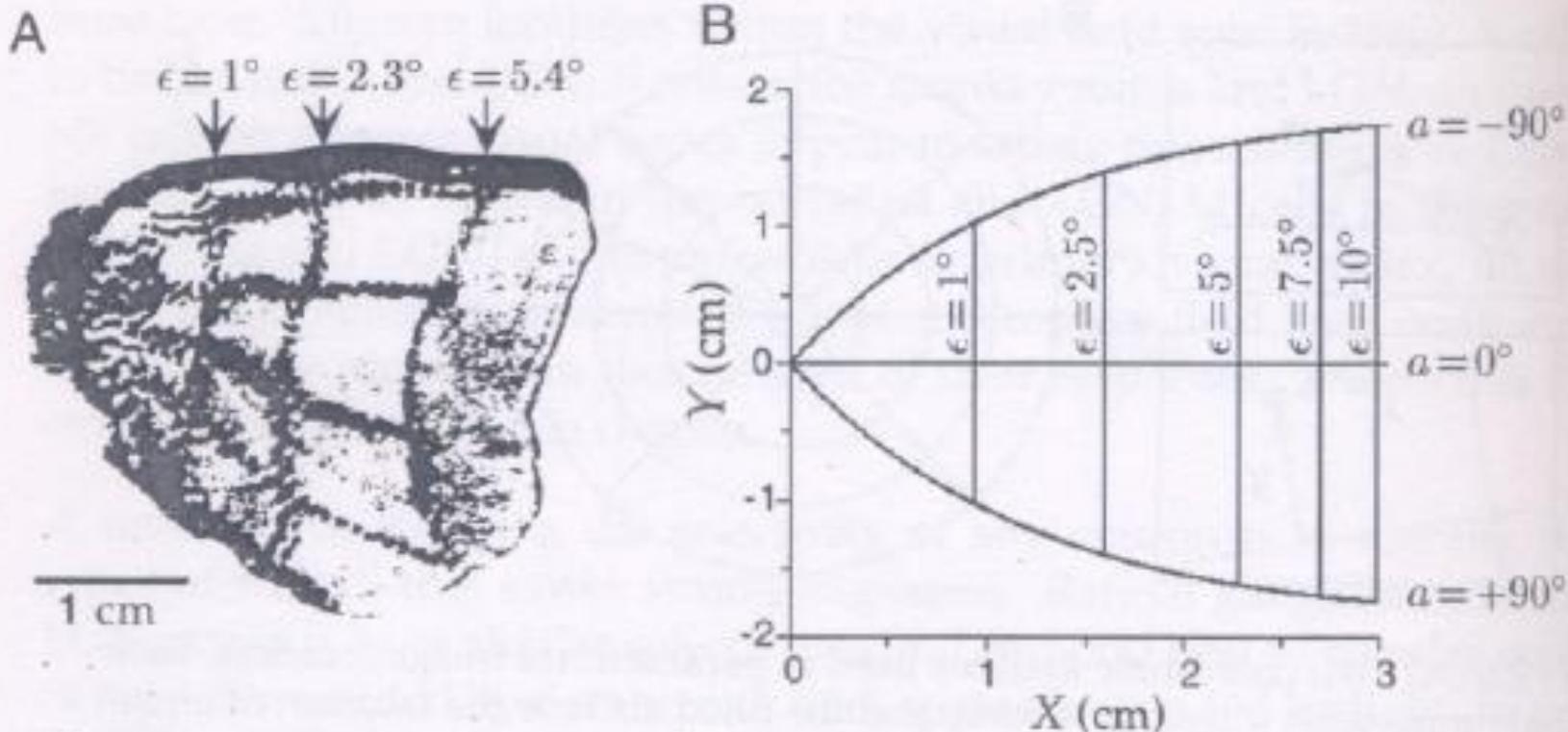
Visual field



Experiment to find Retinotopic Map

- “Bull’s-eye” pattern is displayed on a screen
- Pattern of activity in V1 is imaged by using a radioactive glucose; imaging reveals which neurons are active (taking up glucose)
- Experiment performed on monkey

Retinotopic Map



Retinotopic Map Features

- Vertical lines correspond to circles in the image
- Roughly horizontal lines correspond to radial lines
- Fovea is represented at the leftmost pole
- Azimuthal angles are positive in lower half, and negative in the upper half

Responses of neurons in V1

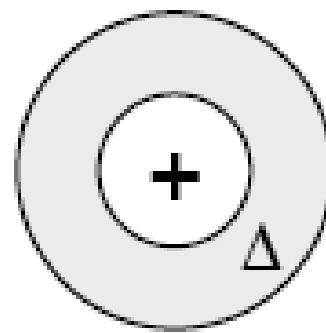
- Work of Hubel & Wiesel at Harvard in '60s.
 - Neurons in V1 respond to oriented bars and edges and not to spots.
 - Some neurons also respond to moving bars.
 - Different neurons respond to different orientations. Within a dia of 1mm, all orientations are represented.

Simple and Complex Cells

- Simple cells: respond to an oriented line present in the center of their RF
- Complex cells: respond to an oriented line present almost anywhere in their RF
- End-stopping: some cells respond best when the line is not too long.

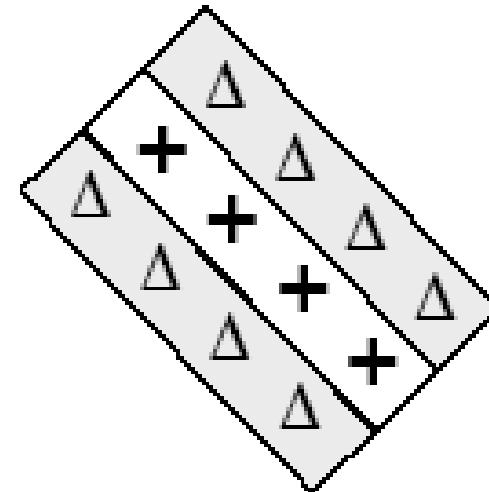
Transformation of Receptive Field Properties from LGN to Primary Visual Cortex

LGN
Neuron



Circular Receptive Field
(e.g., on-center, off-surround)

“Simple Cell”
in Cortex



Rectangular Receptive Field
(e.g., on-center, off-flanks)

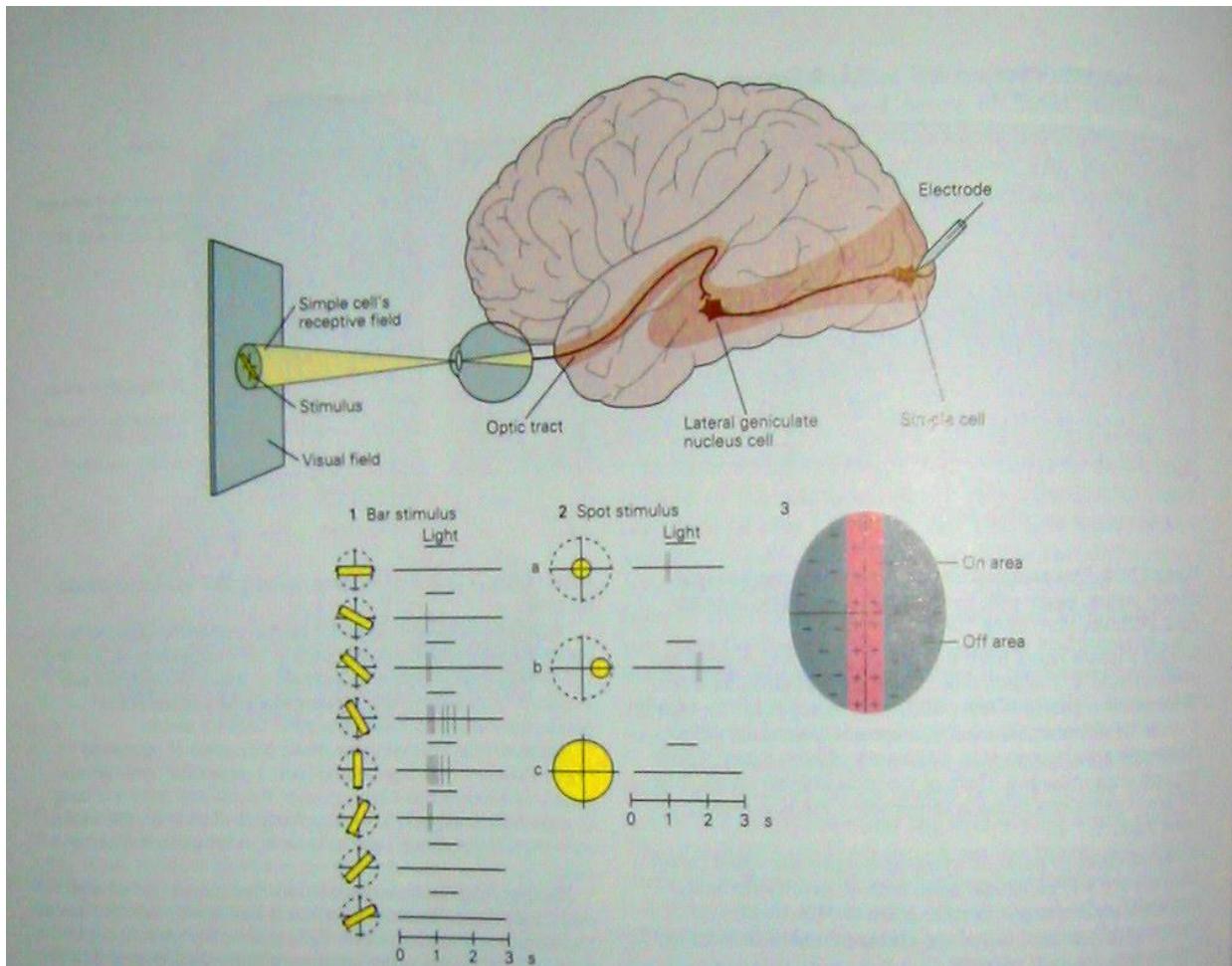


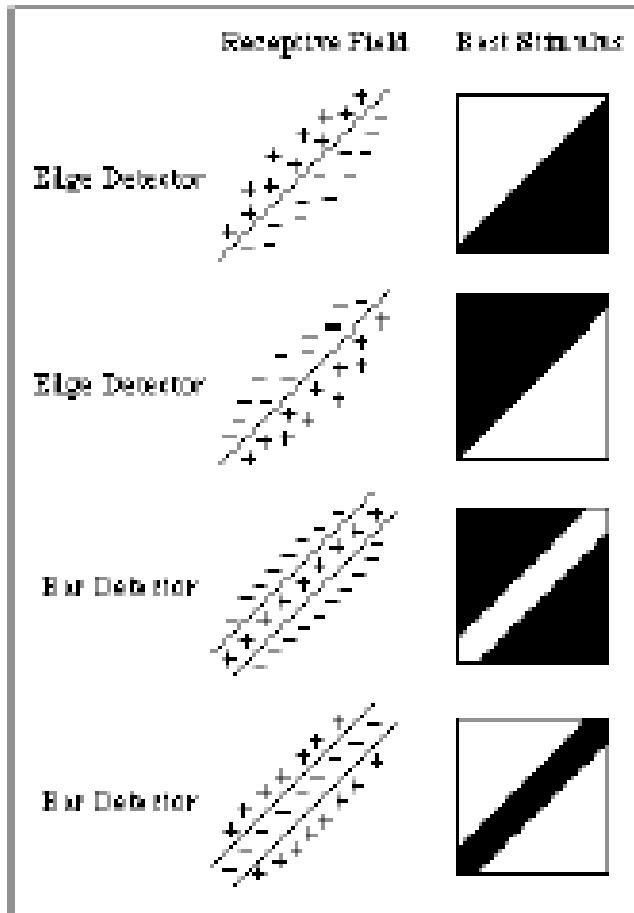
Figure 27-11 Receptive field of a simple cell in the primary visual cortex. The receptive field of a cell in the visual system is determined by recording activity in the cell while spots and bars of light are projected onto the visual field at an appropriate distance from the fovea. The records shown here are for a single cell. Duration of illumination is indicated by a line above each record of action potentials. (Adapted from Hubel and Wiesel 1959 and Zeki 1993.)

1. The cell's response to a bar of light is strongest if the bar of light is vertically oriented in the center of its receptive field.

2. Spots of light consistently elicit weak responses or no response. A small spot in the excitatory center of the field elicits only a weak excitatory response (a). A small spot in the inhibitory area elicits a weak inhibitory response (b). Diffuse light produces no response (c).

3. By using spots of light, the excitatory or "on" areas (+) and inhibitory or "off" areas (-) can be mapped. The map of the responses reveals an elongated "on" area and a surrounding "off" area, consistent with the optimal response of the cell to a vertical bar of light.

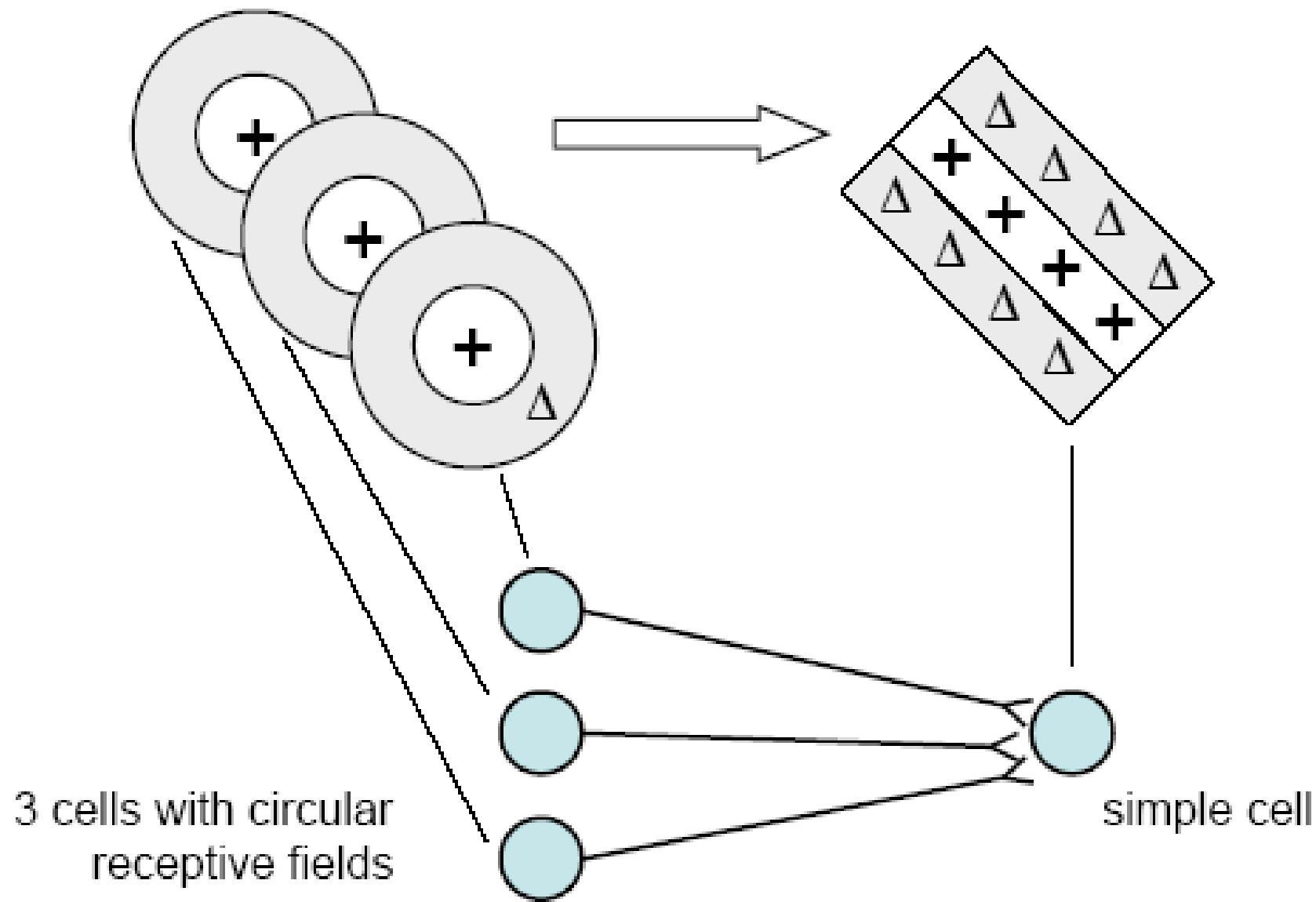
Examples of Simple Cell Receptive Fields



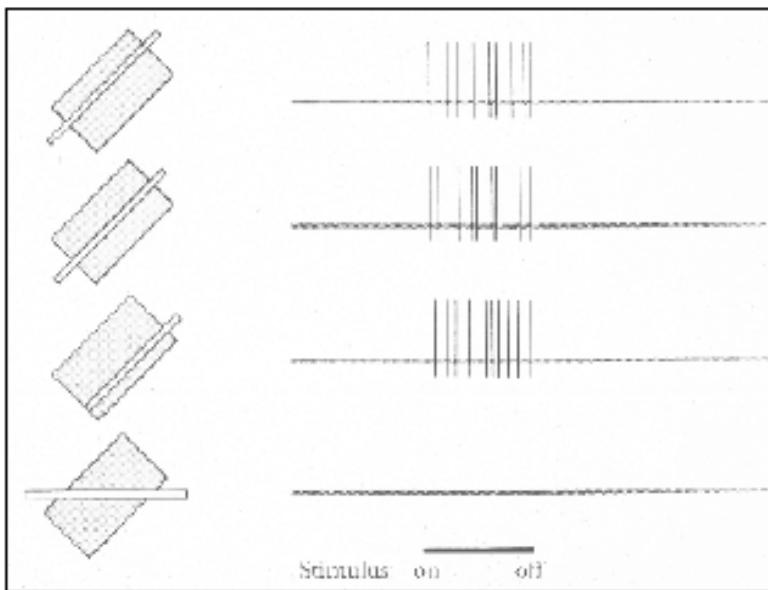
<http://www.cquest.utoronto.ca/psych/psych2800/ch4/orientSelac.html>

Simple cells can be used to detect edges and contours

Hypothetical Wiring Diagram for Generating a Simple Cell Receptive Field



Complex Cell Receptive Field Properties



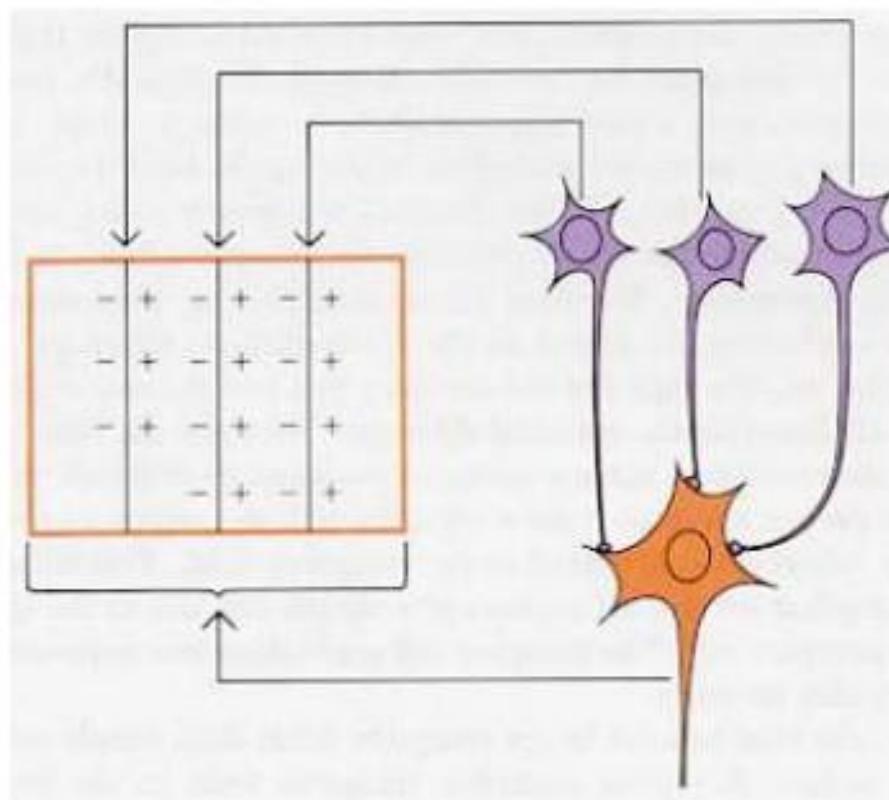
<http://www-psych.stanford.edu/~lera/psych115s/notes/lecture3/figures.html>

	Receptive Field	Best Orientation
Orientation Detector	$\begin{matrix} - & + & + & - & + \\ - & - & - & + & - \\ + & + & + & + & + \\ + & - & - & + & - \\ + & - & + & - & + \end{matrix}$	

<http://www.cquest.utoronto.ca/psych/psy280f/ch4/orientSelc.html>

- “On” and “off” regions throughout receptive field
- Orientation selective

Hypothetical Wiring Diagram for Generating a Complex Cell's Receptive Field



<http://neuro.med.harvard.edu/site/dh/b18.htm>

Construction of complex cell receptive field via
input from multiple simple cells

Schematic of Orientation Selectivity in the Primary Visual Cortex

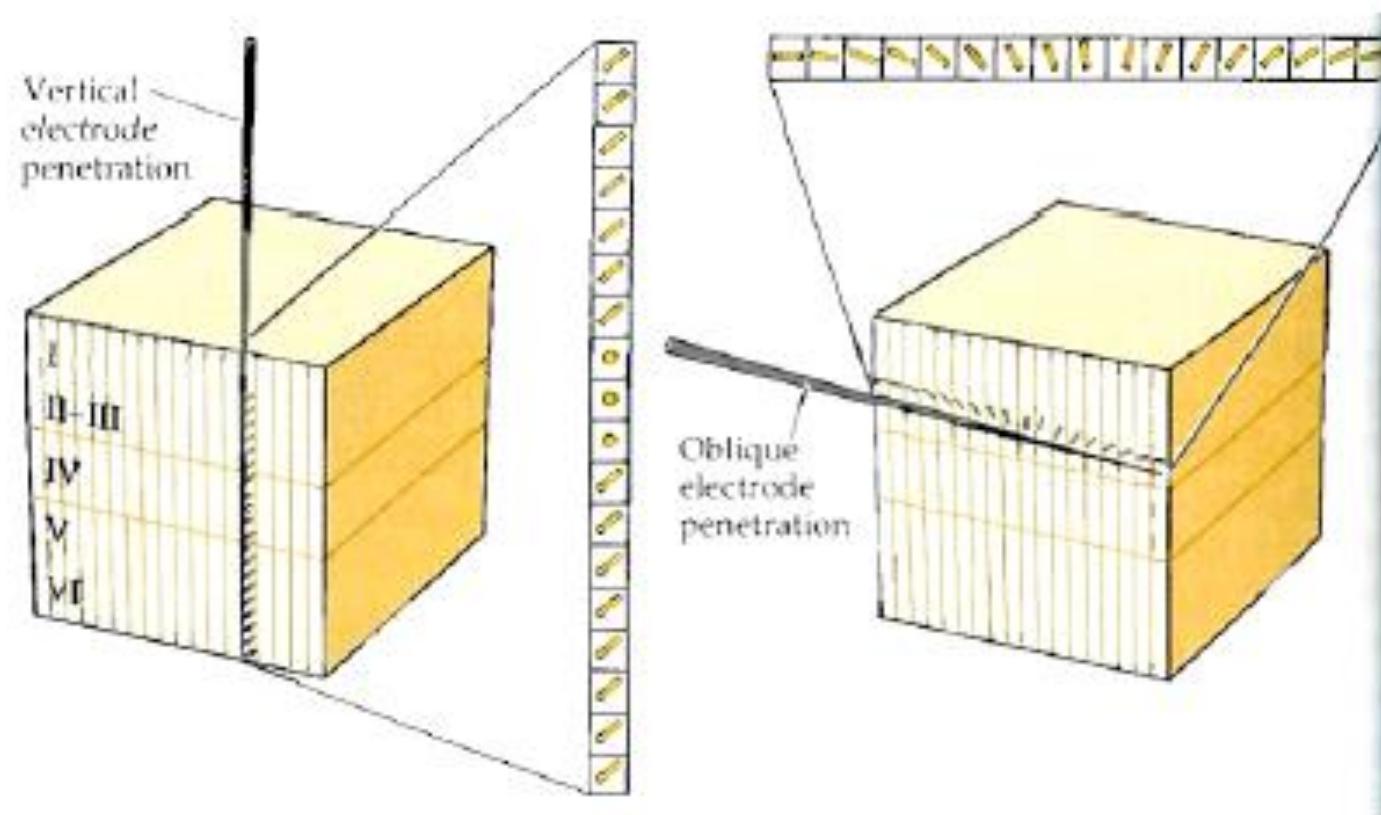
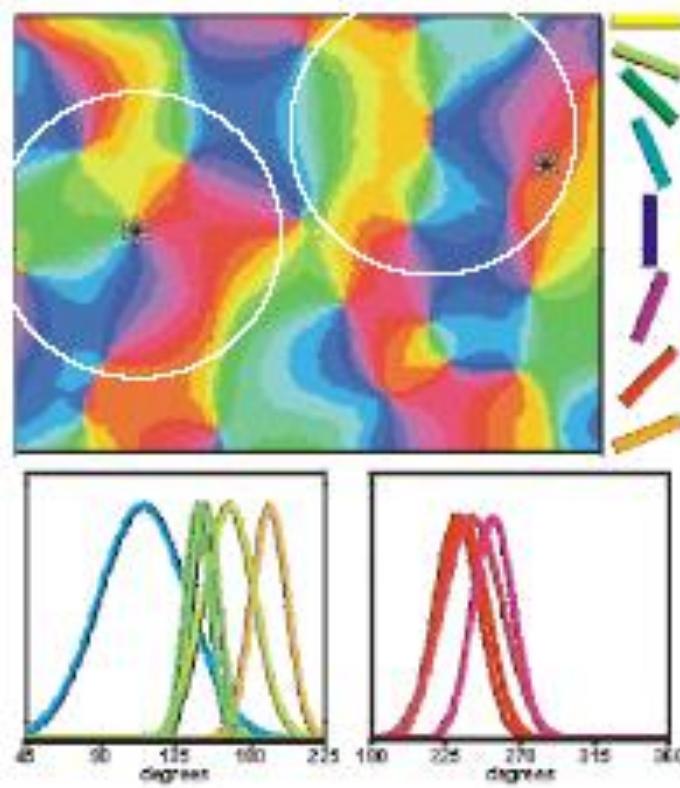


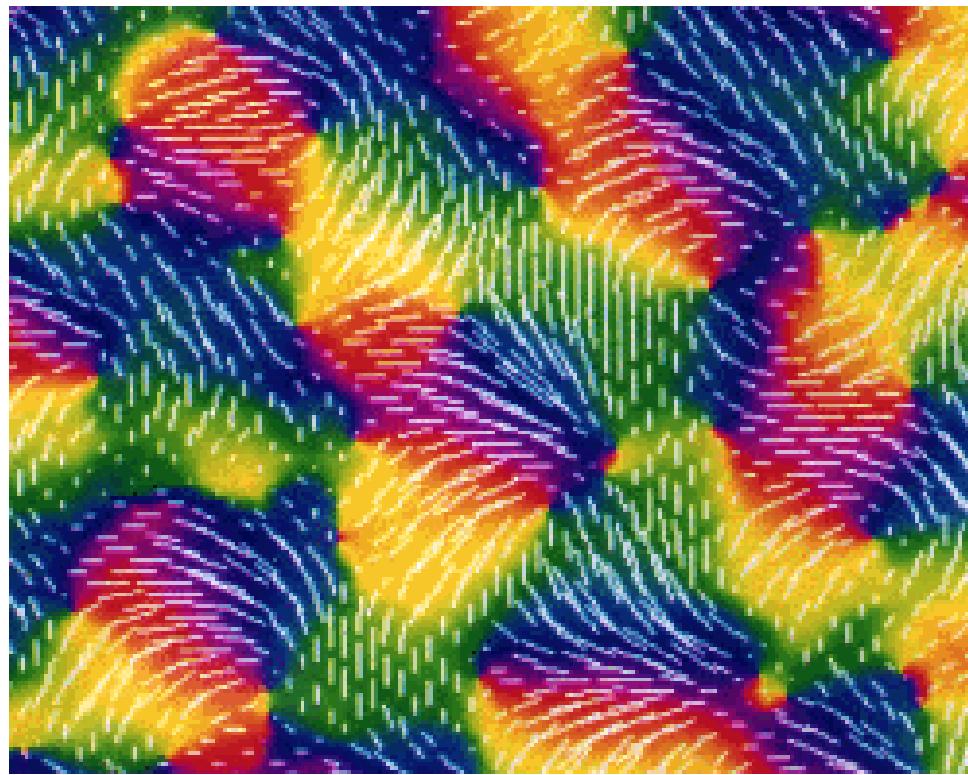
Fig. 11.12, Purves et al., Neuroscience, 3rd edition

- Oblique/tangential penetration reveals progressive change in orientation selectivity
- Vertical penetration reveals columnar organization of orientation selectivity

Pinwheel Arrangement of Orientation Columns Revealed by Optical Imaging of Intrinsic Signals



Orientation Columns



Schematic of Ocular Dominance Columns

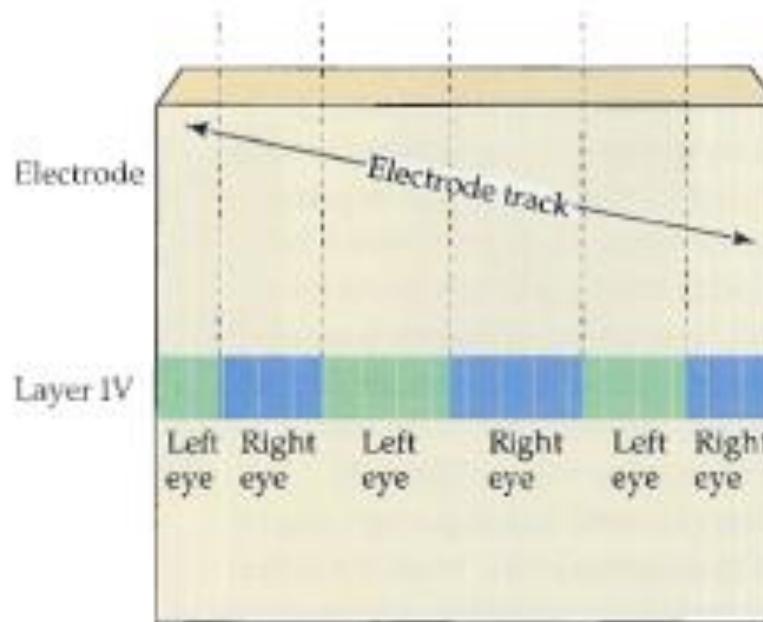


Fig. 11.13, Purves et al., Neuroscience, 3rd edition

- Alternating columns of cells showing preferential responses to right eye or left eye input
 - Monocular cells in layer 4
 - Binocular cells are found in other layers

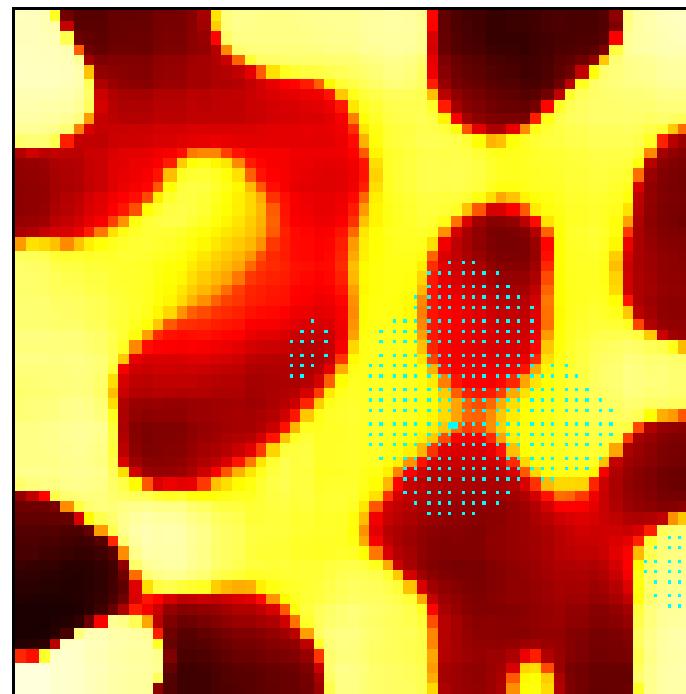
Ocular Dominance Columns in Primary Visual Cortex Revealed by Trans-Synaptic Labeling

- Injection of ^3H amino acid tracer into one eye
- Tracer is transferred trans-synaptically from retina to LGN to cortex
- Autoradiography of flattened cortical sheet reveals interdigitating regions of left eye vs. right eye inputs

Autoradiogram of V1



Ocular Dominance Columns in Simulations



Orientation sensitivity AND Ocular Dominance

Properties of OrientationMaps:

- The maps of O.S. and O.D. are highly repetitive
- Orientation changes continuously as a function of cortical location except at isolated points.
- Orientation changes by 180 deg around singularities
- Both types of singularities appear in equal numbers
- There exist line-like regions (fractures), across which orientation preferences change rapidly with distance.
- (Obermeyer, Blasdel & Schulten 1992)

Orientation sensitivity AND Ocular Dominance

- Properties of Ocular Dominance Maps
 - Ocular dominance changes continuously as a function of cortical location.
 - The ocular dominance pattern is locally organized into parallel strips, which sometimes branch and terminate.
 - **Iso-orientation slabs often cross the borders of ocular dominance bands at approximately right angles.**
 - **The singularities tend to align with the center of the ocular dominance bands.**

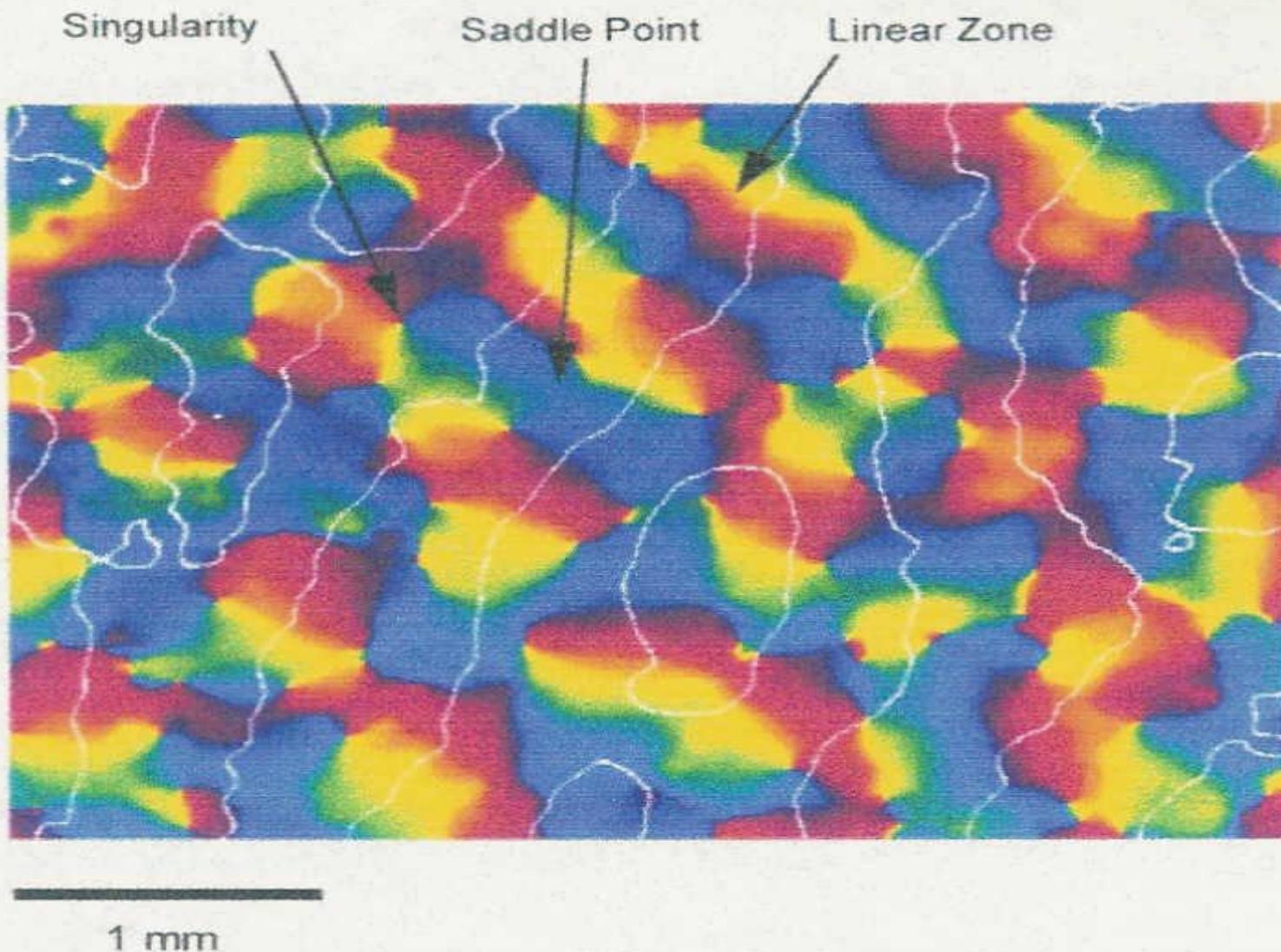


Figure 3. Composite figure showing the arrangement of orientation domains (a single colour represents a unique range of orientation preferences) and their relationship with ocular dominance column boundaries (white lines). The images were obtained by optical recording in macaque monkey striate cortex. Note that the iso-orientation domains tend to intersect ocular dominance column borders at right angles. (Figure supplied by K Obermayer, from data presented in Blasdel (1992b).)

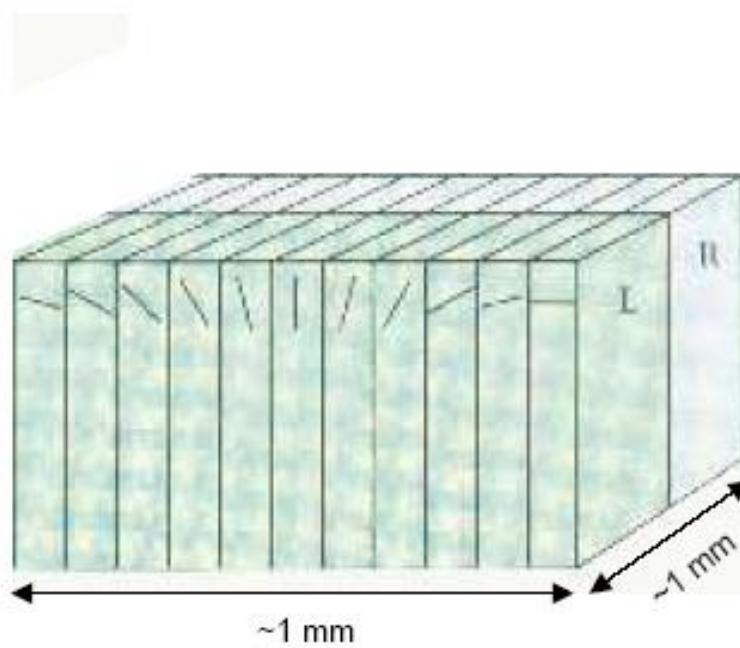
Blobs

- Peg-shaped regions of cells
- In layers 2 and 3 of V1
- These cells respond to color, not orientation
- About 0.2 mm dia

Hypercolumn

- Smallest unit in V1 necessary to analyze all aspects of a region of visual field.
- Area = 1 sqmm
- Complete set of orientation columns (180°)
- Inputs from both eyes.
- Several blobs

Schematic of a Cortical Hypercolumn: A Unit of Information

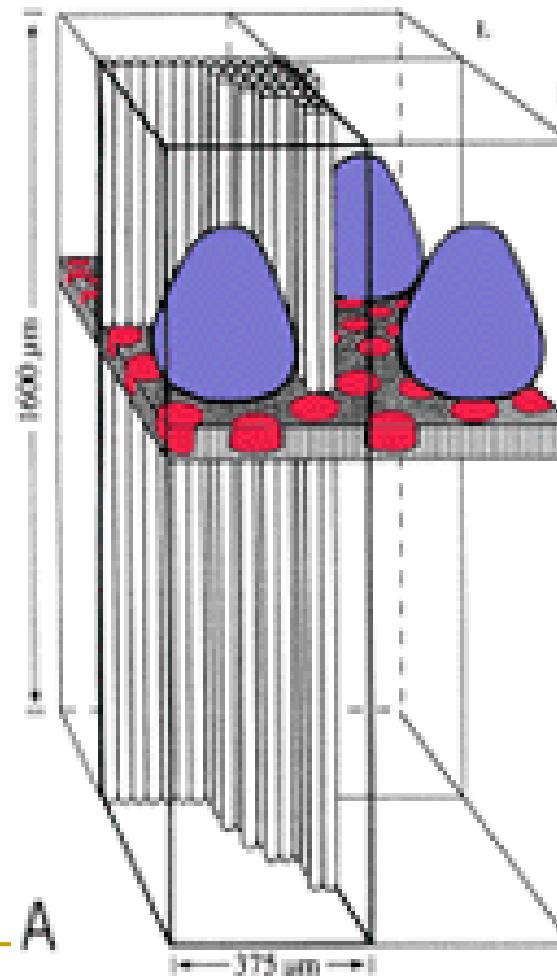


<http://www-psych.stanford.edu/~lera/psych115s/notes/lecture3/figures.html>

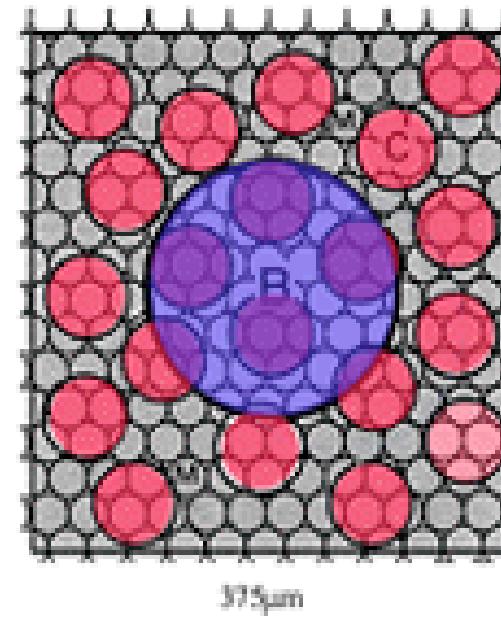
Each column
= 30 – 100 μ m

~1 mm x ~1 mm
180° orientation
one L + R pair

Hypercolumns



IV_A



400 μm

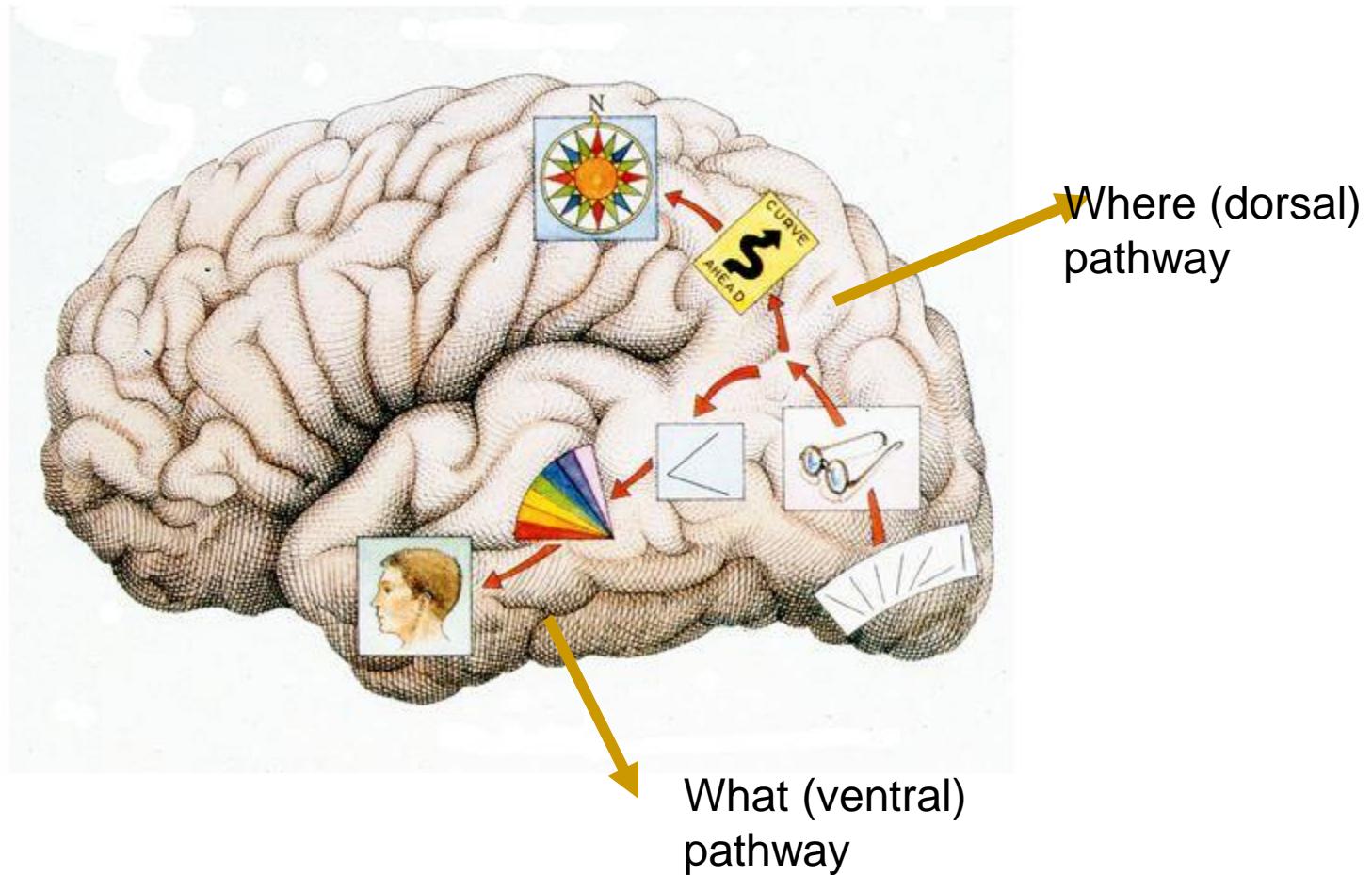
375 μm

B

Horizontal connections among hypercolumns

- Axon collaterals of pyramidal cells in layers 3 and 5 run long distances parallel with layers
- They give rise to clusters of axon terminals at regular intervals that approximate the width of hypercolumn

What and where pathways



Other visual areas

- One in area 17 – V1 (striate cortex)
- Two in area 18 – V2, V3
- Three in area 19 – V3a, V4, V5 (Middle Temporal area)
- Parietal cortex – V5a (Medial Superior Temporal area)

Functions of visual areas

- V1 – primary visual analysis
- V2 – more visual analysis
- V3 – dynamic form
- V4 – color and form
- V5 - motion

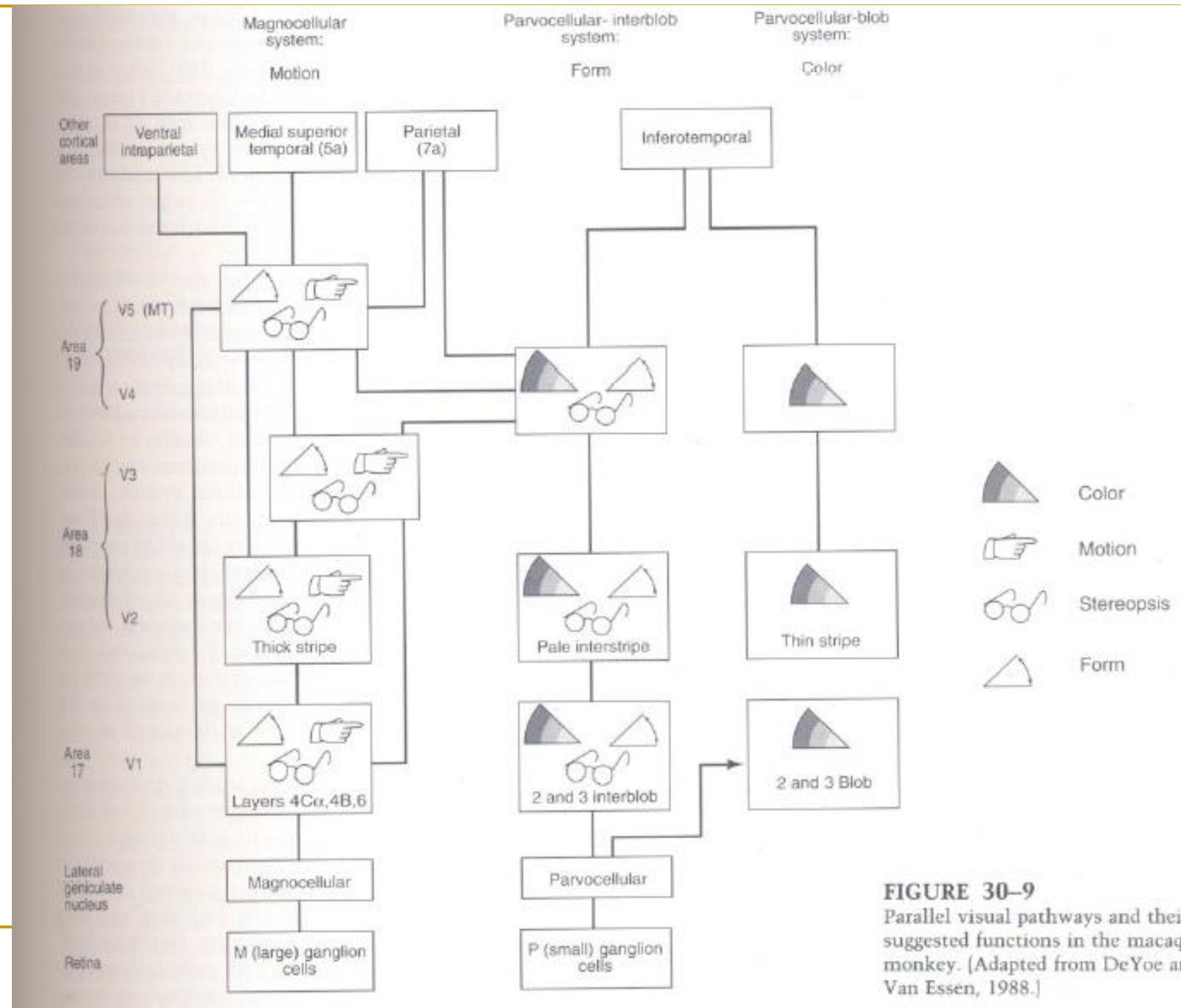


FIGURE 30–9
Parallel visual pathways and their suggested functions in the macaque monkey. [Adapted from DeYoe and Van Essen, 1988.]

Dorsal (“Where”) Pathway

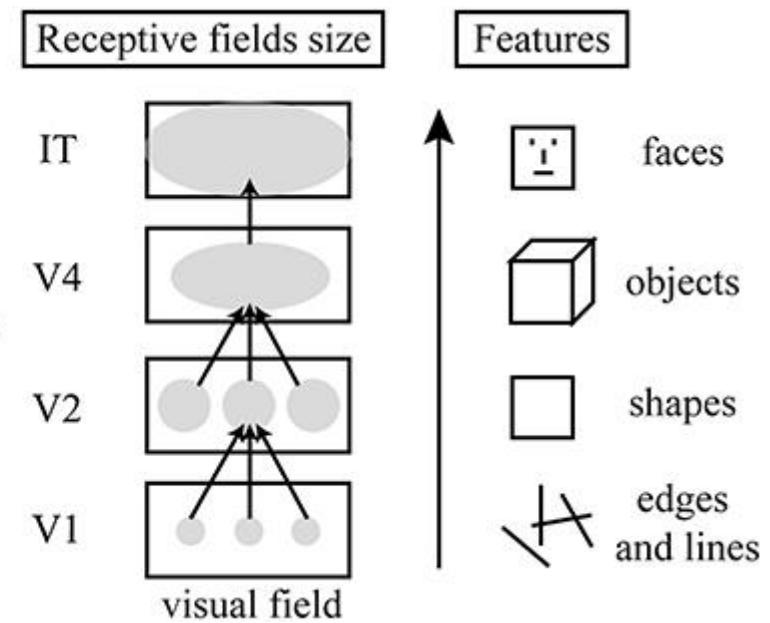
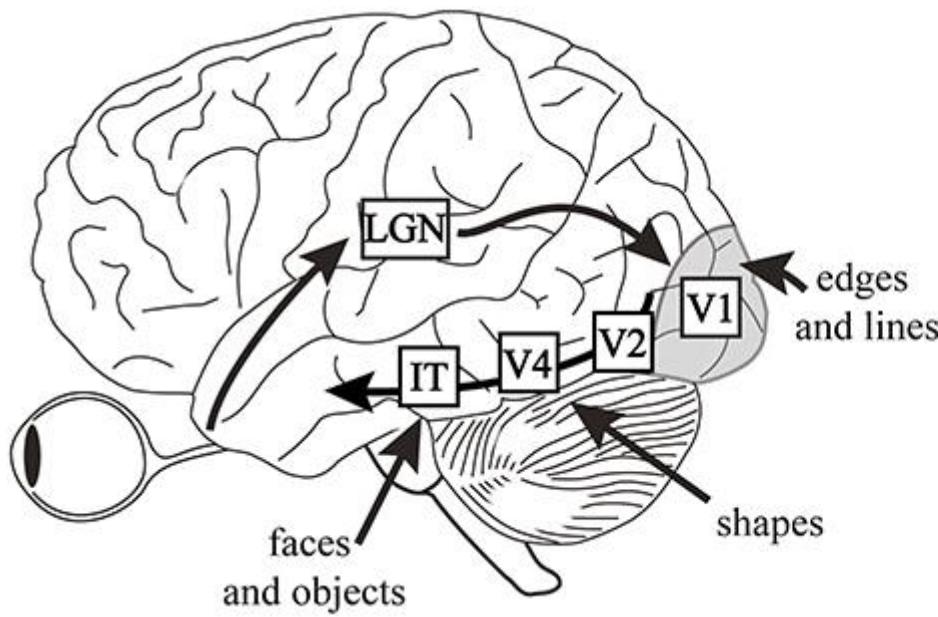
- Also called “how” pathway – connects visual input to motor output
- Starts with visual function in occipital lobe and ends in spatial awareness in posterior parietal cortex (PPC)
- PPC is involved in: "the perception and interpretation of spatial relationships, accurate body image"

Damage to Where Pathway

- **Simultanagnosia:** patients see one object at a time and cannot see several objects as parts of a whole
- **Optic Ataxia:** inability to use visuo-spatial information to guide arm movements
- **Hemineglect:** patient is unaware of the left half of the visual world
- **Akinetopsia:** inability to perceive motion
- **Apraxia:** inability to produce complex and volitional movements

What (ventral) pathway

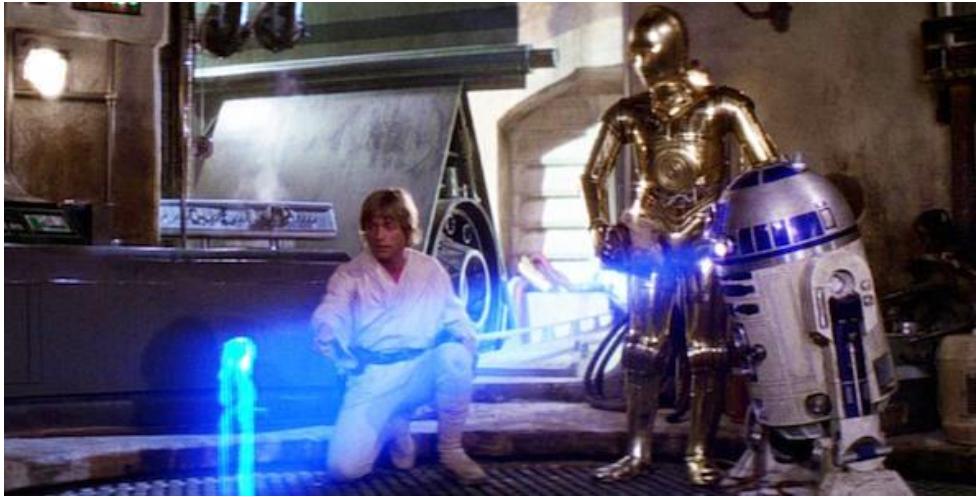
- Object recognition and form representation
- Recognition of complex visual objects.
Particular emphasis on faces.
- Has strong connections to medial temporal lobe (which has hippocampus, which in turn is responsible for declarative memory)
- Damage to this pathway can cause inability to recognize faces - prosopagnosia



<https://neurdiness.wordpress.com/2018/05/17/deep-convolutional-neural-networks-as-models-of-the-visual-system-qa/>

Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future

<https://arxiv.org/abs/2001.07092>



Hopfield networks and Holograms, Magnets and Memories

Hopfield network

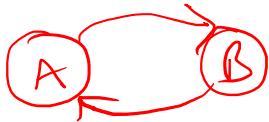
- Hebbian Learning having Associative memory with characteristics
 - Binary Model
 - Update Equations
 - Convergence & Error function
 - Hebbian Learning / Outer product rule

Concepts:

- Attractor as a memory
- Emergent behavior – order in randomness

- Applications:
 - Associative memory/Content Addressable Memory (CAM)
 - Capacity results
 - Role of p/N ratio
 - Spurious memories
- Difficulties:
 - Low capacity
 - Spurious minima

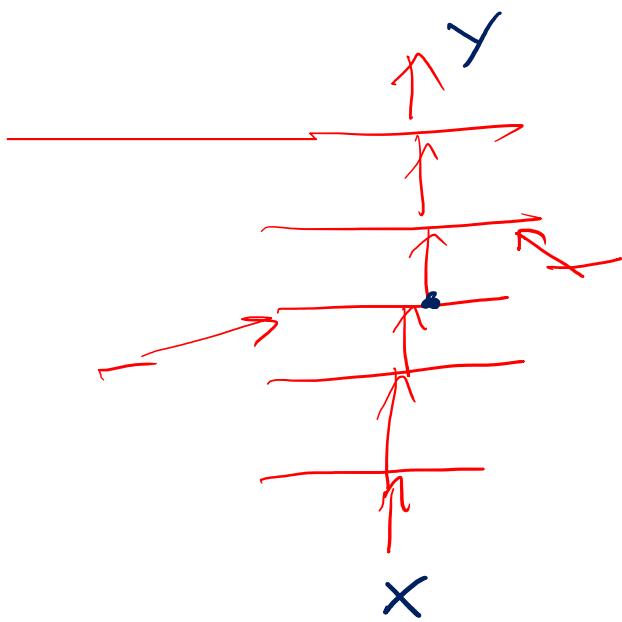
- Extensions:
 - Continuous model
 - Hardware realization
 - Cohen-Grossberg model
 - Bidirectional Associative Memory



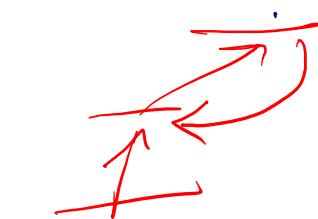
Introduction

- Reentrant connections are more common in Brain than cascades of neural layers connected in feedforward fashion. eg: A cortical area can project in a series of subcortical nuclei, ultimately receives feedback from a nucleus at the end of a long chain, forming a loop.
- Hopfield network is a simple neural network model that has feedback connections. Its significance lies in the fact that it was able to bring together ideas from neurobiology and psychology and present a model of human memory, known as an associative memory.
- The concept of an associate memory can be best explained by contrasting it with computer memory, which is known as indexed memory.

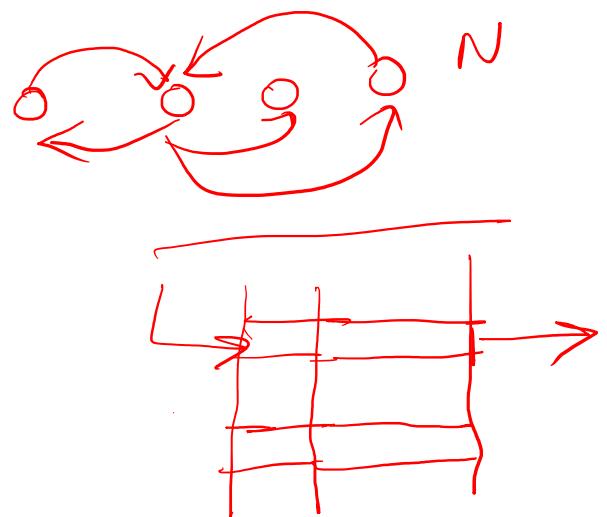
Feedforward Neural Network



Recurrent Neural Network



Hopfield nn



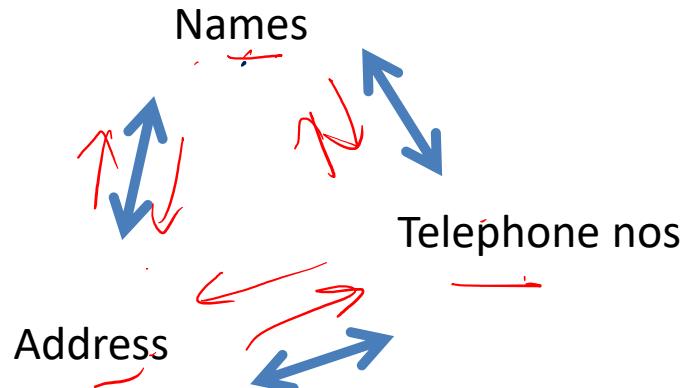
- Indexed memory consists of two columns of data. The first column refers to addresses and the second to contents.
- It is not possible to fetch the address with the help of the content or sub-content₂ with the help of subcontent₁
- in an associative memory, A given item can act as an “address” or “content” depending on the context.
- It is possible to have non-neural network-based based associative memories.
- See Kanerva's memories for example.

Indexed memory:

Name ₁	Name ₂	Name ₃
Address ₁ , Telephone ₁	Address ₂ , Telephone ₂	Address ₃ , Telephone ₃

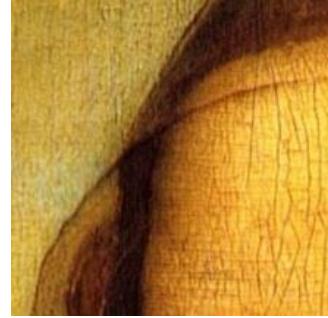
index(t.name)
 name physical
 ↓ ↓

Associative memory:







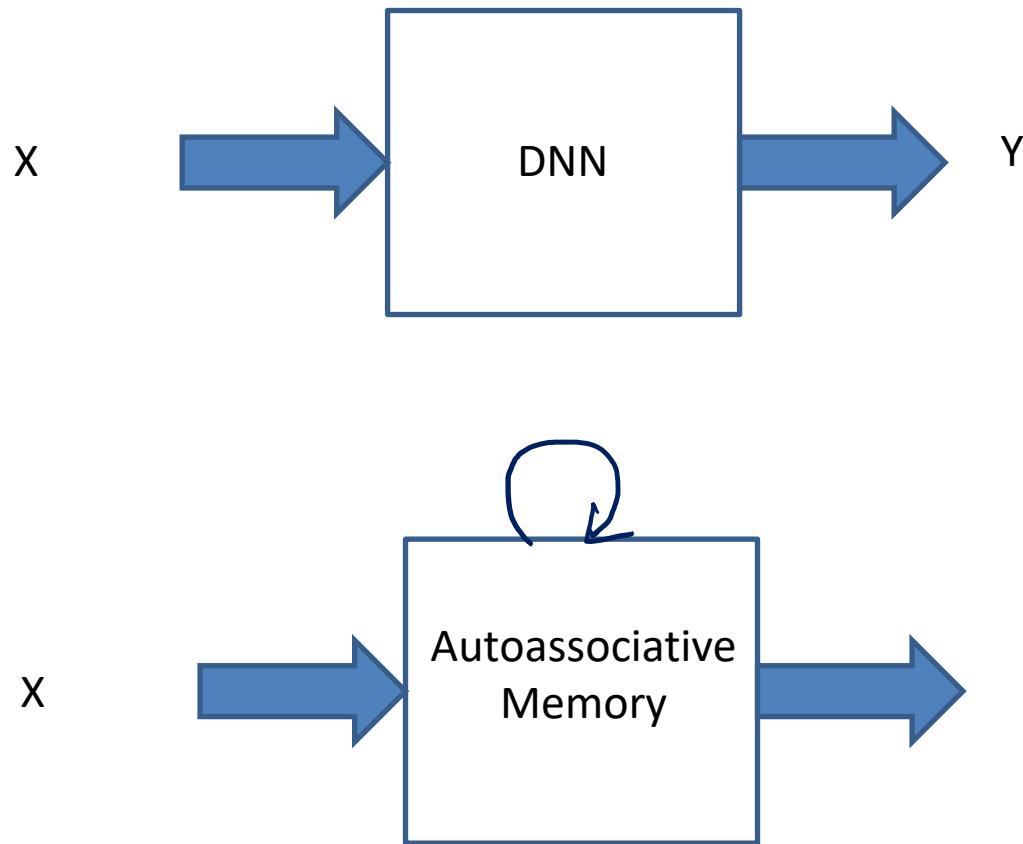


part



whole

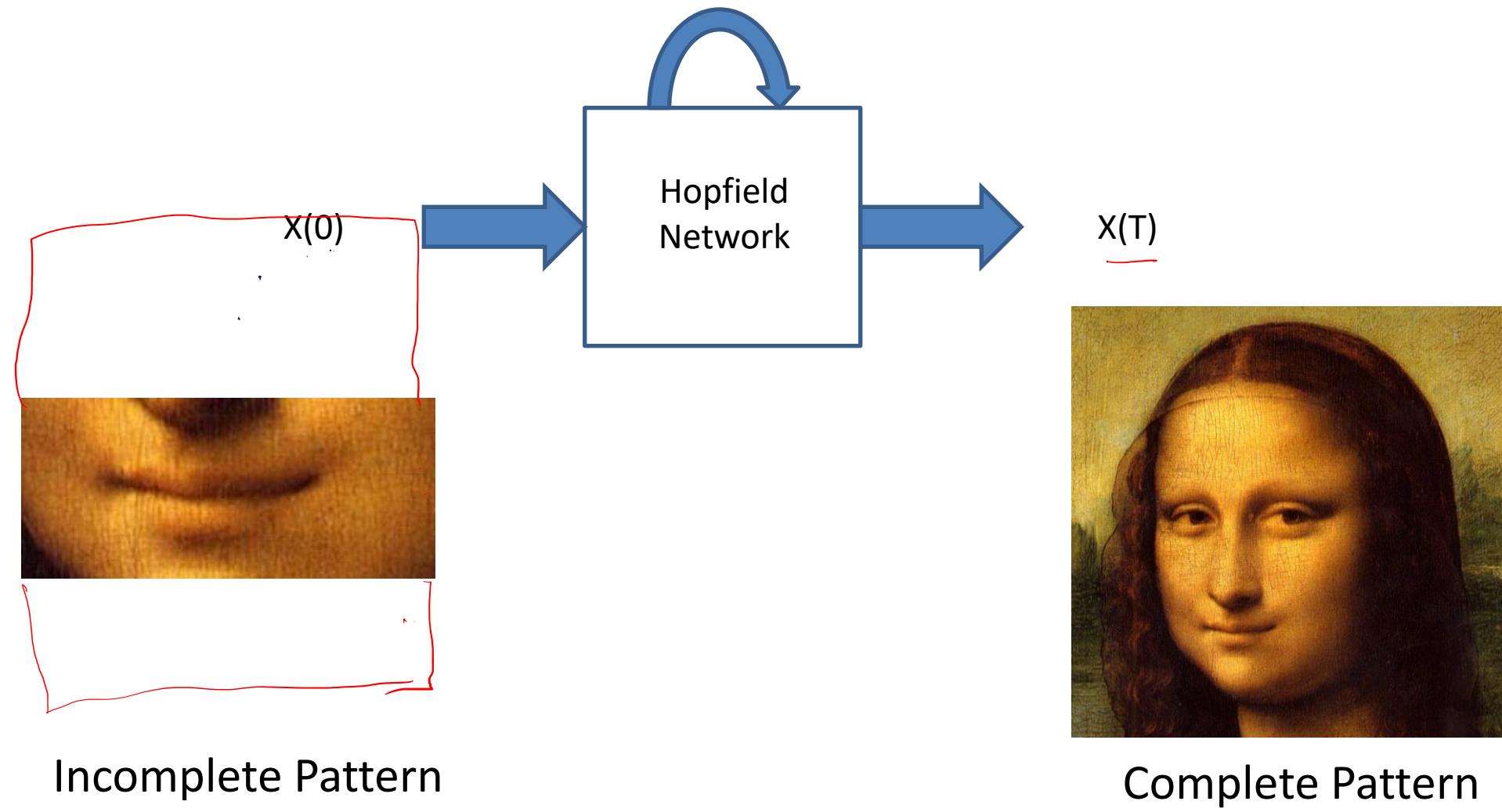
Autoassociative Memory



Also called Content Addressable Memory

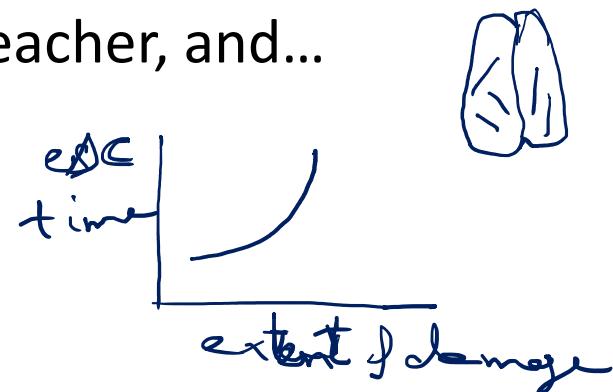
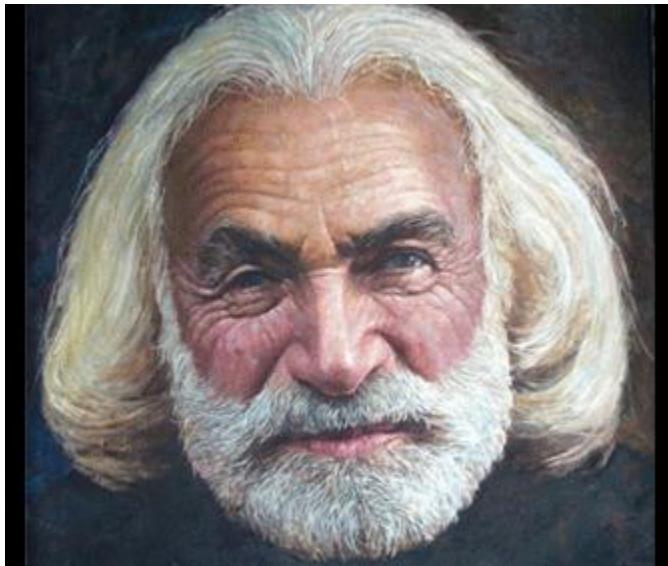
Autoassociative Memory

$$x(0) \rightarrow \underline{x(1)} \rightarrow x(2) \rightarrow \dots \rightarrow x(T)$$



Memories are holograms

- Karl Pribram one of Lashley's students continued the search for the engrams, a lifelong pursuit of his teacher, and...



...saw the analogy between
Memories and **Holograms**

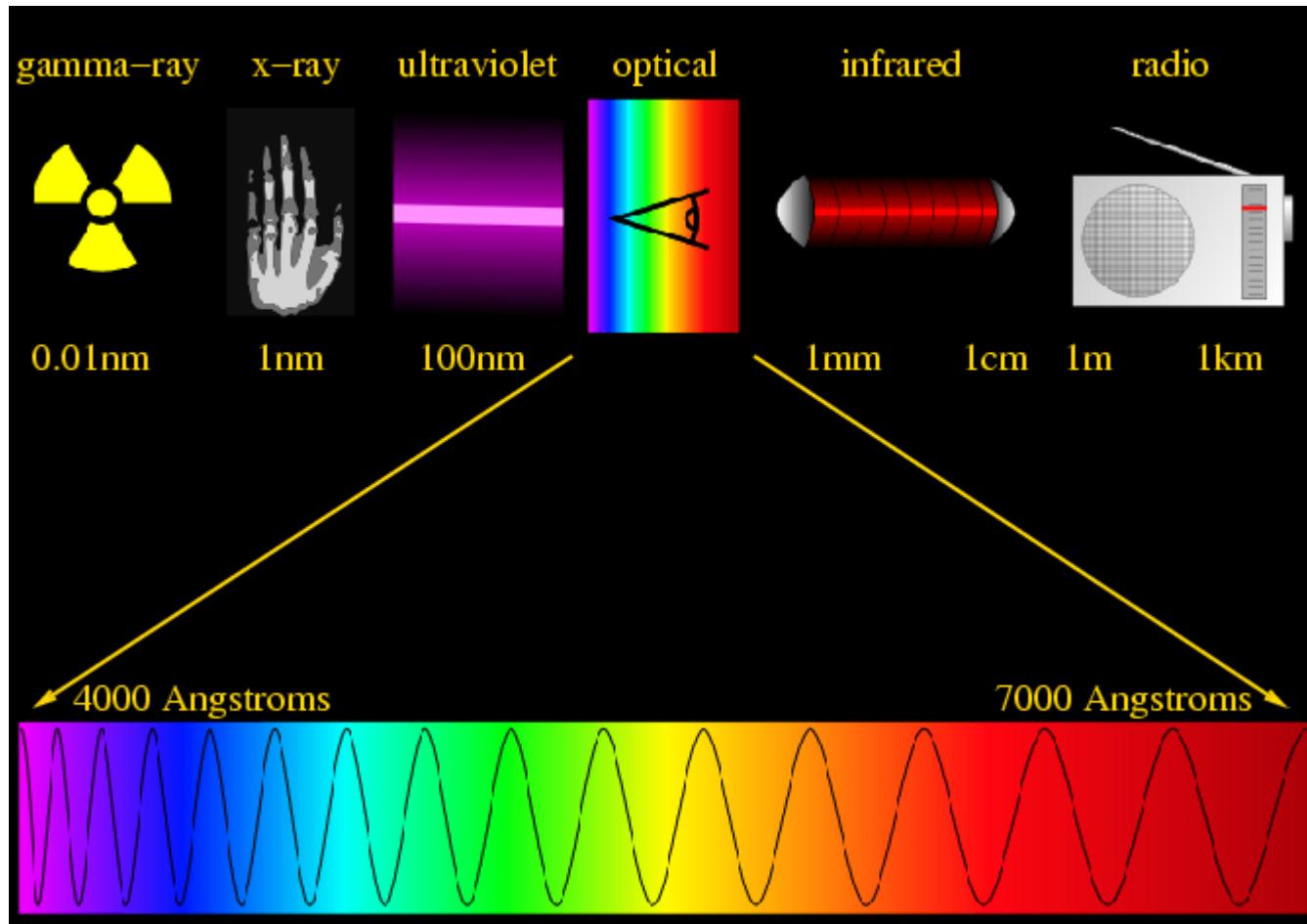
Karl Pribram (1919-2015)

Holograms

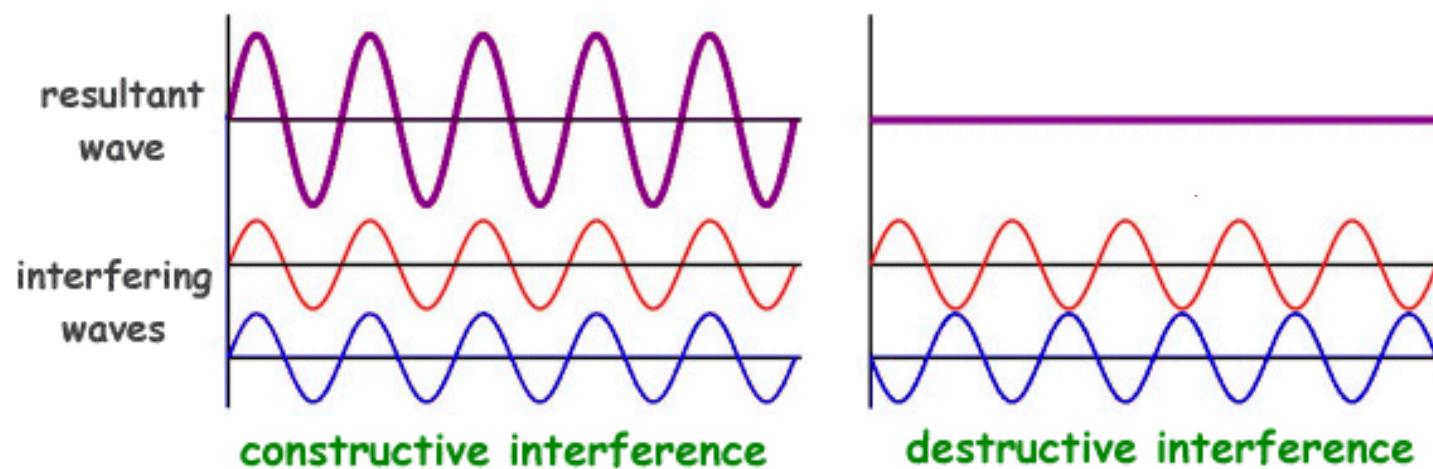
- Holograms are 3D-like displays of objects
 - Unlike images which look the same as view varies...
A hologram varies with varying view
- They are based on the principle of Interference of Light



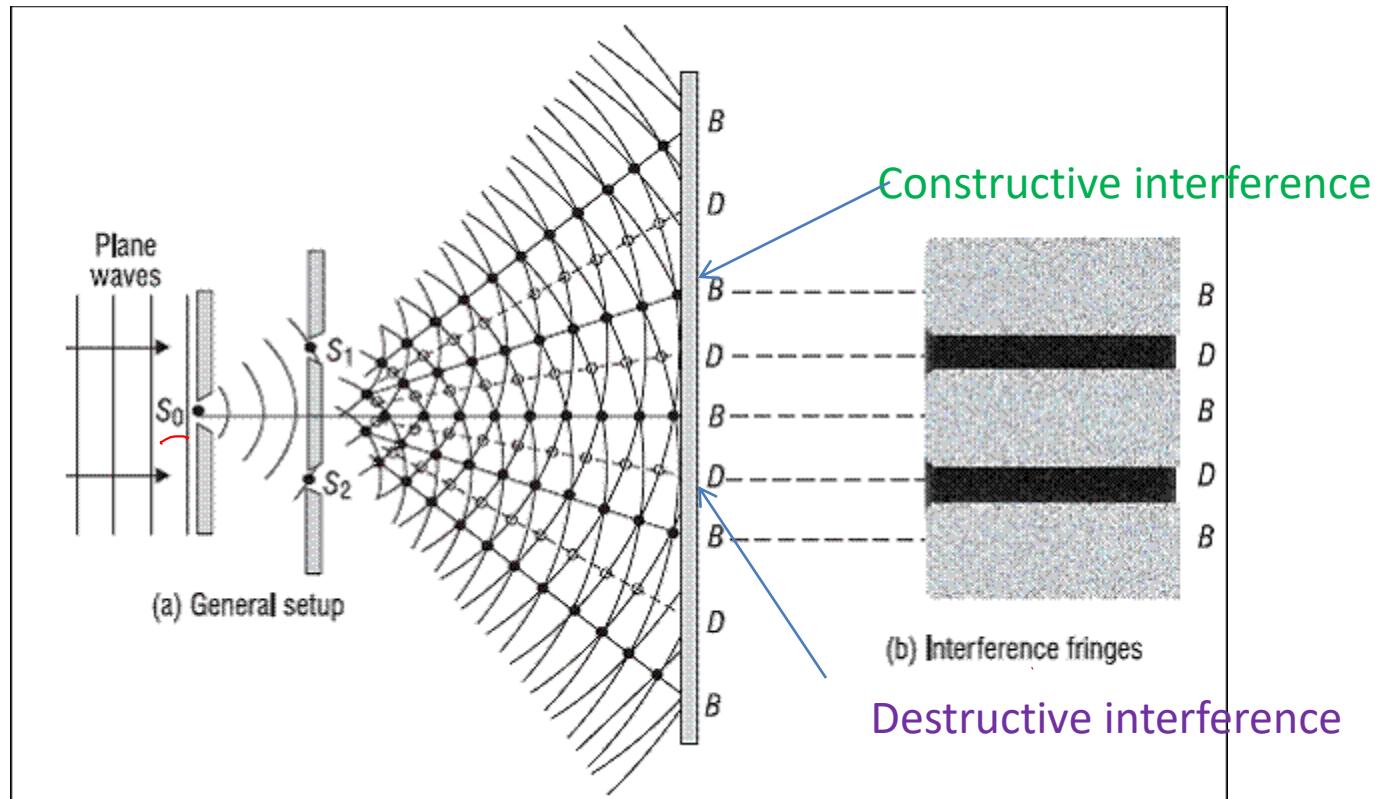
Light is a wave



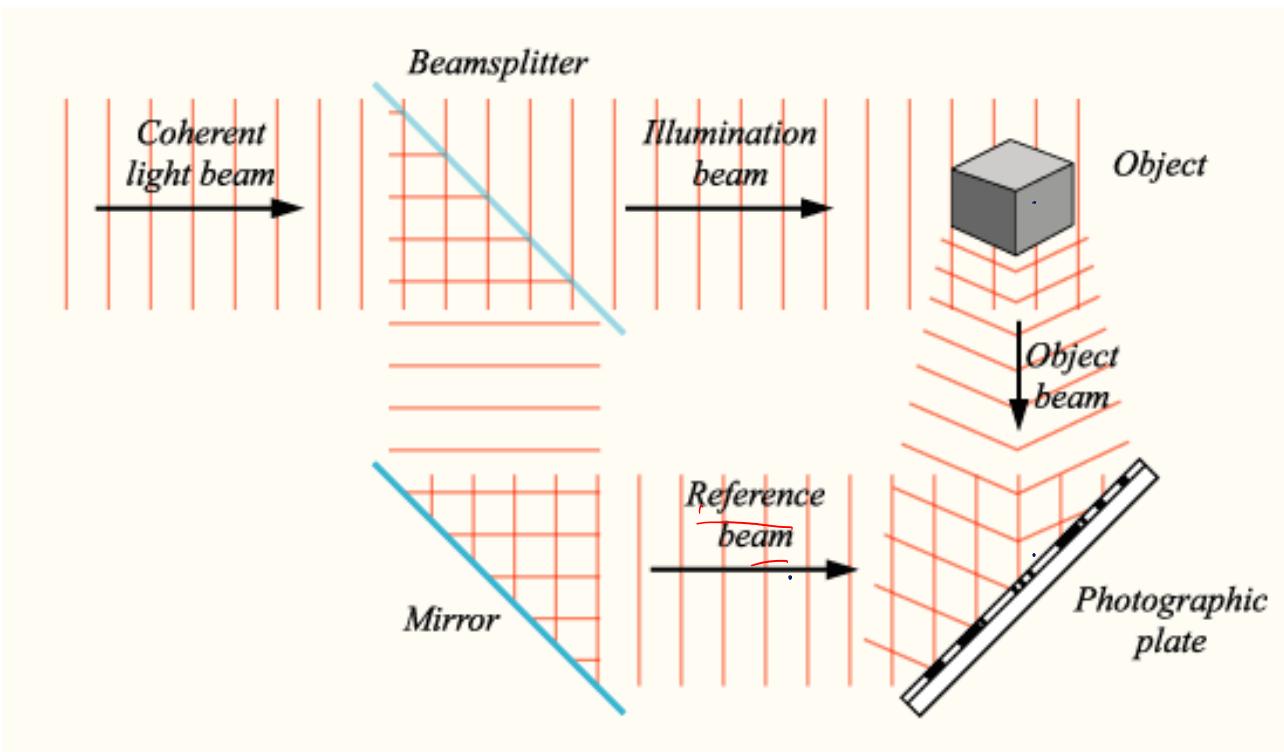
Waves and Interference



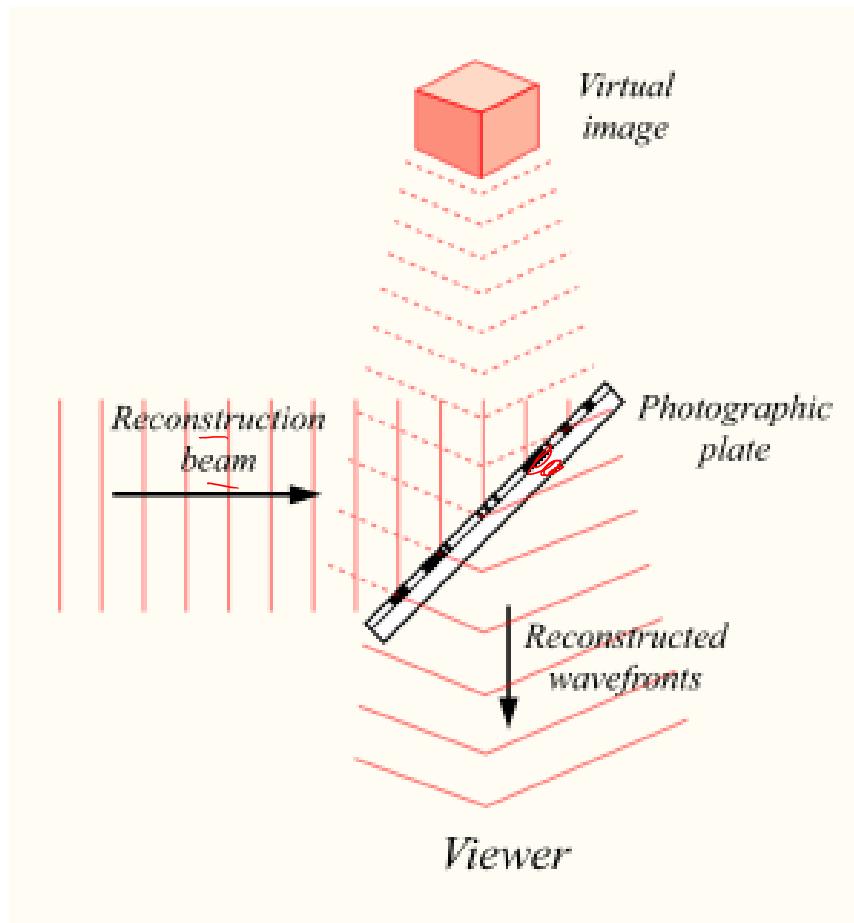
Interference patterns of Light



Hologram Recording



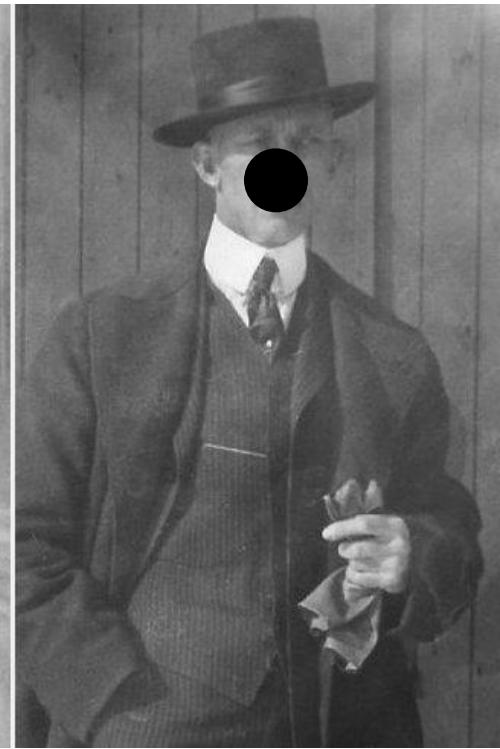
Hologram Reconstruction



-ve

Photograph vs...

+ve

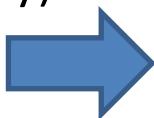


Hologram



Reconstructed from the original Hologram

Reconstructed from a partly (locally) destroyed Hologram



A synthesis of many ideas

Assoc ~~hom~~ memory.

McCulloch-Pitts neuron

≈ holograms.

≈ magnetic materials.

Hebbian learning ← neurobiology

Discrete Hopfield Network

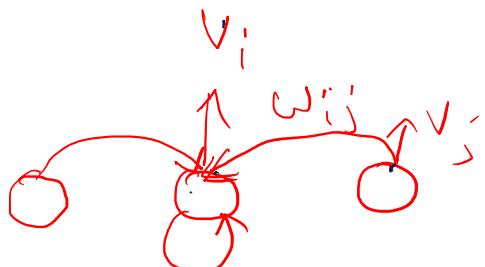
- Hopfield has proposed two basic models of associative memories (Hopfield 1982, 1984).
- In the discrete Hopfield network, each neuron has a binary state $V_i \in \{1, -1\}$ ✓
- The state of the network with N neurons is represented by the vector,

$$V = [V_1, \dots, \underbrace{V_i, \dots, V_N}_{}]^T$$

- The network is fully-connected, i.e., each neuron connected to all others.
- The weight from j'th neuron to i'th neuron is given by, and weight matrix is given as

$$W = \{w_{ij}\}$$

- Since the network has loops, computations are dynamic and the network state evolves through time, which is a discrete variable.

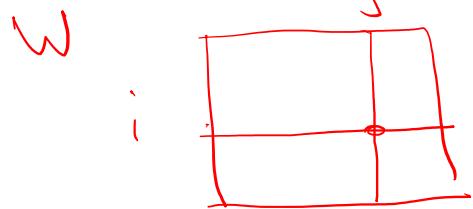


N

$$v_i \in \{+1, -1\}$$

$$v_i = \sigma \left(\sum_{j=1}^N w_{ij} v_j \right)$$

N^2 connection



$$\sigma = \begin{cases} 1 & x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$\sigma(x) = +1, \quad x \geq 0 \\ = -1, \quad \text{otherwise}$$

~~v_i~~

$$v_i(t+1) = \sigma \left(\sum_{j=1}^N w_{ij} v_j(t) \right)$$

Signum()

Update:- $v(t) \rightarrow v(t+1)$

Update:-

Update

- At any time step, t , each neuron receives inputs from all other neurons, and updates its own state. The next state of i 'th neuron is expressed as a function of current state of the network as follows:
- Update Equations:

$$V_i(t+1) = \sigma\left(\sum_{j=1}^N w_{ij} V_j(t)\right) \quad \checkmark$$

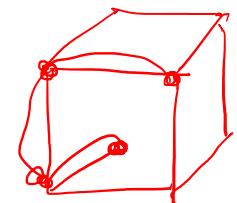


Where $s()$ is the sign function, $\text{sign}()$.

$$V_i = \pm 1$$

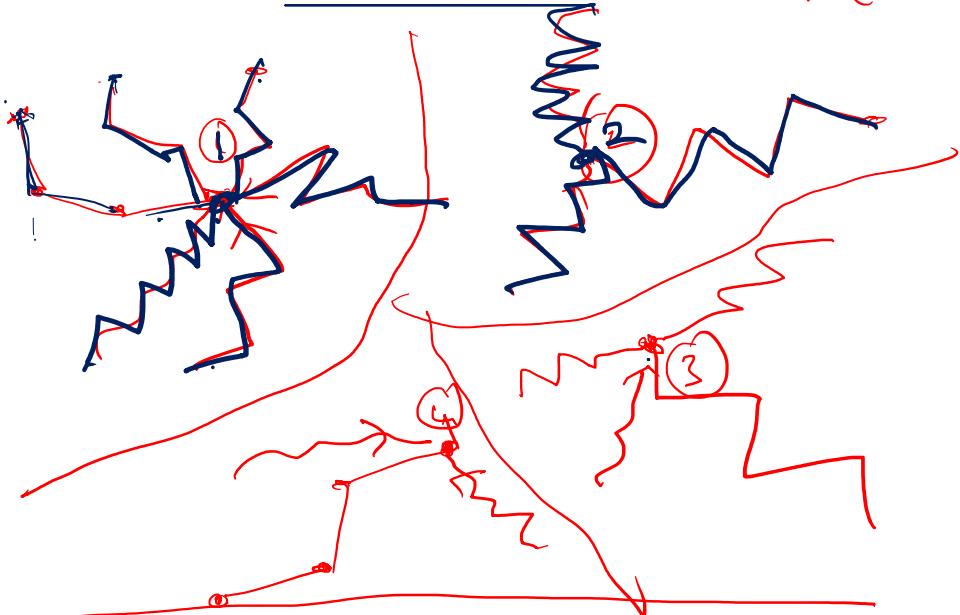
$$N = 3$$

$V(0) \rightarrow V(1) \rightarrow V(2) \rightarrow$

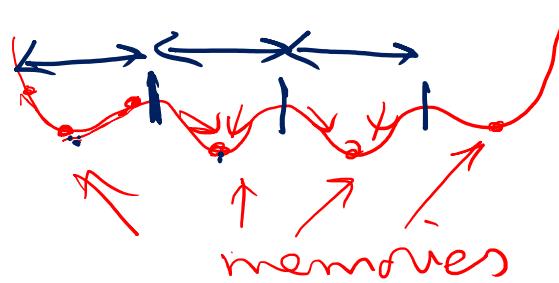


$V(0) \rightarrow V(1) \rightarrow V(2) \dots$

$V(t+1) = V(t)$ for $t > T$

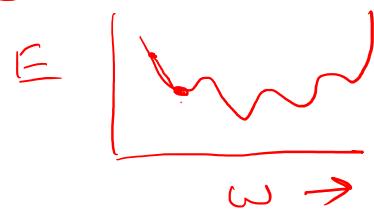


"Energy" function (V)



2^N states

~~NLP/DP~~
~~NP~~ for n.m.c.



Convergence and Energy function

Energy function:

$$E = -1/2 \left(\sum_{i=1}^N \sum_{j=1}^N w_{ij} V_i V_j \right)$$

$$\begin{aligned}V_i &: - \\-1 &\rightarrow 1 \\+1 &\rightarrow -1\end{aligned}$$

E is bounded since V is bounded.

$$w_{ij} = w_{ji} \quad (\text{Symmetry})$$

Assume we are updating V_i , calculate the change in E. Consider only the terms in E

$$\Delta E = -(\Delta V_i) \left(\sum_{j \neq i}^N w_{ij} V_j \right) = (2V_i(t)) \left(\sum_{j \neq i}^N w_{ij} V_j \right)$$

which contains V_i .

$$\begin{aligned}& +2w_{ii}V_i^2 - 2w_{ii}V_i^2 \\& < 0 \quad \text{if } w_{ii} > 0 \\V_i(t+1) &= \sigma \left(\sum_{j=1}^N w_{ij} V_j(t) \right)\end{aligned}$$

If $\delta(V_i) < 0$, ($1 \rightarrow -1$): the sum is also -ve.

- If $\delta(V_i) > 0$, ($-1 \rightarrow 1$): the sum is also +ve.
- Therefore, $\delta(E)$ is always -ve or 0.
- But since E is bounded the updating process for 'E' should eventually converge.

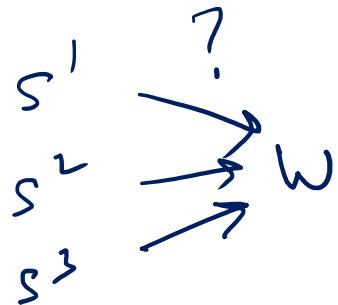
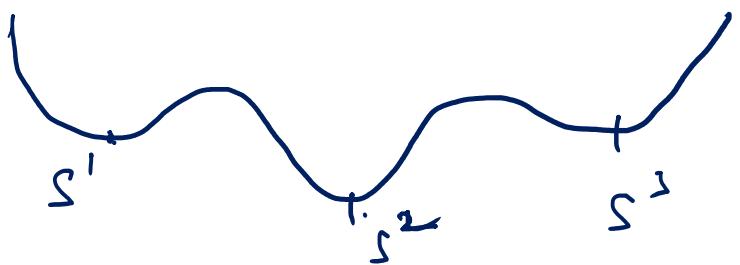
$$V_i(t) \quad V_i(t+1) \quad \Delta V_i = V_i(t+1) - V_i(t) = -2V_i(t)$$

$$1 \quad -1 \quad -2 = -2V_i(t)$$

$$-1 \quad 1 \quad +2 = -2V_i(t)$$

$$E(v; \omega)$$

E



Green #1

$$E = \|v - s\|^2$$

$$= (v - s)^T (v - s)$$

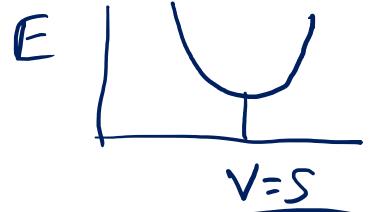
$$= \frac{v^T v}{n} + \frac{s^T s}{n} - \frac{s^T v - v^T s}{2} - \frac{2(s^T v)}{2}$$

Store, $s \rightarrow \omega$

$$s_i = \pm 1$$

$$\underline{E(v; \omega)}$$

must have a min
at $v = s$.
(quadratic)



Green #2

$$E = -(s \cdot v)^2$$

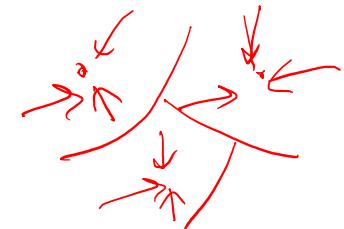
$$= -(\bar{s}^T v)(s^T v)$$

$$= -(v^T s)(\bar{s}^T v) = -\underbrace{v^T (s s^T)}_{\text{constant}} v$$

Hebb's rule

Suppose we want to store $S = \{S_i\}$ in a network. For that E should be minimum at S . So we chose, E is minimum at $\underline{S = V}$.

$$E = -1/(2N) \left(\sum_i S_i V_i \right)^2$$



Putting E in the standard form given above:

$$E = -1/(2N) \left(\sum_i S_i V_i \right) \left(\sum_j S_j V_j \right) = -(1/2) \sum_{ij} \left(\frac{1}{N} S_i S_j \right) V_i V_j$$

So the weight matrix required to store a pattern S is:

$$\cancel{W_{ij}} = (1/N)(S_i S_j).$$



To store multiple patterns, S^p the rule is:

$$w_{ij} = (1/N) \sum_{p=1}^P S_i^p S_j^p$$

$$P = \# \text{ patterns}$$

Hebb's rule

Single Element form:

$$w_{ij} = \frac{1}{N} S_i S_j$$

Single pattern storage

$$w_{ij} = \frac{1}{N} \sum_{p=1}^P S_i^p S_j^p$$

Multiple pattern (P) storage

Matrix form:

$$W = \frac{1}{N} S S^T$$

Single pattern storage

$$W = \frac{1}{N} \sum_{p=1}^P S^p S^{pT}$$

Multiple pattern storage

MLP vs Hopfield network

	MLP	Hopfield
State Update	Instantaneous	Iterative
Weight Update	Iterative	Instantaneous



State update

$$V_i(t+1) = \sigma\left(\sum_{j=1}^N w_{ij} V_j(t)\right) \quad (\text{Online update})$$

$$V(t+1) = \sigma(W V(t)) \quad (\text{Batch update})$$

When only a single pattern, S , is stored,
 S is a stationary state.

$$W = \frac{1}{N} SS^T$$



$$V(t+1) = \sigma(W V(t))$$

$$= \sigma\left(\frac{1}{N} SS^T V(t)\right)$$

Consider, $V(0) = S$,

$$= \sigma\left(\frac{1}{N} SS^T S\right) = \sigma\left(\frac{1}{N} SN\right) = \sigma(S) = S$$

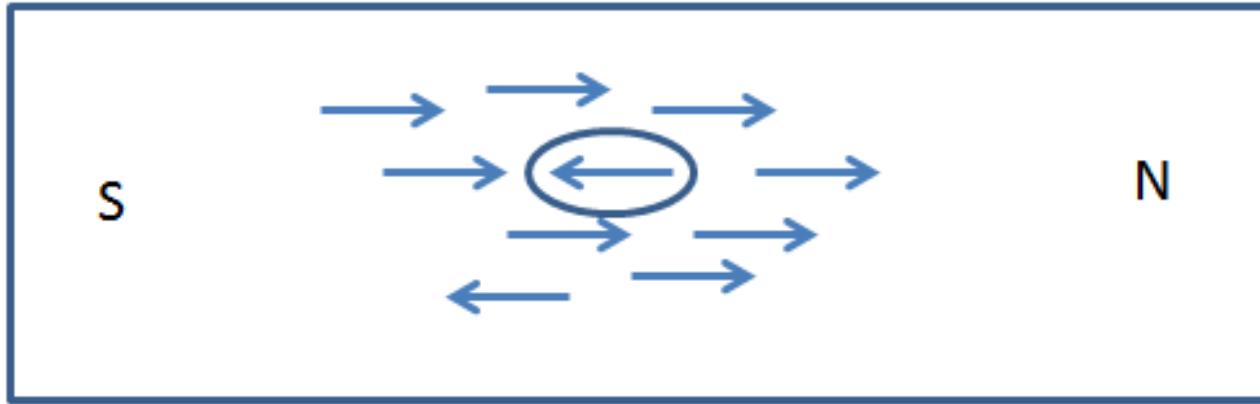
Therefore, $V = S$ is stationary.

Hopfield Networks and Spin Glass Systems

- A magnetic material consists of a set of atomic magnets arranged on a crystal lattice.
- Each atomic magnetic has a spin associated with it. Spin = +/- 1
- Spin of an atomic magnet is influenced by the magnetic field, h , at its location.
- Field = external field (h_{ext}) + internal field (produced by other spins)

Analogy with Magnetic Materials

Ising
Supreme



Spin → Neuron's Binary State

The spin of a given particle is influenced by the spins of the neighboring particles.

“Energy function” → Hamiltonian

Minimum Energy States → memories

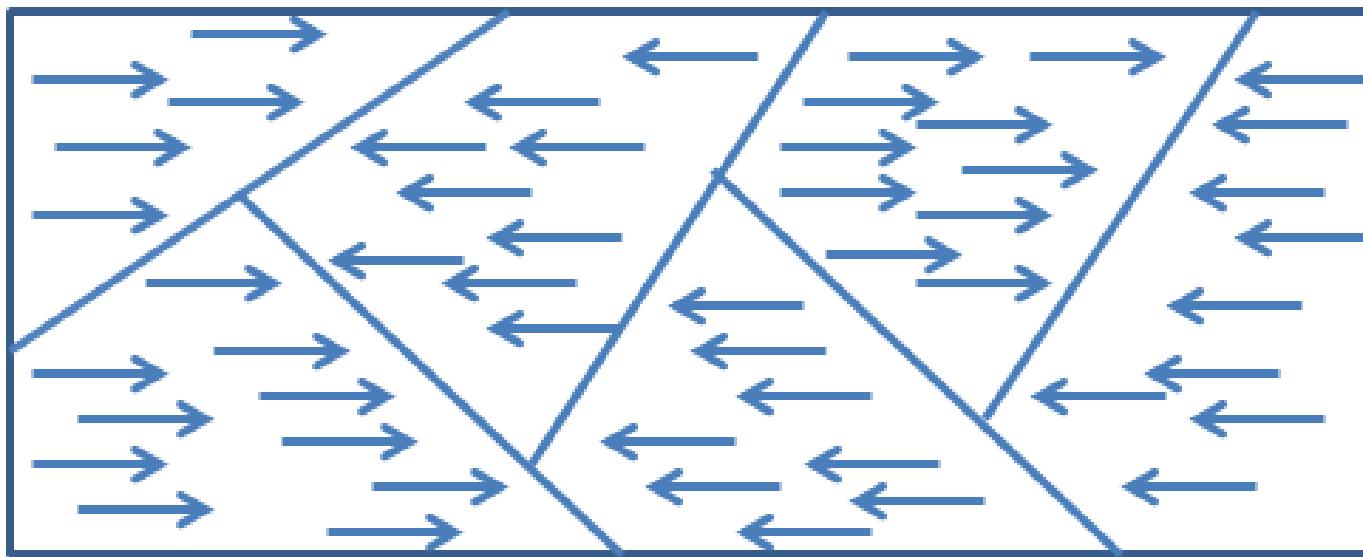
$$h_i = \sum_j w_{ij} S_j + h_{ext}$$

- W_{ij} = (**exchange interaction strength**) measure of strength of interaction of influence of spin S_j on the field at S_i .
- Interactions are symmetric. $W_{ij} = W_{ji}$.

Potential energy corresponding to the interaction is:

$$H = -\frac{1}{2} \sum_j w_{ij} S_i S_j - h_{ext} \sum_i S_i$$

Magnetic Memory



Distinct stable domains:
all spins in a given domain are pointed in the same direction

Hebbian Learning

- “When an axon of a cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.” (‘Organization of Behavior’ by Donald Hebb)
- When two neurons are simultaneously active, the connection between them must be strengthened; when one of them is active, while the other is inactive, the connection strength must be weakened.
- “Neurons that fire together, wire together.”

Pavlovian Conditioning or Classical Conditioning



Unconditioned Response
(Salivation)



Unconditioned Stimulus
(Food)



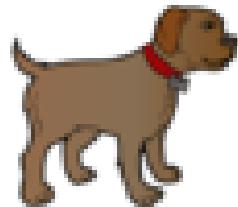
Unconditioned Response
(Salivation)



Neutral Stimulus
(Bell Ringing)



Unconditioned Stimulus
(Food)



No Response



Neutral Stimulus
(Bell Ringing)



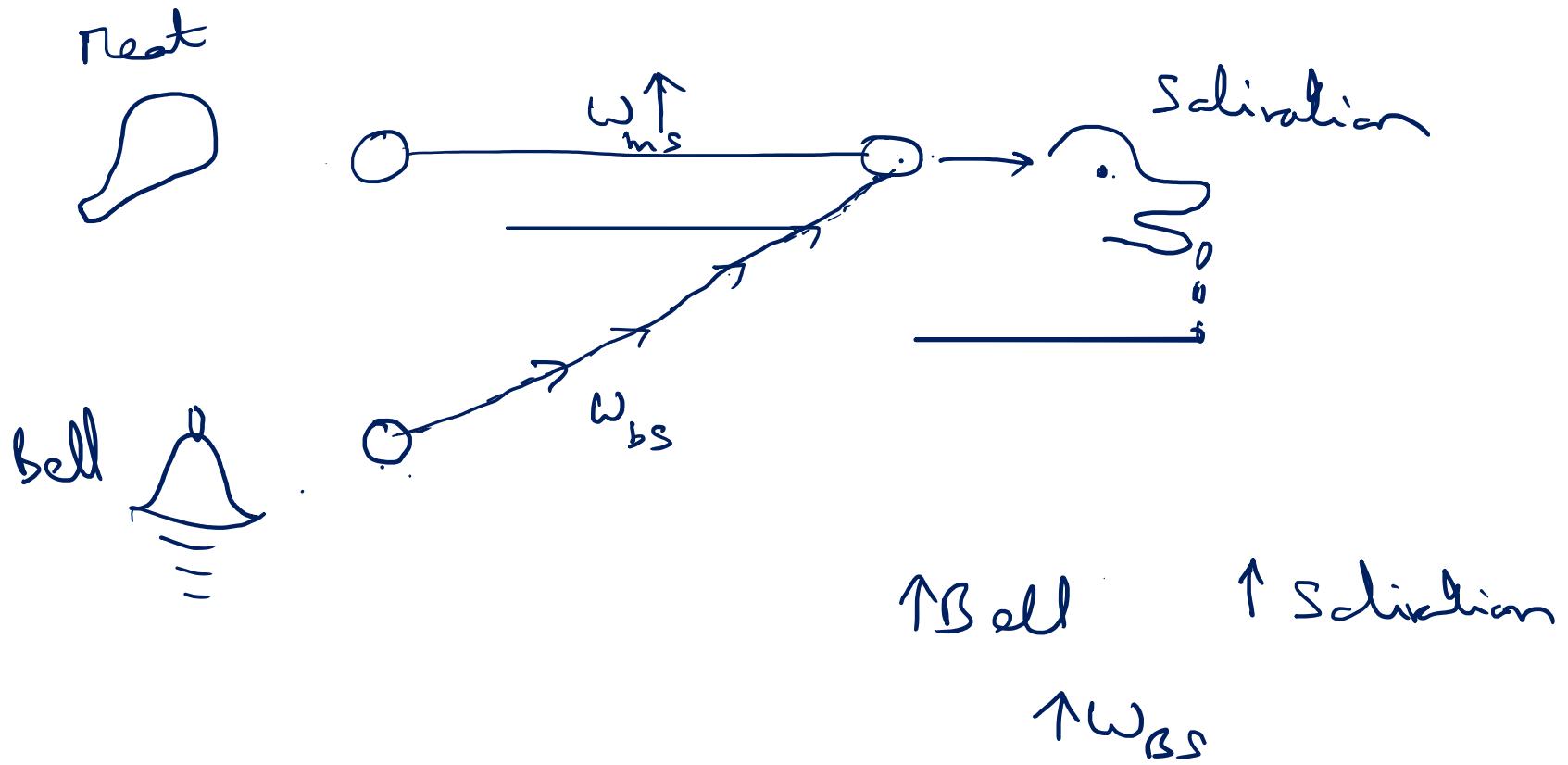
Conditioned Response
(Salivation)



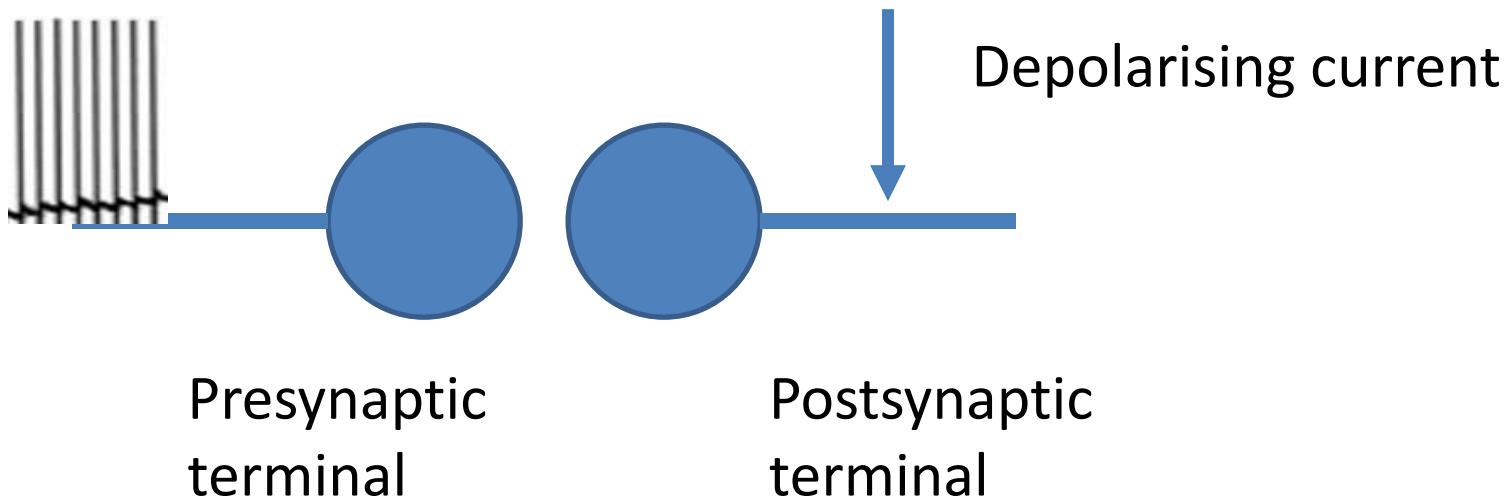
Conditioned Stimulus
(Bell Ringing)

https://en.wikipedia.org/wiki/Classical_conditioning

Pavlov was awarded the Nobel Prize in 1904 for Physiology and Medicine

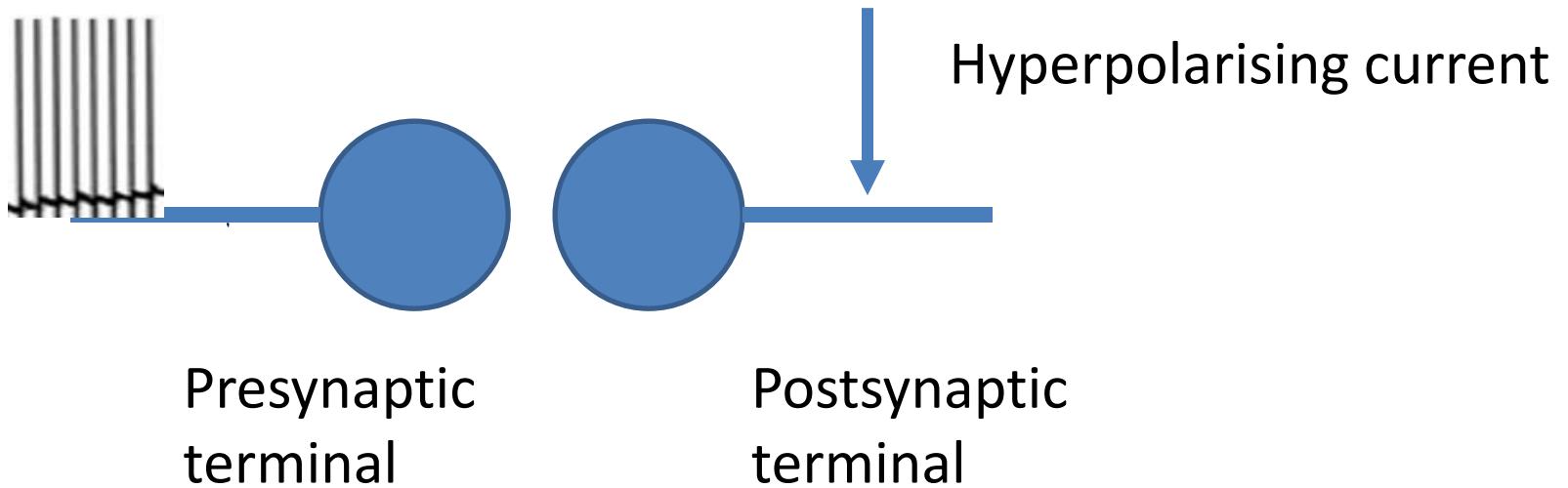


Long-term Potentiation (LTP)



Presynaptic	Postsynaptic	Outcome
High Frequency APs	Depolarization	Synaptic strengthening (LTP)

Long-term Depression (LTD)



Presynaptic	Postsynaptic	Outcome
High Frequency APs	Hyperpolarization	Synaptic weakening (LTD)

Storing Multiple Patterns

$$w_{ij} = (1/N) \sum_{p=1}^P S_i^p S_j^p$$

The problem of Spurious states

- A problem arises when we try to store multiple patterns using Hebb's rule.
- In addition to the stored states, certain other states known as spurious states, also end up becoming stable, interfering with the network operation.
- Therefore, when the network is prompted with the corrupt version of a stored pattern, the network might retrieve a pattern which is not one of the stored patterns, thereby diminishing the network's retrieval performance



Types of Spurious states

- **Mirror states:** It can be shown that if S is a stable state, $-S$ is also a stable state.

Proof: If S is stable, we know that

$$S = g(WS)$$

If g is the sign() function, we have $-x = g(-x)$, if $x = \pm 1$. Therefore, we have,

$$-S = g(-WS).$$

$$P(S_i = +) = 0.5$$

$$N \gg 1$$

Mixture states: $S^{\text{mix}} = \text{sign}(\pm S_1 \pm S_2 \pm S_3)$

S^p are random

Proof: S^{mix} and S^p ($p = 1, 2, 3$) have the same sign on average 3 times out of 4.

Hence, $\sum_i S_i^p S_i^{\text{mix}} = N/2$

Therefore, $h_i^{\text{mix}} = 1/N \sum_{jp} S_i^p S_j^p S_j^{\text{mix}} = S_i^1/2 + S_i^2/2 + S_i^3/2 + \text{cross-terms}$

If the cross-terms can be ignored, the mixture state is stable.

- **Spin glass states:** Stable states that are not correlated with the stored patterns.

- If N is large, from Central Limit Theorem we know that C_i has a Gaussian distribution with the above mean and variance.

$$\begin{aligned}
 P(C_i > 1) &= 1 / (\sqrt{2\pi} \text{ sigma}) \int_1^{\infty} \exp(-x^2 / 2\sigma^2) dx = (1/2)[1 - \operatorname{erf}(1/\sqrt{2\sigma^2})] \\
 &= (1/2)[1 - \operatorname{erf}(\sqrt{N/2P})]
 \end{aligned}$$

P(Ci>1)	P/N
0.001	0.105
0.0036	0.138
0.01	0.185
0.05	0.37
0.1	0.61

$$\frac{P}{N} = 0.138$$

Memory Capacity

Rapid degradation of retrieval performance beyond,

$$P/N = 0.138$$

Hopfield got $P/N = 0.15$ through experiments.

Improving Capacity:

The self-weights must be set to zero for improved retrieval performance.

$$W_{ii} = 0$$

To understand this, consider, $w_{ii} \gg 1$,

$V_i(t+1) \approx g(w_{ii} V_i(t)) = V_i(t)$. That is every state is stable. Therefore, there is a greater chance of obtaining spurious states.

$$\begin{array}{ll} N \rightarrow 2^N \text{ states} & \sum w_{ij} v_j \approx w_{ii} v_i \\ 0.138 \approx \text{patterns} & \sigma(w_{ii} v_i) = v_i \end{array}$$

Numerical Example

$$S = \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}^T$$

$$N = 3$$

$$W = \frac{1}{N} SS^T$$

$$W = \overline{SS^T} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix}$$

$$V(0) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$$

$$V_i(t+1) = \sigma\left(\sum_j W_{ij} V_j(t)\right)$$

$$V_1(t+1) = \overline{\sigma\left(\sum_j W_{1j} V_j(t)\right)}$$

$$\begin{aligned} V_1(1) &= \sigma(1 \times 1 + 1 \times 1 + (-1) \cdot 1) \\ &= \sigma(1) = 1 \end{aligned}$$

$$\sigma(\overline{\lambda x}) \xrightarrow{\lambda \geq 0} \sigma(x)$$

$$V_2(1) = \sigma\left(\sum_{j=1}^3 W_{2j} V_j(0)\right)$$

$$= \sigma(1 \times 1 + 1 \times 1 + (-1) \cdot 1)$$

$$= 1$$

$$\begin{aligned} V_3(1) &= \sigma((-1) \times 1 + (-1) \cdot 1 + 1 \times 1) \\ &= \sigma(-1) = -1 \end{aligned}$$

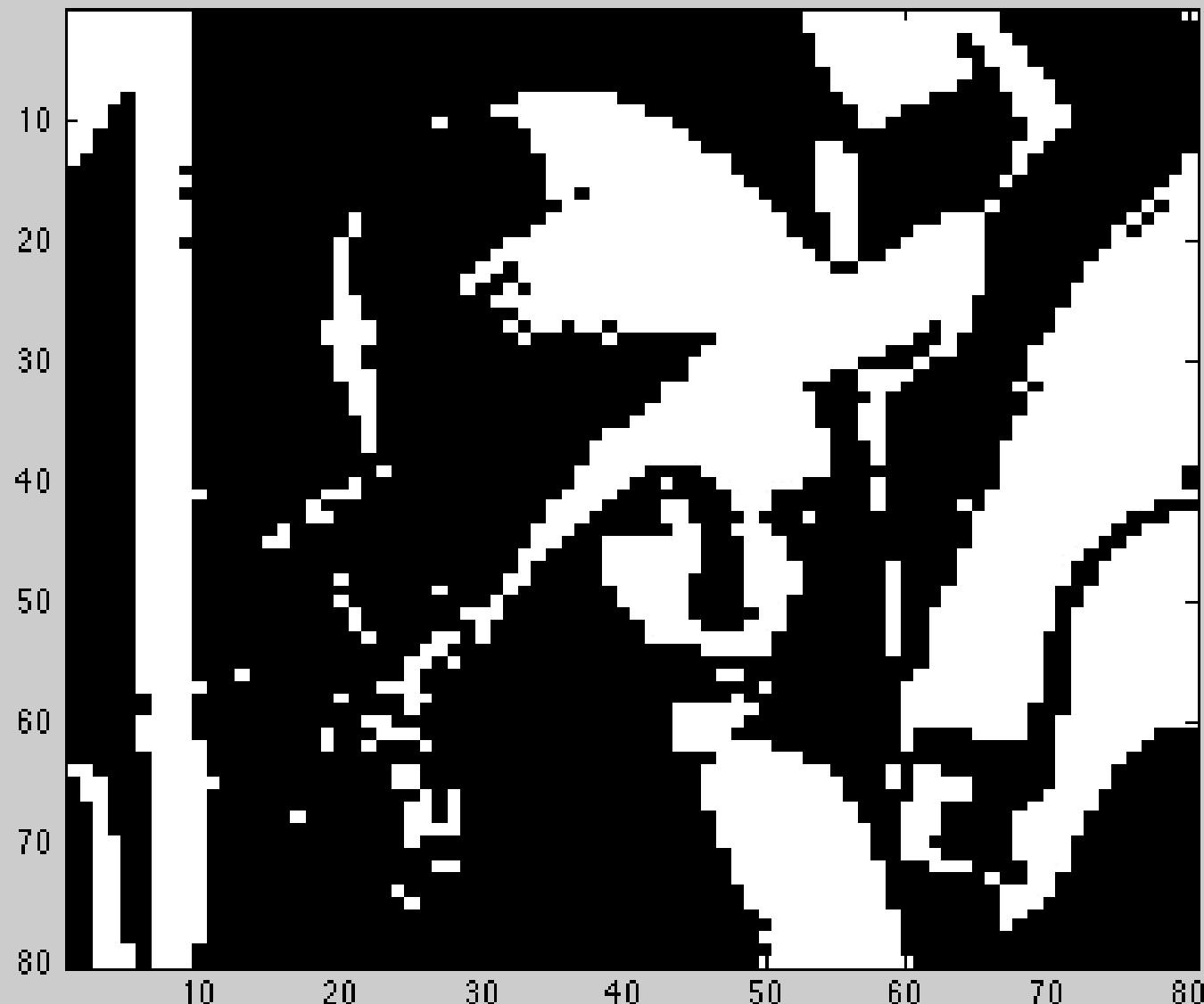
$$V_1(2) = 1$$

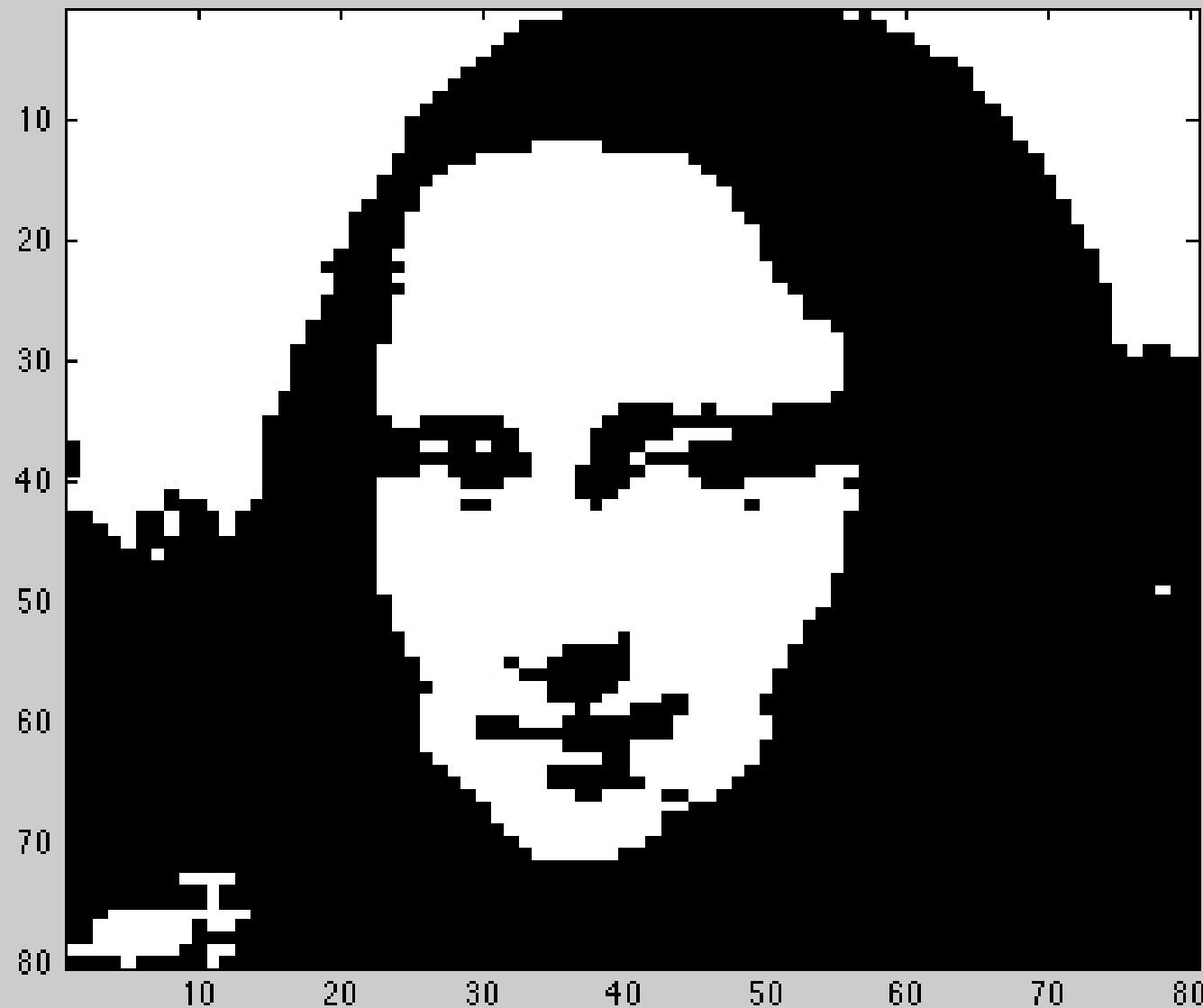
$$V_2(2) = 1$$

$$V_3(2) = -1$$

$$S = \sigma(WV)$$

no change





Continuous Hopfield Network

$$\checkmark \quad V_i = \pm 1$$

- $V(t+1) \leftarrow V(t)$
- $dV/dt = f(V)$

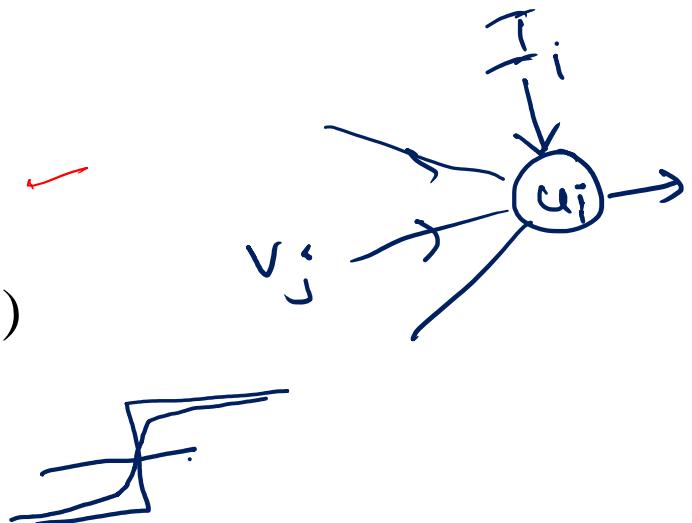
- discrete model $V_i = \pm 1$
- continuous model $V_i \in (-1, 1)$

Equations:

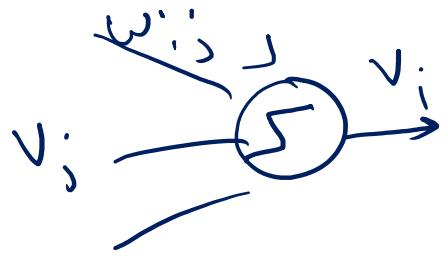
$$\frac{du_i}{dt} = -u_i + \sum_{j=1}^N w_{ij} V_j + I_i \quad (1)$$

$$V_i = g(\lambda u_i) = \tanh(\lambda u_i) \quad (2)$$

$\lambda \gg 1$

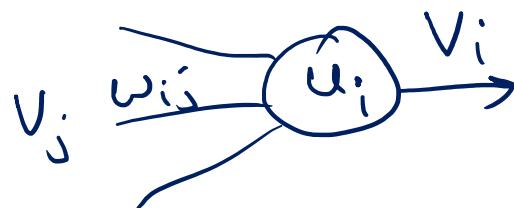


Discrete HNN



$$v_i(t+1) = g\left(\sum_{j=1}^n w_{ij} v_j(t)\right)$$

Cont. HNN



diff eqn.

Dt · dt/dt

$$\dot{u}_i = 0$$

$$u_i = \sum_{j=1}^n w_{ij} v_j + I_i$$

$$v_i = g(u_i) = g(\sum w_{ij} \dots)$$

$$V = g(u)$$

$$g^{-1}(v) = u$$

- Lyapunov or “Energy” Function:

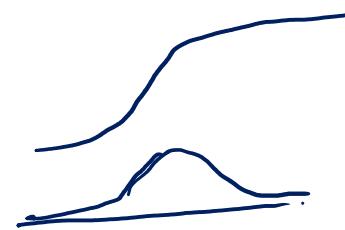
$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} V_i V_j + \sum_{i=1}^N \int_0^{V_i} g^{-1}(V) dV - \sum_{i=1}^N I_i V_i$$

$$\frac{dV_i}{dt} = g'(u_i) \frac{du_i}{dt}$$

- Proof:

$$\frac{dE}{dt} = - \sum_{i=1}^N \left(\sum_{j=1}^N w_{ij} V_j - u_i + I_i \right) \frac{dV_i}{dt}$$

$$= - \sum_{i=1}^N \frac{du_i}{dt} \frac{dV_i}{dt} = - \sum_{i=1}^N g'(u_i) \left(\frac{du_i}{dt} \right)^2 \leq 0$$



Therefore, E decreases monotonically with time.

Single neuron: Condition for stability

- Dynamics of a single neuron in the continuous Hopfield network may be given as:

$$\frac{du}{dt} = -u + w g(\lambda u)$$

assuming that the external input I_i is zero.

- At convergence we have,

$$du/dt = 0, \text{ or}$$

$$u = w \tanh(\lambda u).$$

$$\frac{du}{dt} = -u + w \tanh(\lambda u) \\ \Rightarrow ?$$

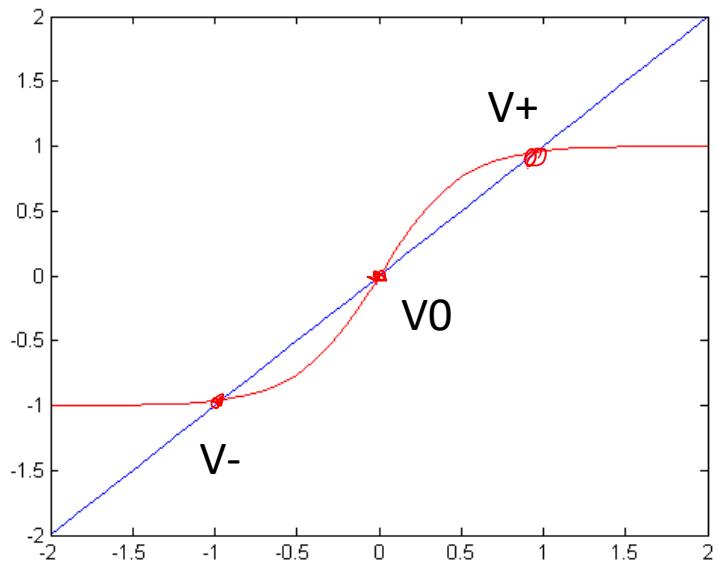
$$u = \tanh(\lambda u) \\ y = u; \quad y = \tanh(\lambda u)$$

- Solution to the last equation may be represented as intersection of two plots,

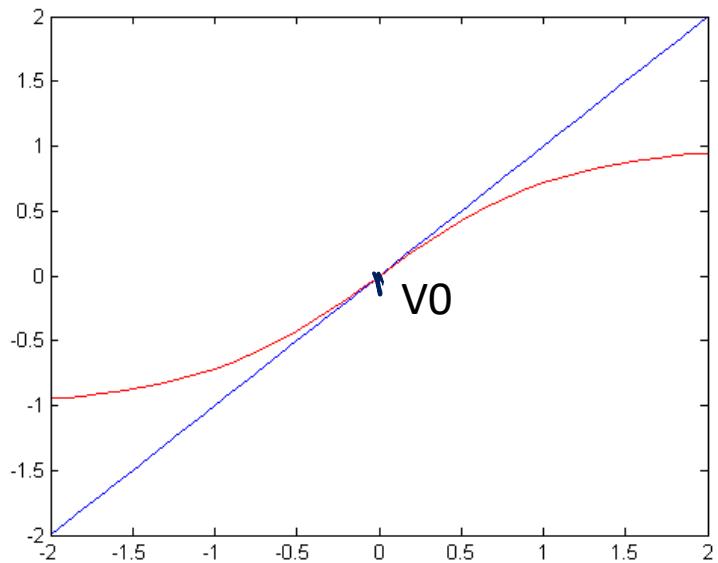
– $y = u$ (1)

– $y = w \tanh(\lambda u)$ (2)

which is graphically represented.



- The number of intersection points obtained depends on the value of λ .
- For $\lambda > 1$, we have three intersection points, and for $\lambda < 1$, we have only one intersection point at the origin



- For $\lambda > 1$: Let us call the three intersection points V_+ , V_- , V_0 (origin), which are also the stationary points of eqn.
- A stationary point is stable, if

$$\frac{d^2u}{dt^2} < 0 \quad \frac{d^2u}{dt^2} = -1 + w\lambda g'(\lambda u)$$
- The two terms on the right side are the slopes of the eqns. (1, 2) respectively.

- Therefore, $\frac{d^2u}{dt^2} < 0$ when the slope of the sigmoid, which is true for both V_+ , V_- but not true at the origin.
- Therefore, V_+ and V_- are stable but the origin is unstable.

- For $\lambda < 1$:
- In this case, there is only one intersection point at the origin.
- Since the slope of the sigmoid is less than 1 at the origin, the origin is stable.

A two-neuron system

- Let us consider a simple two-neuron system with weight matrix $W = \underline{[1 \ -1 ; -1 \ 1]}$.

$$\frac{du_1}{dt} = -u_1 + w_{11}V_1 + w_{12}V_2$$

$$S = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$\frac{du_2}{dt} = -u_2 + w_{21}V_1 + w_{22}V_2$$

$$\Rightarrow W = SS^T$$

$$V_1 = \tanh(\lambda u_1)$$

$$= \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

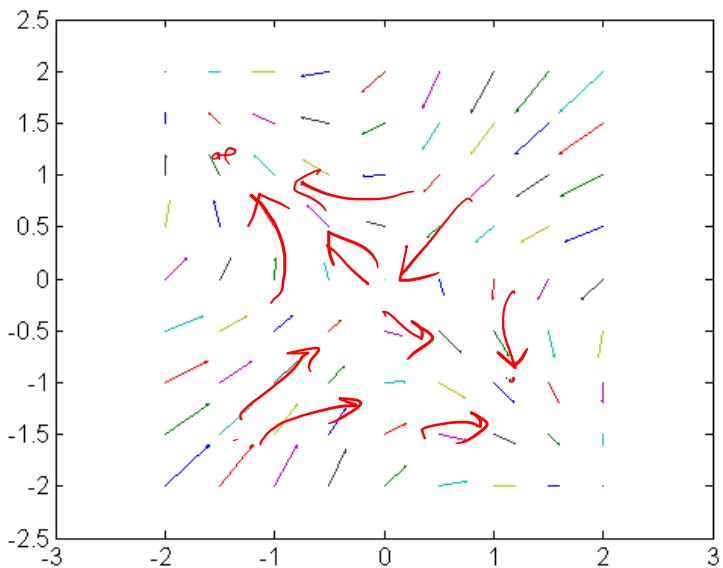
$$V_2 = \tanh(\lambda u_2)$$

Hebb's rule

for pattern storage.

$$\lambda = 2.$$

- The figure depicts the vector field defined by the above system.
- There are stable nodes at $(-2, 2)$ and $(2, -2)$ and the origin is a saddle node.



Problem

- In the original Hopfield model,
 - Neural activity evolves through time
 - Weights are calculated in one-shot
- It would be biologically more plausible if weights can be trained over time

The Adaptive Hebbian Learning in Hopfield Network

- In both discrete and continuous Hopfield network weights trained in a one-shot fashion and not trained incrementally as was done in case of Perceptron and MLP. There is a variation of Hopfield network in which weights are adjusted incrementally. Dynamics of the Hopfield network with adaptive weights is described below.

- (Activation dynamics) (1) $\tau_a \frac{du_i}{dt} = -u_i + \sum_{j=1}^N w_{ij} V_j + I_i$ ✓

- (Weight dynamics) (2) $\tau_w \frac{dw_{ij}}{dt} = -w_{ij} + (V_i V_j)$ —

- Where $(\tau_a \ll \tau_w)$ and

$$V_i = \tanh(\lambda u_i), \quad \lambda \gg 1$$

Helmholtz
 $w_{ij} = \Sigma_i \Sigma_j$
 no dynamics

- The combined system has an expanded “energy” function given as,

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} V_i V_j + \sum_{i=1}^N \int_0^{V_i} g^{-1}(V) dV - \sum_{i=1}^N I_i V_i + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij}^2 \quad (3)$$

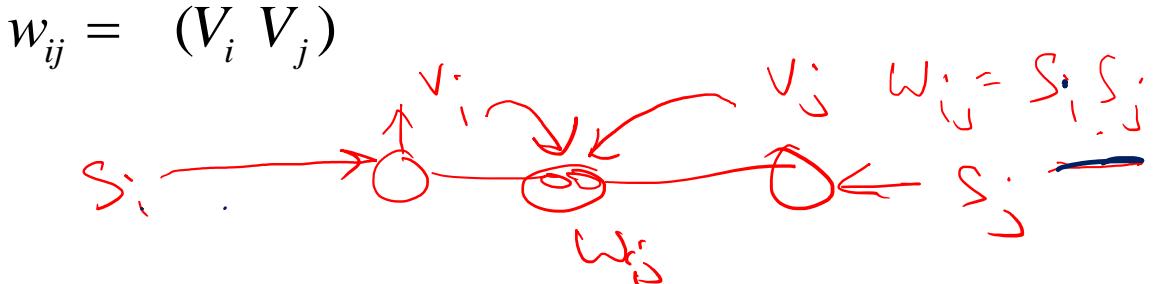
- Note that when the weights stabilize,

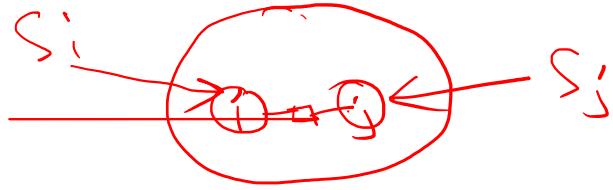
or (4)

$$\frac{dw_{ij}}{dt} = 0 \qquad w_{ij} = (V_i \ V_j)$$

- Which is similar to the Hebb's rule of the earlier Hopfield models.
- But the rule would amount to storing the network state in the weights,

$w_{ij} = s_i s_j$





- The external input imposes itself on the network state V , which in turn can be written onto the network weights w .

$$I \rightarrow V \rightarrow w$$

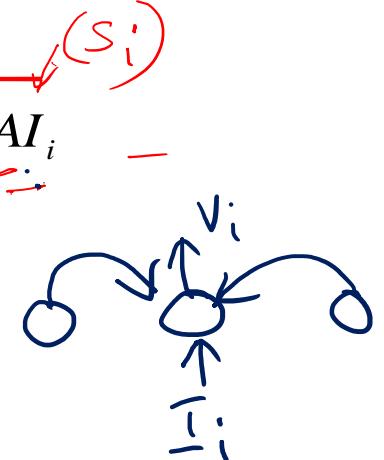
To ensure such a steady transfer of information from the external input to the network weights, we need certain additional mechanisms. To this end, we add a few coefficients (A , B and C) to the above eqns. as follows

Activation dynamics: (5)

$$\tau_a \frac{du_i}{dt} = -u_i + B \left(\sum_{j=1}^N w_{ij} V_j \right) + AI_i$$

Weight dynamics: (6)

$$\tau_w \frac{dw_{ij}}{dt} = C \left(-w_{ij} + (V_i V_j) \right)$$



Storage

$$w_{ij} = v_i v_j \quad \text{already } \propto q_m(z)$$

(not $s_i s_j$)

$$= s_i s_j \quad (\text{desired})$$

Want :-

$$s_i \rightarrow v_i \rightarrow w$$

(ok)

$$\tau \frac{du_i}{dt} = -u_i + A s_i$$

$$u_i = A s_i$$

$$v_i = r_h(u_i) = r_h(-A s_i)$$

$$\underline{v_i \approx s_i}$$

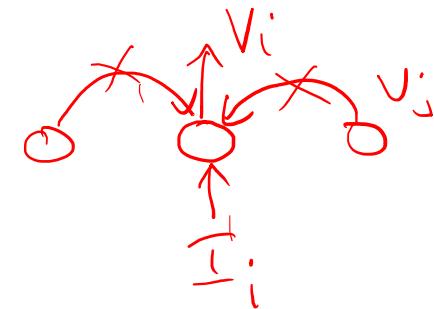
$$A > 0$$

eqn 2

$$\frac{dw_{ij}}{dt} = -w_{ij} + v_i v_j$$

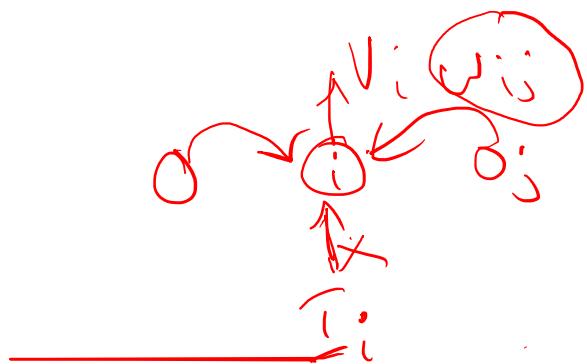
$$0 = -w_{ij} + v_i v_j$$

$$w_{ij} = v_i v_j \\ \approx s_i s_j$$



Recall

$$\Delta w_{ij} = 0 \text{ (no change in weights)}$$



- We divide the operation of the network into separate
 - 1) storage and
 - 2) retrieval stages.

Storage dynamics

- During storage the pattern to be stored is presented as external input I . The network state, V , must copy I . The network state V is then written onto the weights by eqn. (6).
- For this to be possible, in eqn. (5), 'A' must be a large positive number. Similarly 'B' must be a small positive number so that it does not allow the preexisting weights to influence the current state of the network.
- During storage, obviously the weights have to be adapted. Therefore, 'C' must be a finite positive number.

Retrieval dynamics

Retrieval dynamics must be obviously dominated by the preexisting weights;

- Therefore B should be large positive number.
- The external input should only be suggestive of the pattern that needs to be retrieved, and therefore need not be very large. Therefore 'A' must be a small number.
- During retrieval the weights are in principle not adjusted; therefore 'C' is zero.

Summary

- The Continuous Hopfield network with adaptive weights must have its coefficients A, B, C to be set as follows:
 - Storage stage: high A, low B, and high C.
 - Retrieval stage: low A, high B and zero C.
- We will now see that mechanism of the type described above can be seen in Hippocampus, a part of the brain involved in memory.

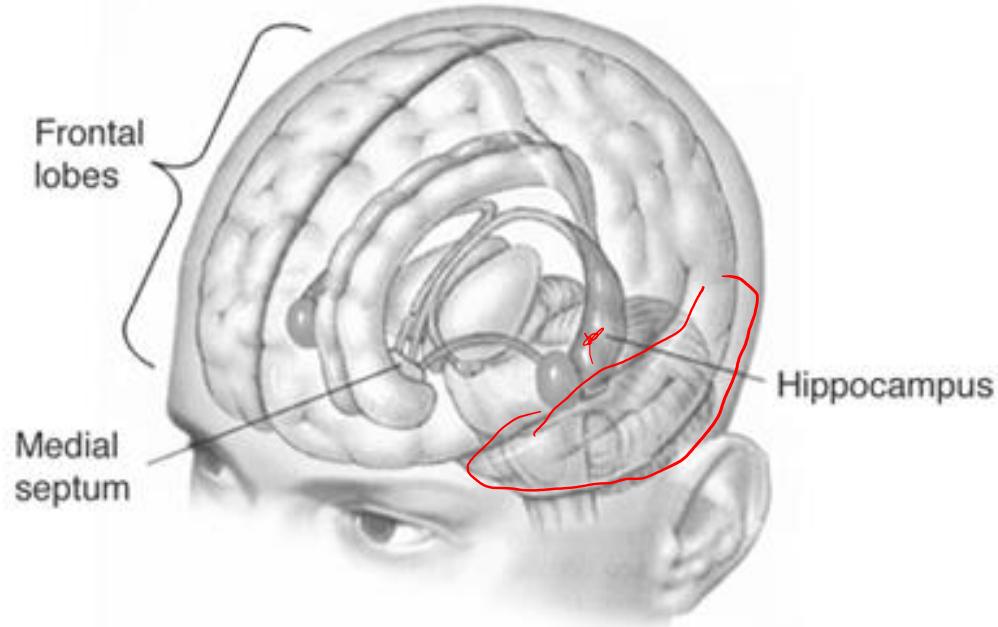
Hippocampus – memory storage and recall

- We have seen in Chapter 2 that hippocampus (HC) is a “scratch pad” of memories.
- Damage to HC is known to impair our ability to store and retrieve memories.
- Earliest knowledge about the memory-related functions of HC came out of studies of a patient known as Henry Molaison, usually referred to as HM.



HM

Both hippocampi removed
To treat epilepsy

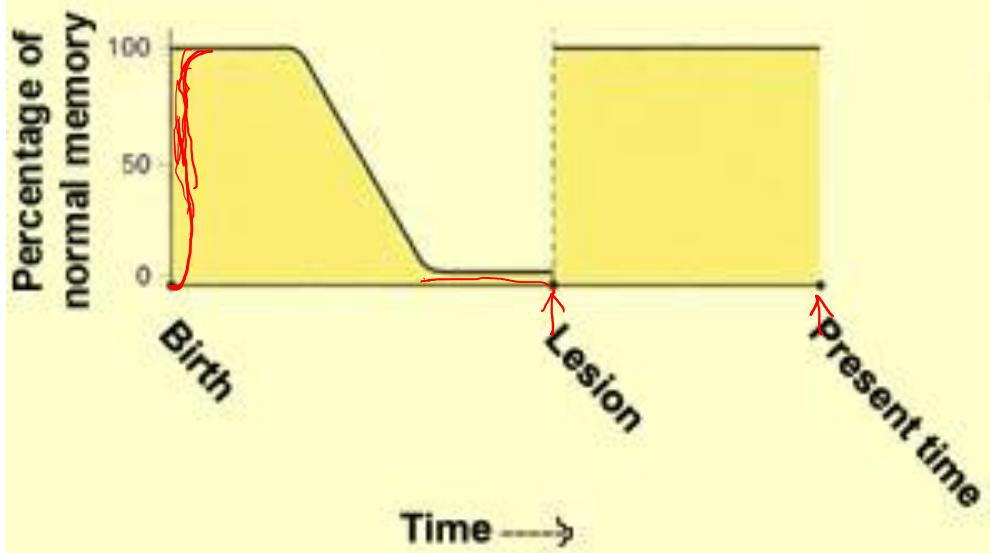


- HM had his medial temporal lobe removed as a treatment to his epileptic attacks, with which he was suffering since his childhood.
- Loss of HC which is located in the temporal lobe, cause memory impairment or amnesia. He suffered from two kinds of amnesias.

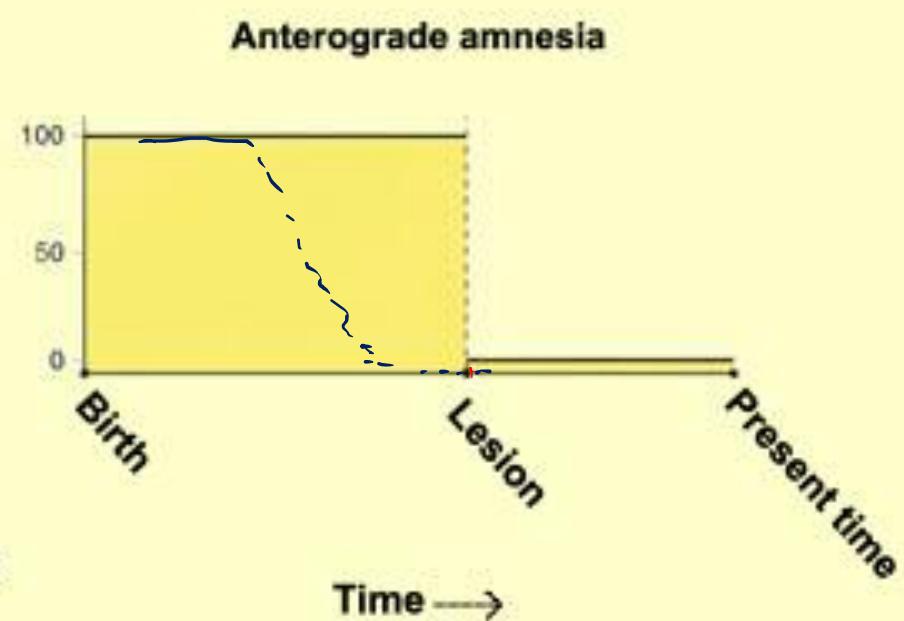
Amnesias

- Retrograde amnesia: inability to recall old memories.
 - Had poor memory of events that happened immediately before his surgery, but older memories were intact.
- Anterograde amnesia: inability to store new memories.

Retrograde amnesia



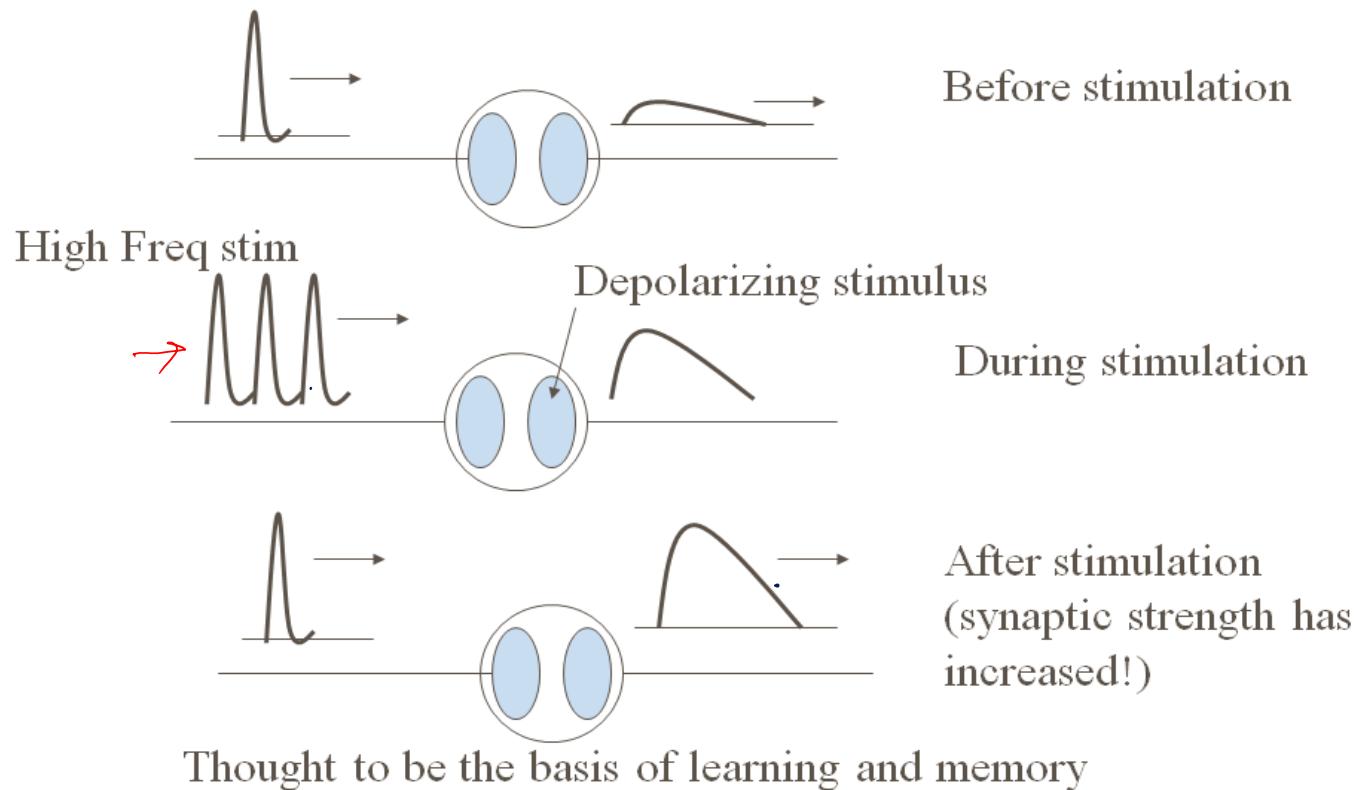
Anterograde amnesia



Hebbian Learning and Long-term Potentiation (LTP) in hippocampus

- We have seen that synaptic modification is the key substrate to learning and memory phenomena in the brain.
- Particularly we have seen how Hebbian learning can enable storage of information as attractors in network dynamics.
- Although when Hebbian learning was proposed in late '40s, it was only a hypothesis, experimental evidence for the same was first discovered in the synapses of HC in the '60s.

Synaptic Strength can vary: Long Term Potentiation (LTP)



A schematic of Long-term Potentiation (LTP). Top figure shows a weak EPSP in response to the single Action Potential. Middle figure shows the synapse stimulated by a high frequency volley of APs on the pre-synaptic side, and a depolarizing stimulation on the post-synaptic side. The bottom figure shows that the response of the synapse to a single AP is stronger than before.

- These studies have essentially shown that when both pre- and post-synaptic ends are simultaneously stimulated, the synapse gets strengthened or potentiated.
- This phenomenon is known as Long term Potentiation (LTP).
- When only one side is stimulated, the synapse gets weakened or depressed, a phenomenon known as Long-term Depression (LTD).

- In studies of this kind, typically, the presynaptic terminal is stimulated by a high-frequency volley of impulses, which amounts activation of the presynaptic side.
- On the post-synaptic side the membrane is depolarized by injecting a positive current. Initial strength of the synapse is assessed by stimulating the presyanptic side with a low frequency volley of impulses, and amplitude of the resulting PSP is noted.

- After stimulating the pre- and post-synaptic sides simultaneously, as described above, for about 10-15 mins, the synaptic strength is again assessed by stimulating the pre-synaptic side with a low frequency volley.
- It is seen that the PSP now is greater than the PSP observed before stimulation

- LTP, which is a biological substrate of Hebbian learning, has been found to occur in many hippocampal synapses.
- But for pattern storage we require Hebbian learning occurring in a network with strong recurrent connections.
- Does HC have strongly recurrent connectivity?
- To answer this question, let us consider the anatomical connectivity patterns within the HC and also its connections to other parts of the brain, particularly the cortex.

- What is the connection between the Hopfield network and Hippocampus?

Cortex and the Hippocampus

connections:

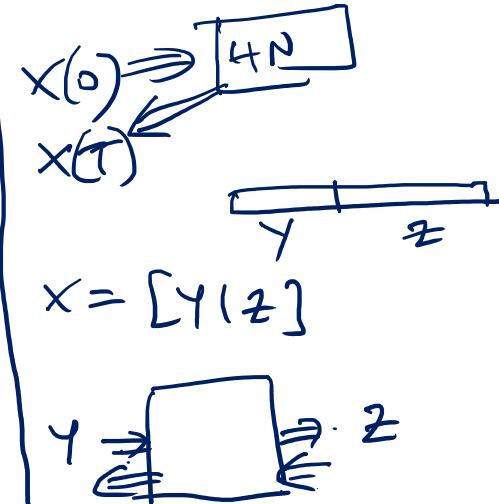
- HC is basically a system for association. It captures relationships among representations of the same object present in different sensory modalities.
- For example, when you holding a cup in your hand, your visual system processes the appearance of the cup, - its size, color, distance from your eyes etc.
- Your somatosensory system processes the feel of the cup, its hardness or softness, its temperature etc.

- Both these streams of sensory information contain information about the cup. This high level information about the cup is extracted at higher stages of sensory cortices and forwarded to HC.
- Thus HC receives compact high-level representations of the object. HC also receives reward related information from prefrontal cortex and subcortical structures like amygdala.
- HC combines the compact representation of the object, and the relevant reward information and decides if the information is worth storing for longer-term back in the cortex, or must be discarded.

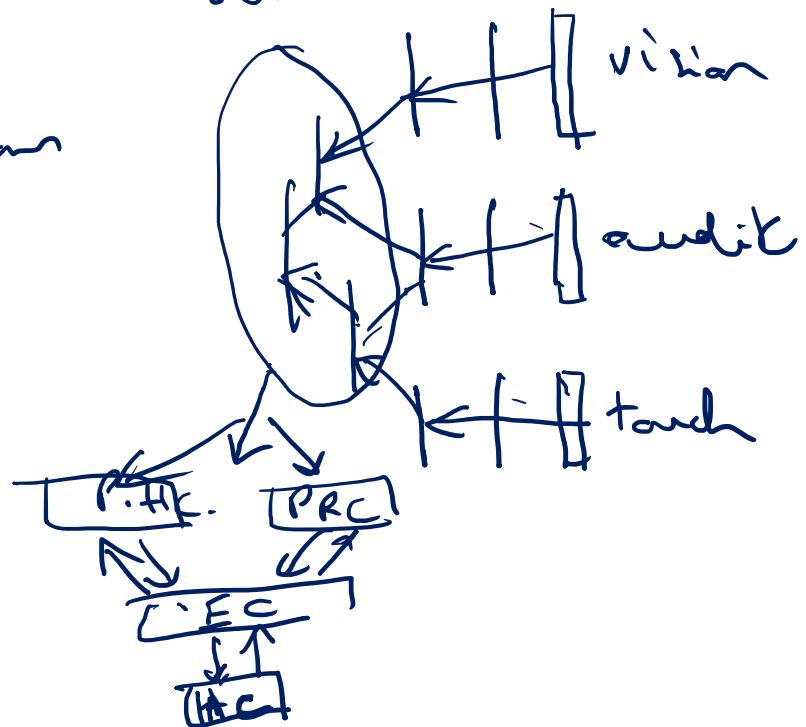
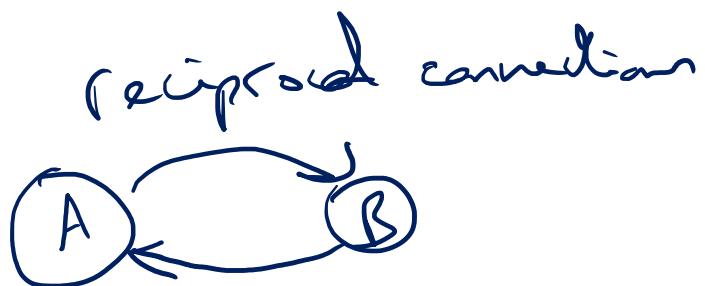
- HC receives inputs from large parts of the neocortex which project to parahippocampal gyrus and perirhinal cortex.
- The last two areas project to Entorhinal cortex which is considered the gateway to HC.



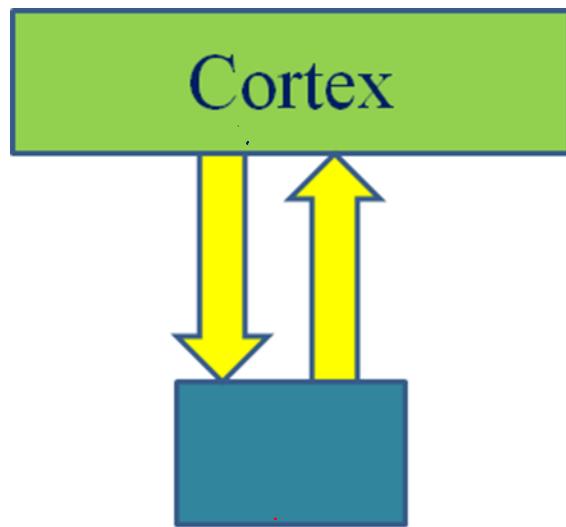
Sensory association areas.



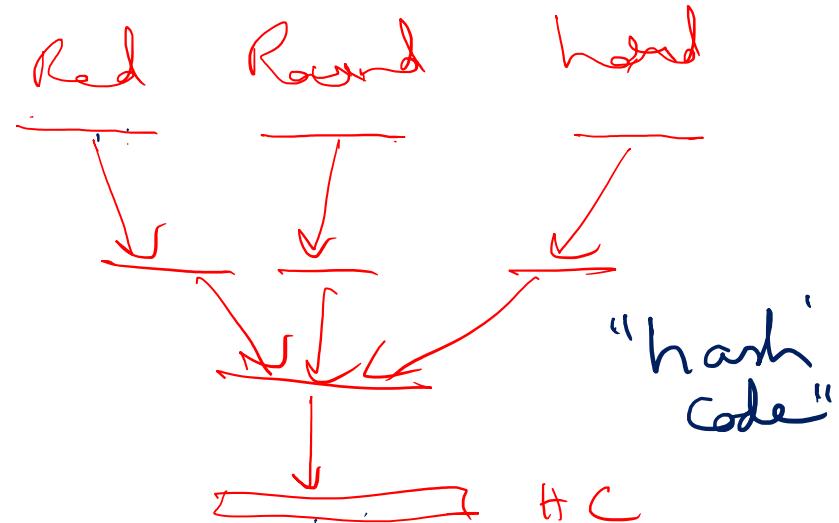
Sens. areas.



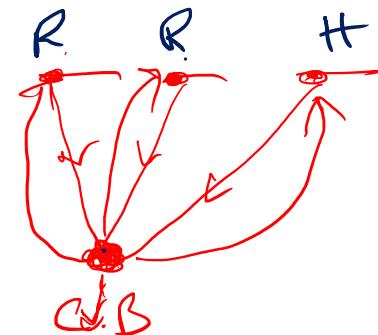
Amoebic memory



Hippocampus



H C

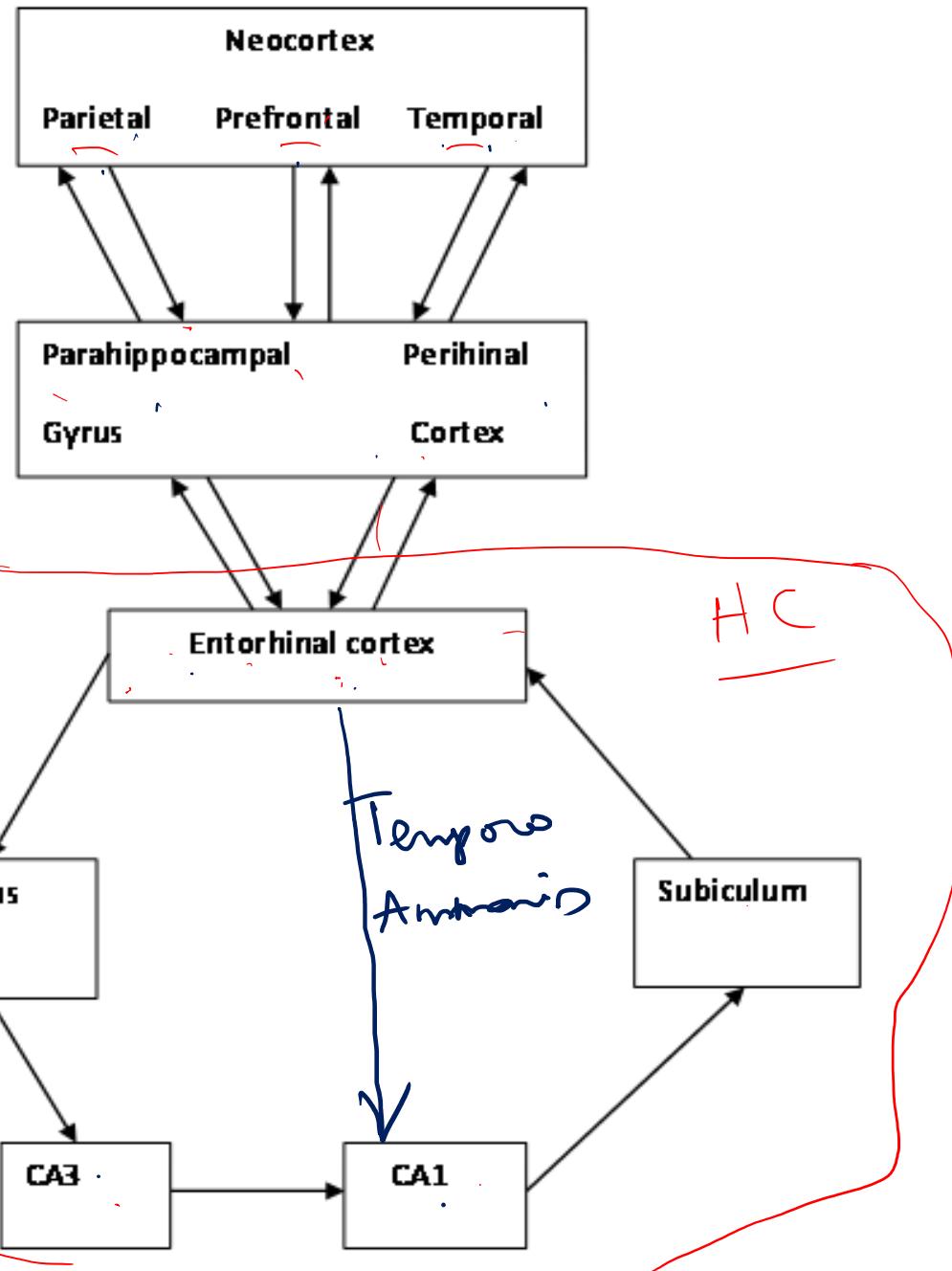
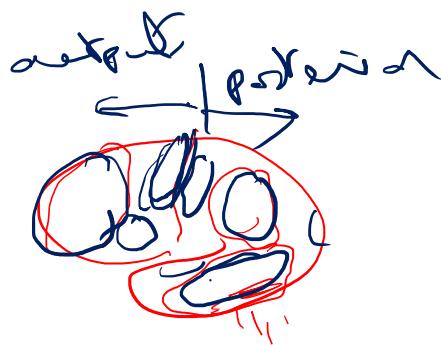


G.C.B.

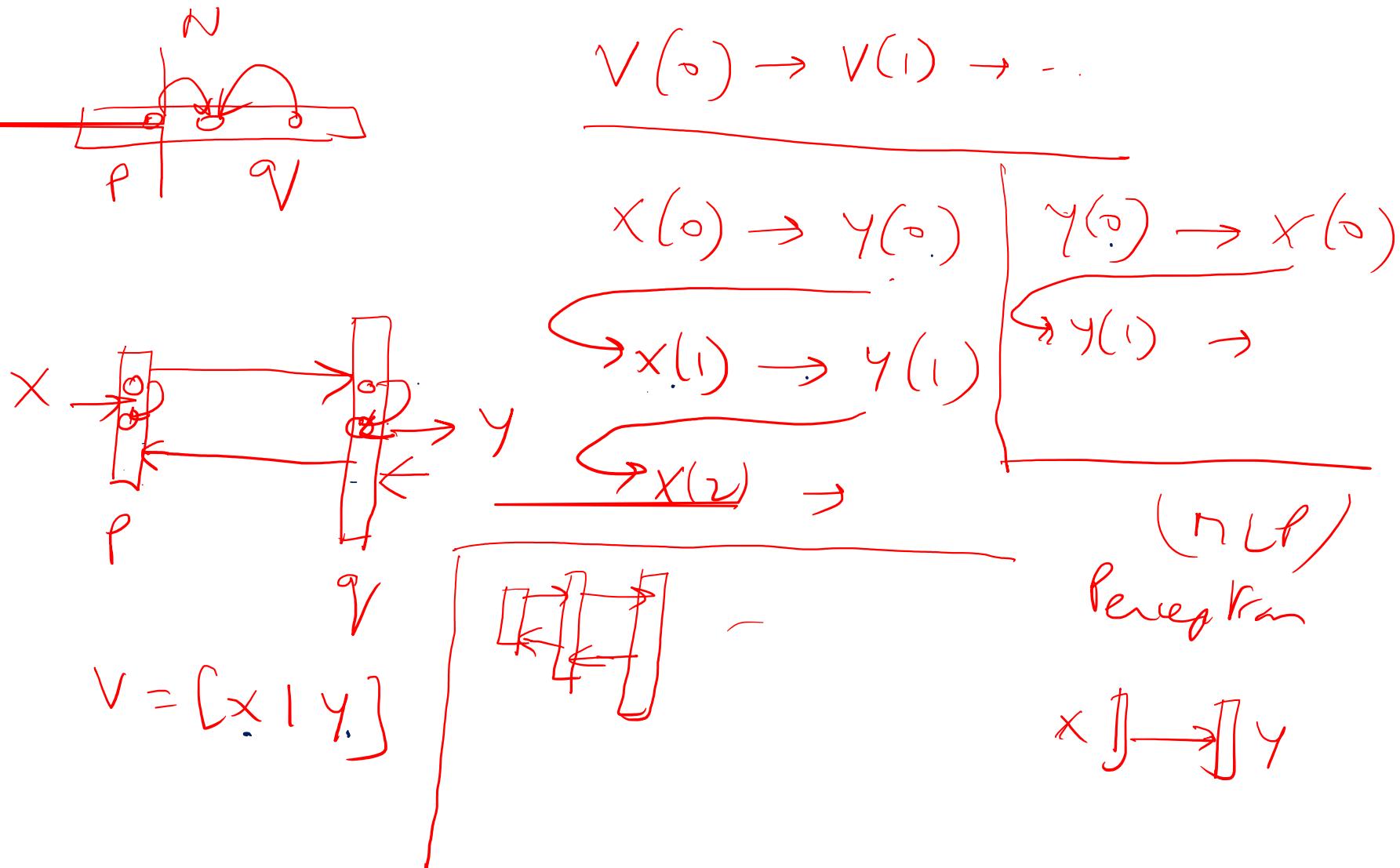
R

R

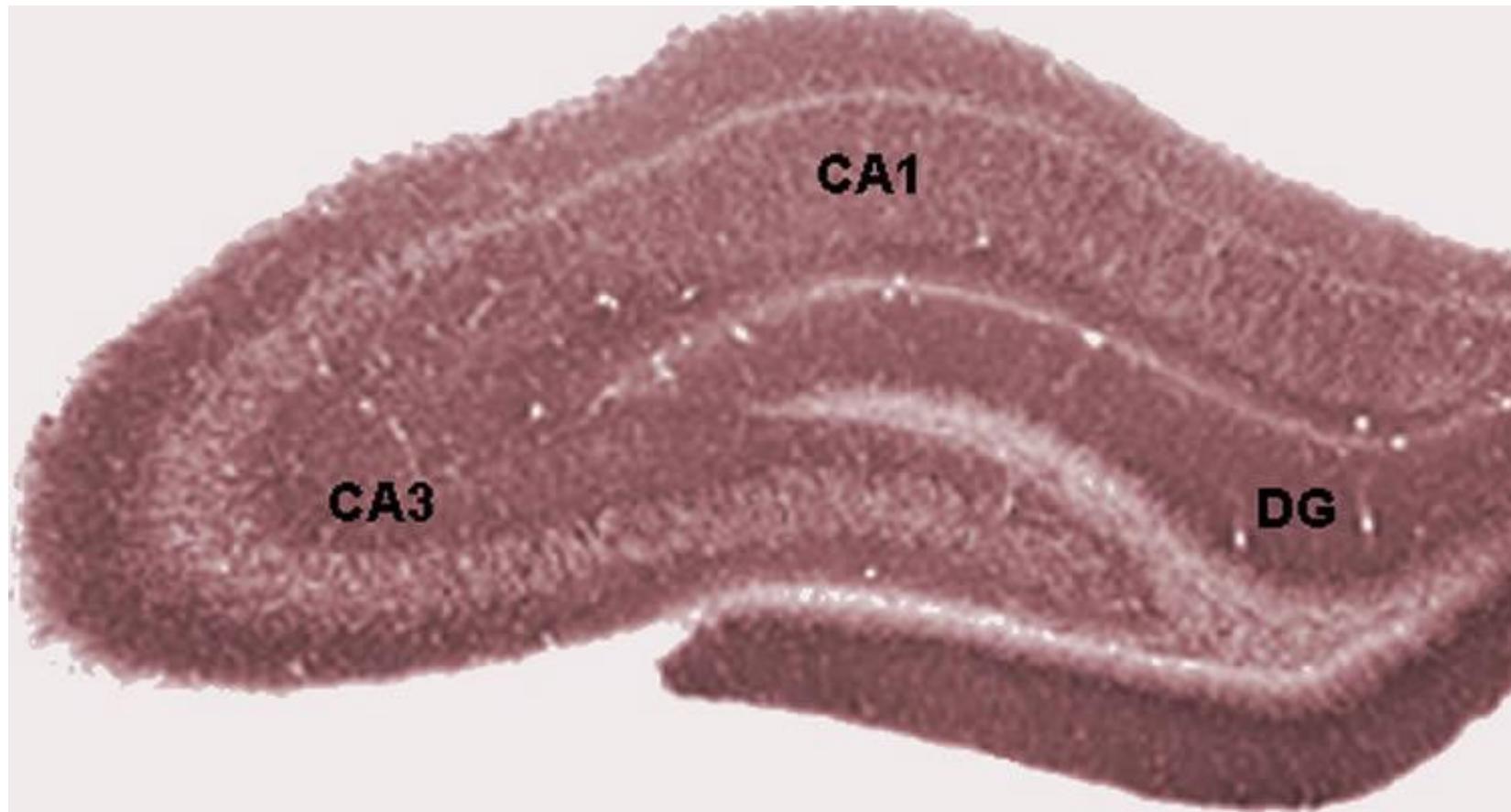
H



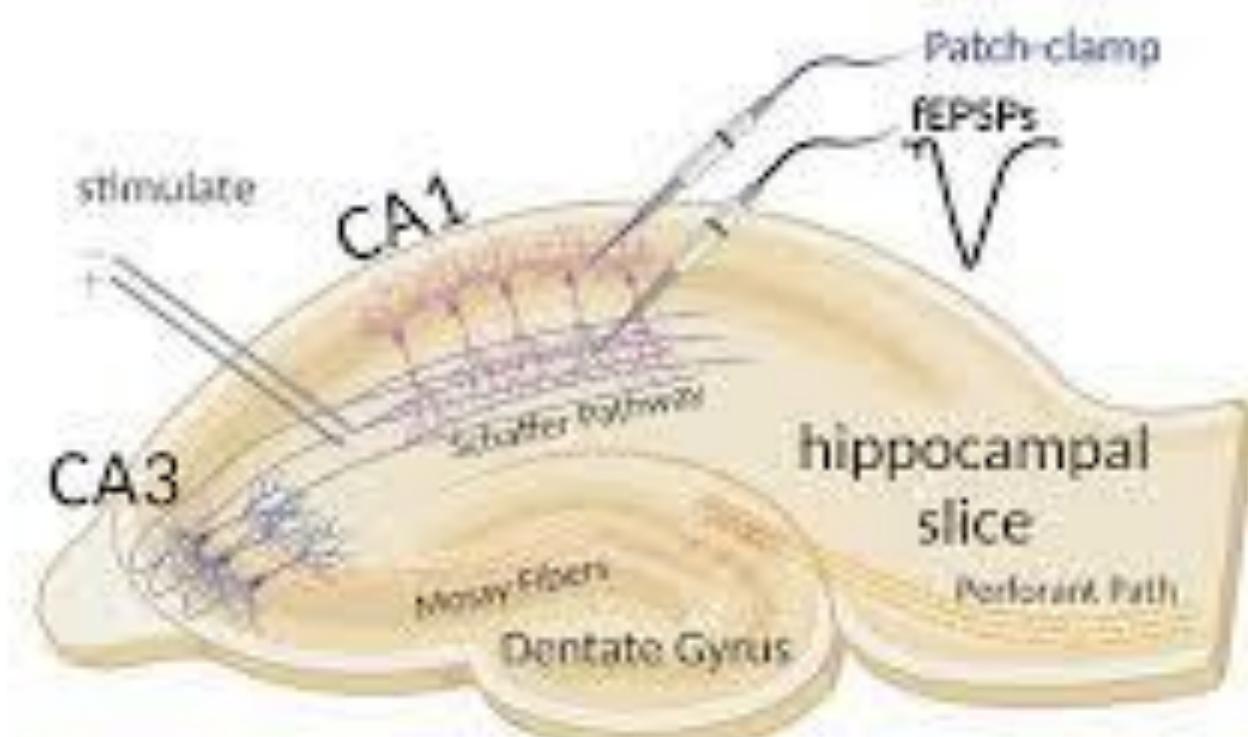
Bidirectional Associative Memory

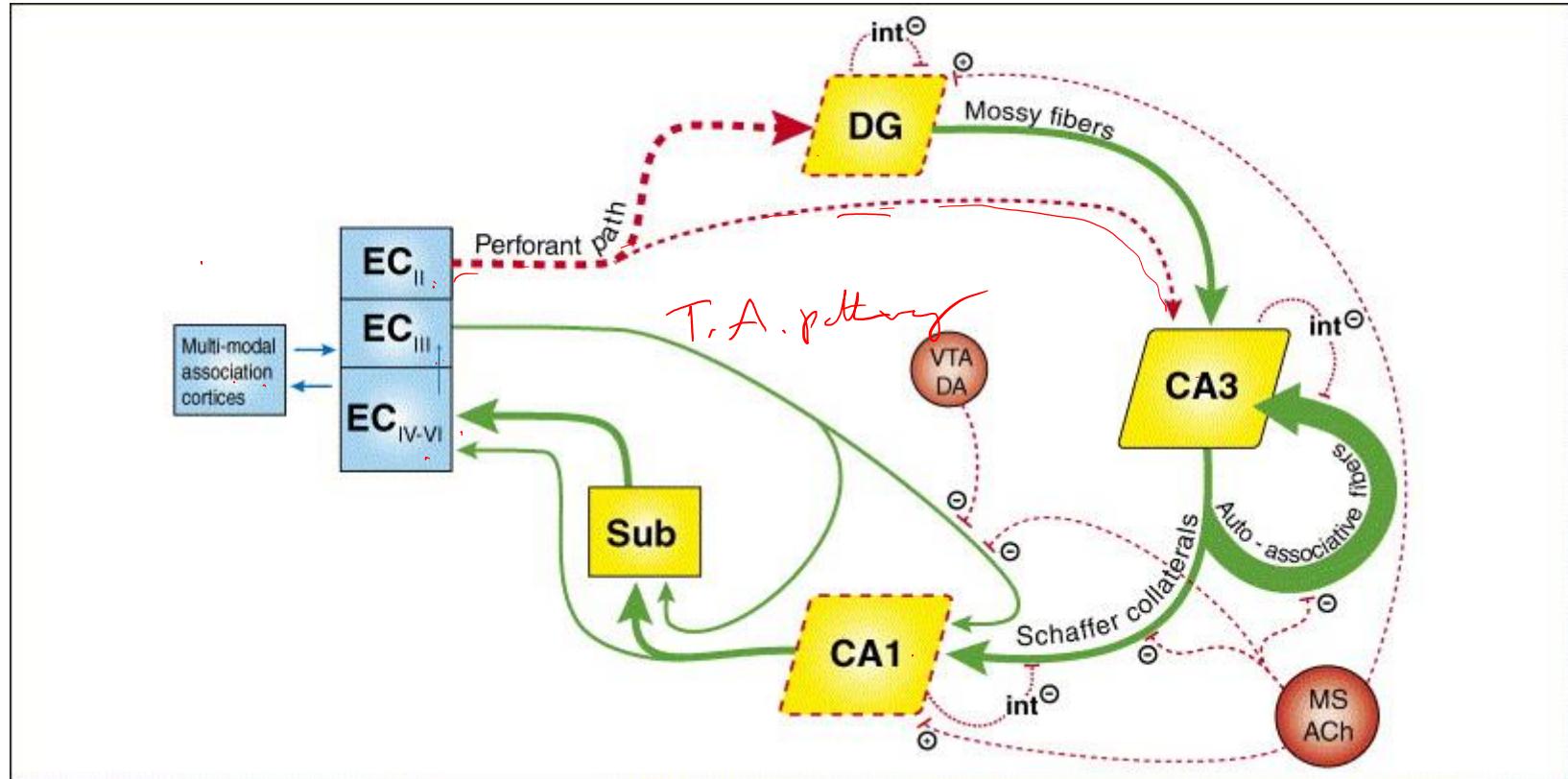


Hippocampal slice

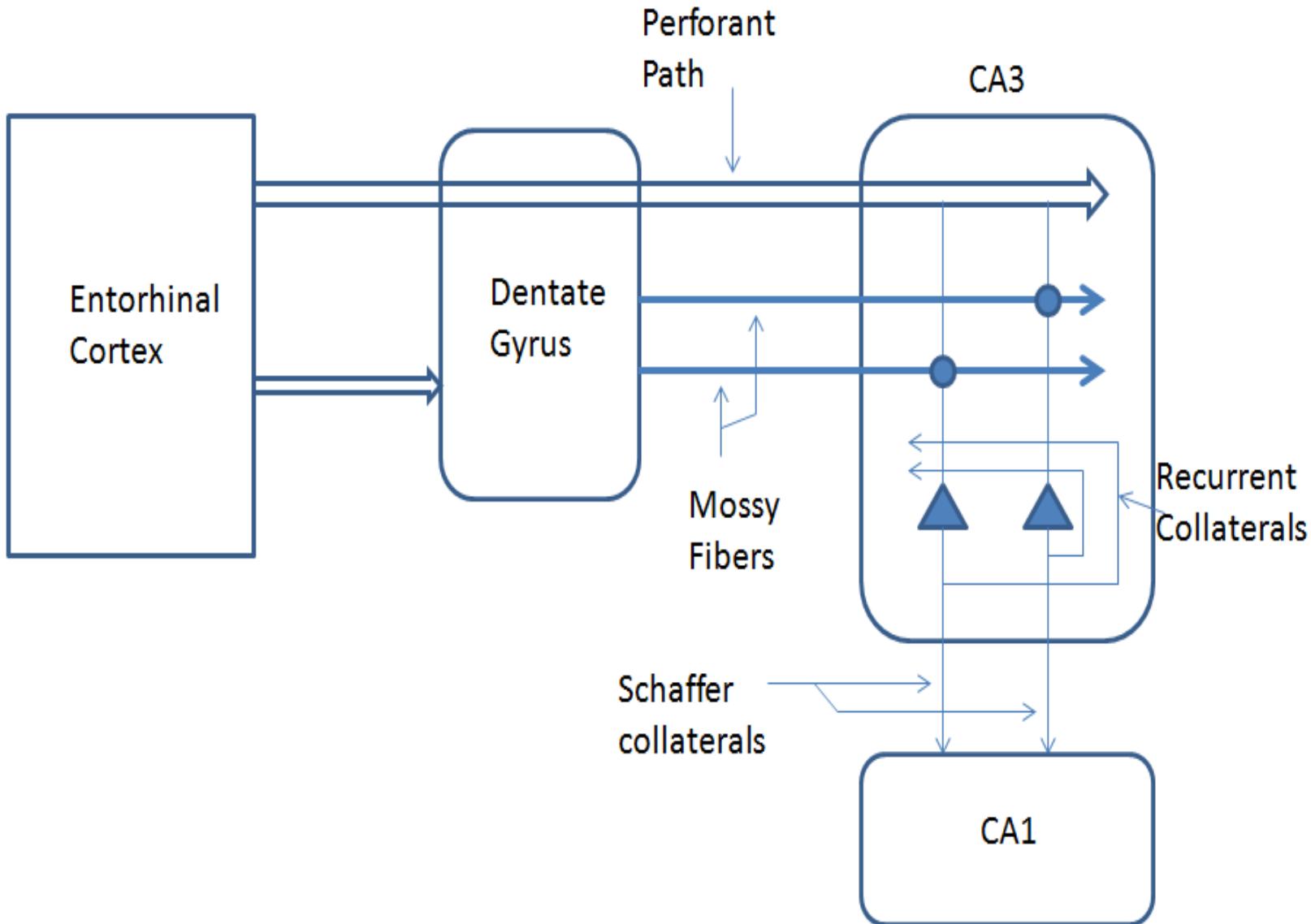


Hippocampal slice





<https://www.memorangapp.com/flashcards/127486/Limbic+System/>

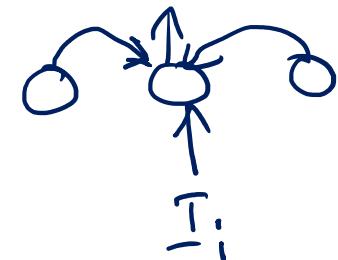
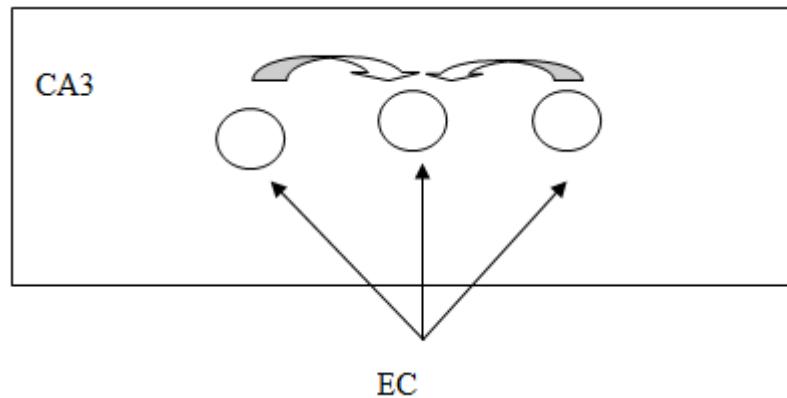


Parallel pathways within HC

- One branch of EC output into the HC goes to Dentate Gyrus, which in turn projects to CA3 via fibers known as mossy fibers.
- EC also projects directly to Dentate gyrus via the perforant pathway.
- The mossy fibers are known to have strong synapses, while the perforant pathway has a weaker influence on CA3.

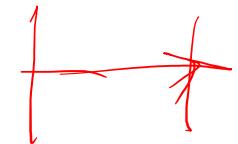
Is CA3 a Hopfield Network?

- There is a theory that CA3 could be operating like a Hopfield network

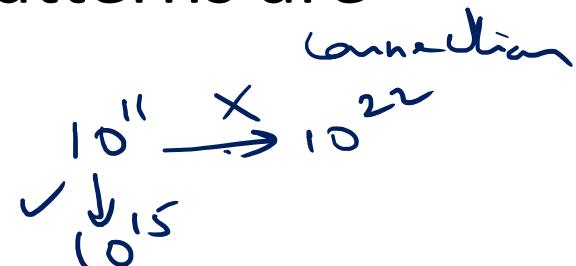


Recurrent connections in CA3

- CA3 neurons have extensive recurrent connections.
- In rat there are about 300,000 pyramidal neurons in CA3, with each of them receiving inputs from about 12,000 other CA3 pyramidal neurons. In contrast, each CA3 pyramidal neuron receives only 4000 inputs from EC.



- Each CA3 neuron receives inputs from about 4% of all CA3 pyramidalis,
 - high recurrence - high compared any other brain region.
- These recurrent connections are modifiable by LTP. *plastic*
- A strong possibility of CA3 acting as some sort of a Hopfield network in which patterns are stored as attractors.

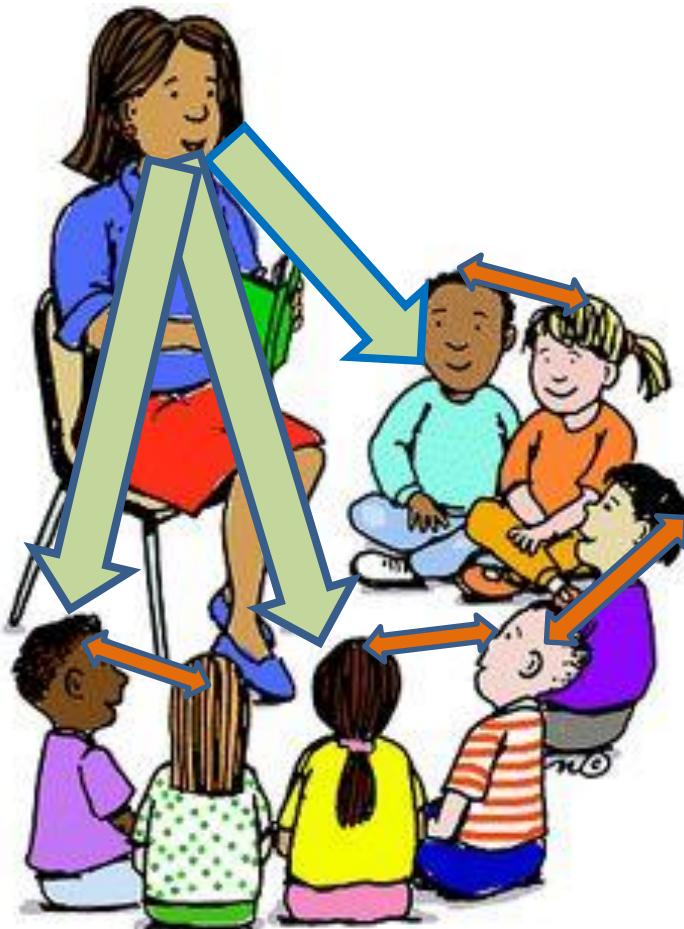


Storage and Retrieval in CA3:

- We have considered evidence that CA3 has features of an autoassociative network –
 - high recurrence and plastic synapses.
- But for the network to be useful as a memory, it must be operable in **two modes** –
 - **Storage** ↗
 - **Retrieval** ↘
- Special conditions are required

The Classroom Analogy (“STORAGE”)

“Learning
the Lesson”
Stage



- Teacher’s instruction dominates ↑ A
- Chatter among the children is low ↓ B
- Children are in a state of receptivity, eager to LEARN the lesson

↑ c

The Classroom Analogy (“RETRIEVAL”)

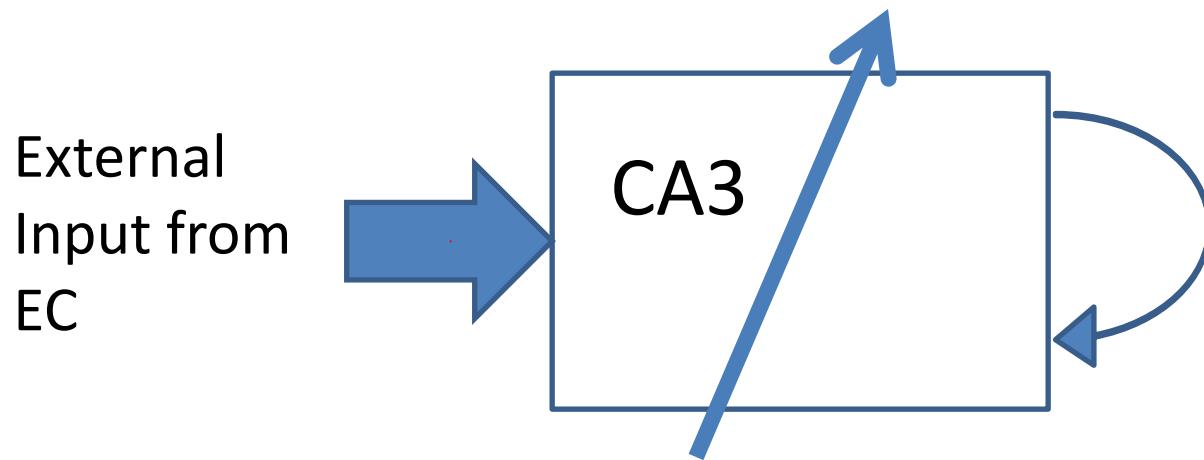
“Discussing
and
Reproducing
the Lesson”
Stage



- Teacher's instruction is LOW A ↓
- Conversations among the children are HIGH B ↑
- Children DO NOT LEARN anything new; they only discuss among themselves and reproduce what they know C ↓

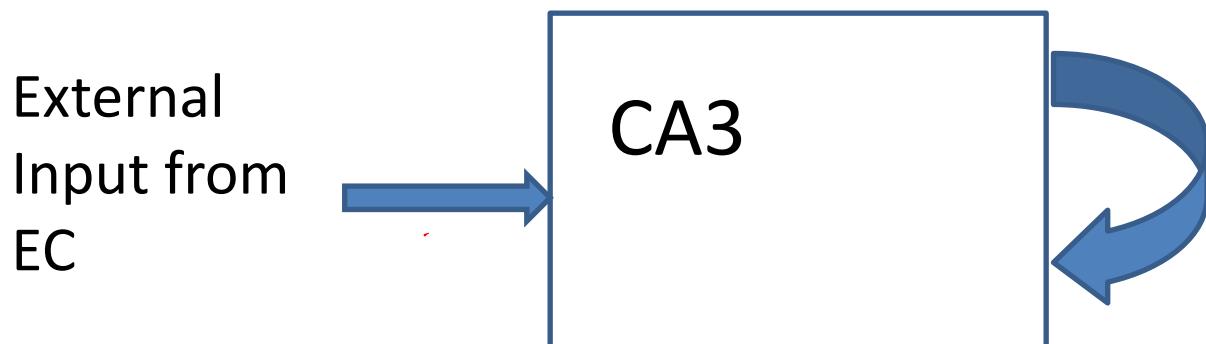
Conditions for Storage

- During storage:
 - External input is STRONG
 - Recurrent connections are WEAK
 - Recurrent connections are PLASTIC



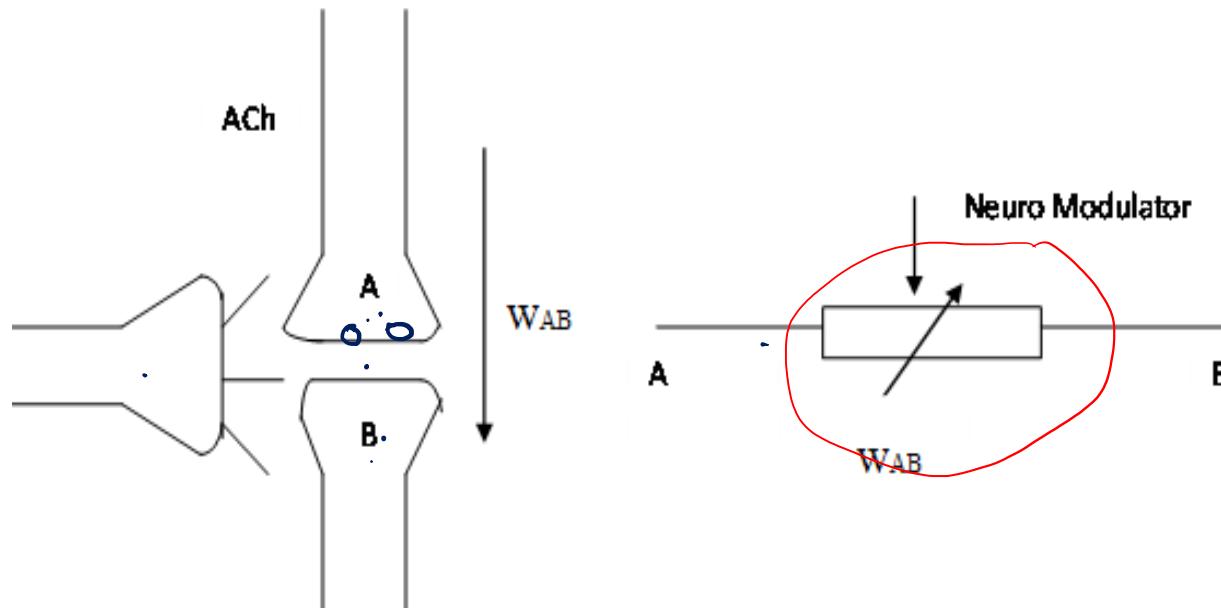
Conditions for Retrieval

- During ~~storage~~^{retrieval}:
- External input is WEAK
- Recurrent connections are STRONG
- Recurrent connections are NOT PLASTIC

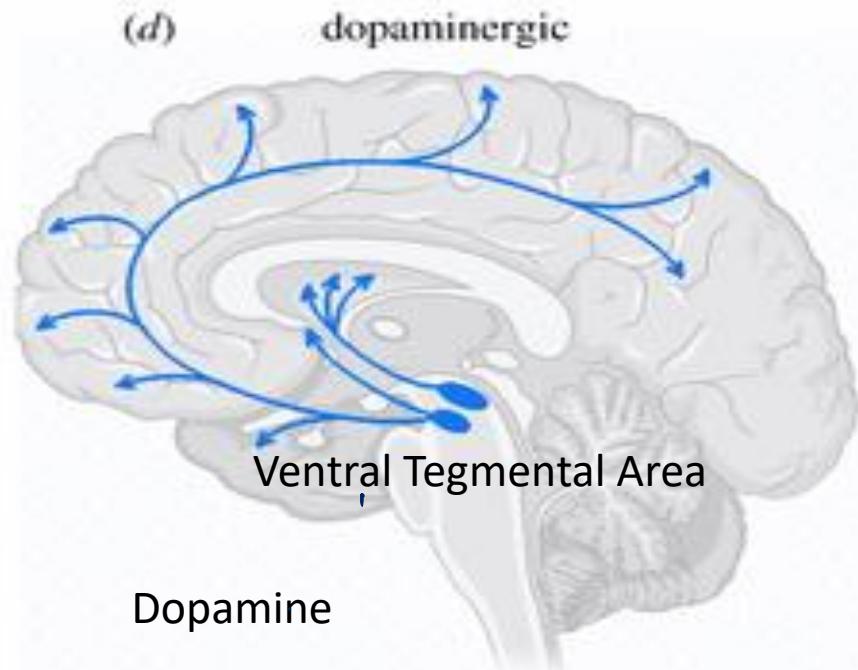
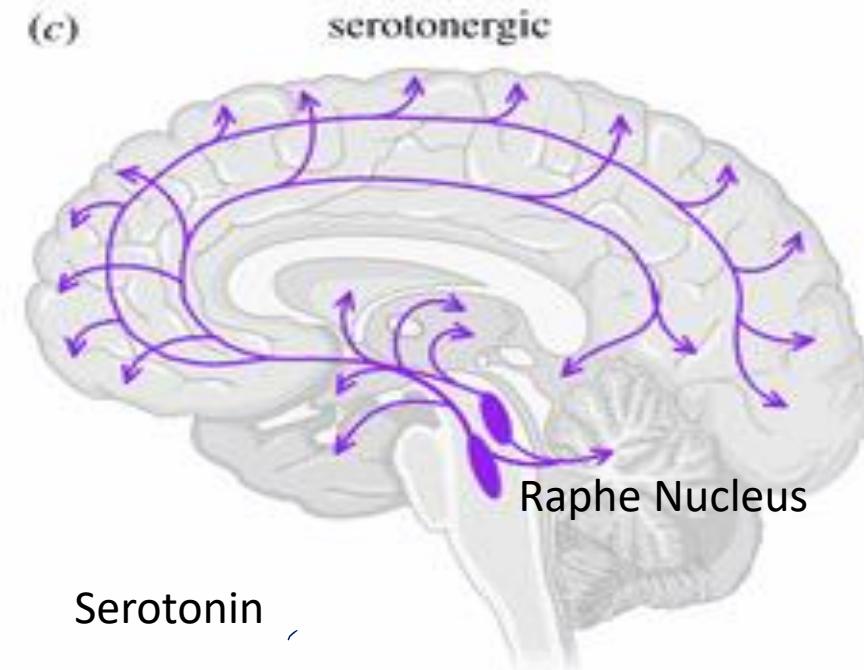
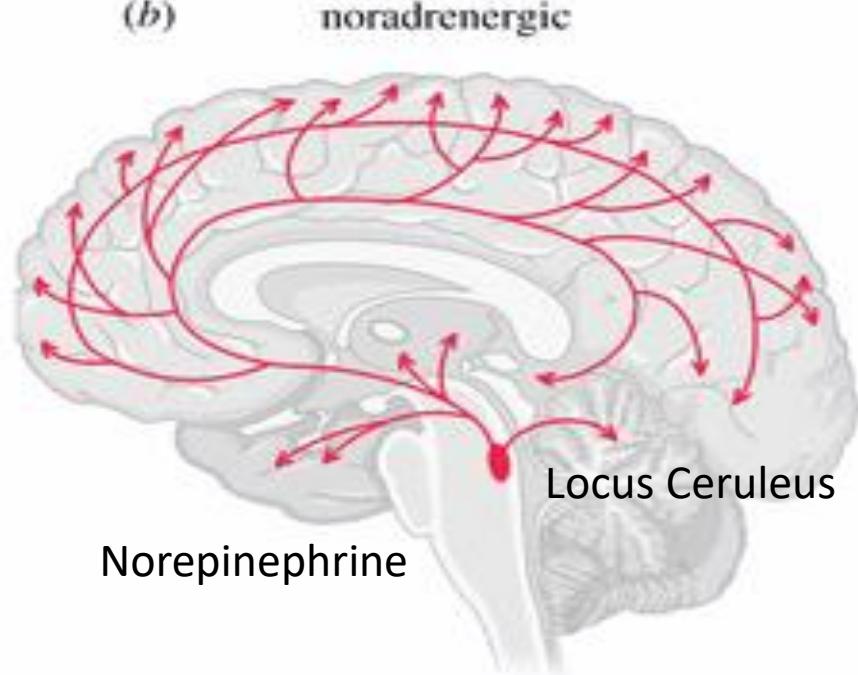
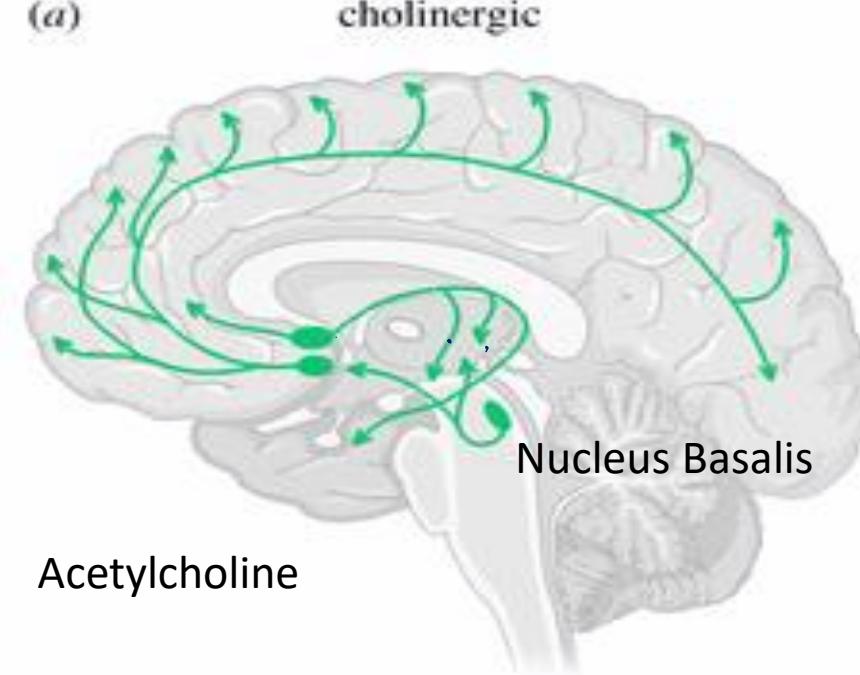


- CA3 has two external inputs:
 - from EC by perforant pathway, and the other
 - from EC via Dentate Gyrus by mossy fibers.
 - Perforant pathway inputs are weaker than mossy fibers.
- Plasticity of CA3 recurrent connections:
 - Acetylcholine (Ach) is a neuromodulator that is capable of controlling the strength and plasticity of recurrent connections in CA3.

Neuromodulator



Neuromodulator controls plasticity
It can rapidly change the synaptic strength



Acetylcholine (ACh) as a Neuromodulator

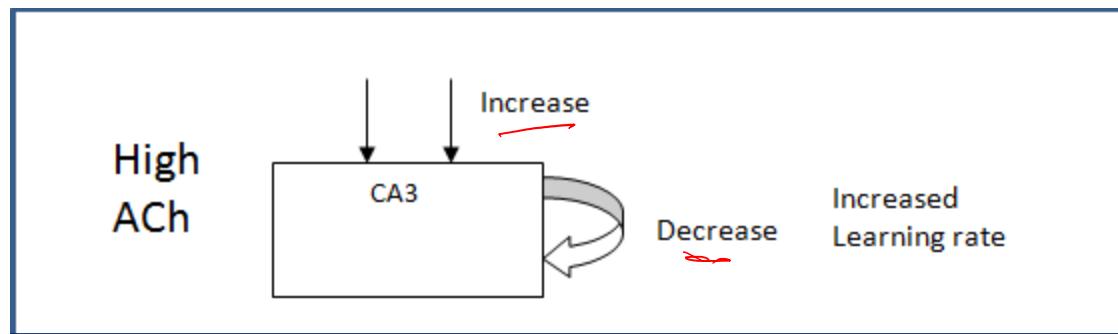
- Some common neuromodulators in the brain are:
 - Acetylcholine, dopamine, norepinephrine and serotonin
- Blockage of ACh transmission in HC is known to interfere with memory storage.
- Scopolamine (ACh antagonist) → temporary amnesia

Scopolamine in a memory task

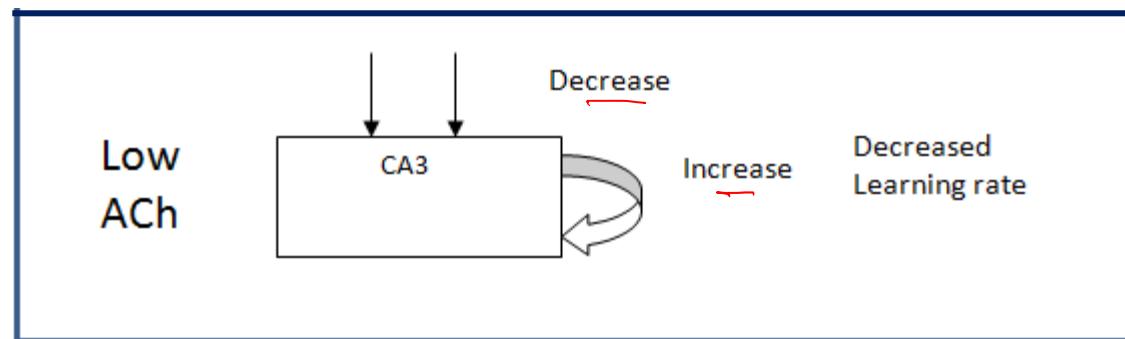
- Volunteers were given a dose of scopolamine and were asked to memorize words.
 - They had little or no memory of the information presented during the test
 - Some volunteers even forgot about the trial.
- Scopolamine before experiment:
 - Controls = 45 out of 128 words
 - Scopolamine subjects = 6 out of 128 words

- When subjects were given scopolamine after the words were presented, no impairment in performance.
- Specifically, there is also evidence that Ach controls LTP in the recurrent connections of CA3.
- Michael Hasselmo and coworkers have suggested that Ach satisfies the criteria necessary to switch between storage and retrieval as follows:

- At high levels of ACh,
 - Input is strengthened $\uparrow A$
 - Recurrent connections of CA3 are weakened $\downarrow B$
 - Increased plasticity in recurrent connections of CA3 $\uparrow C$



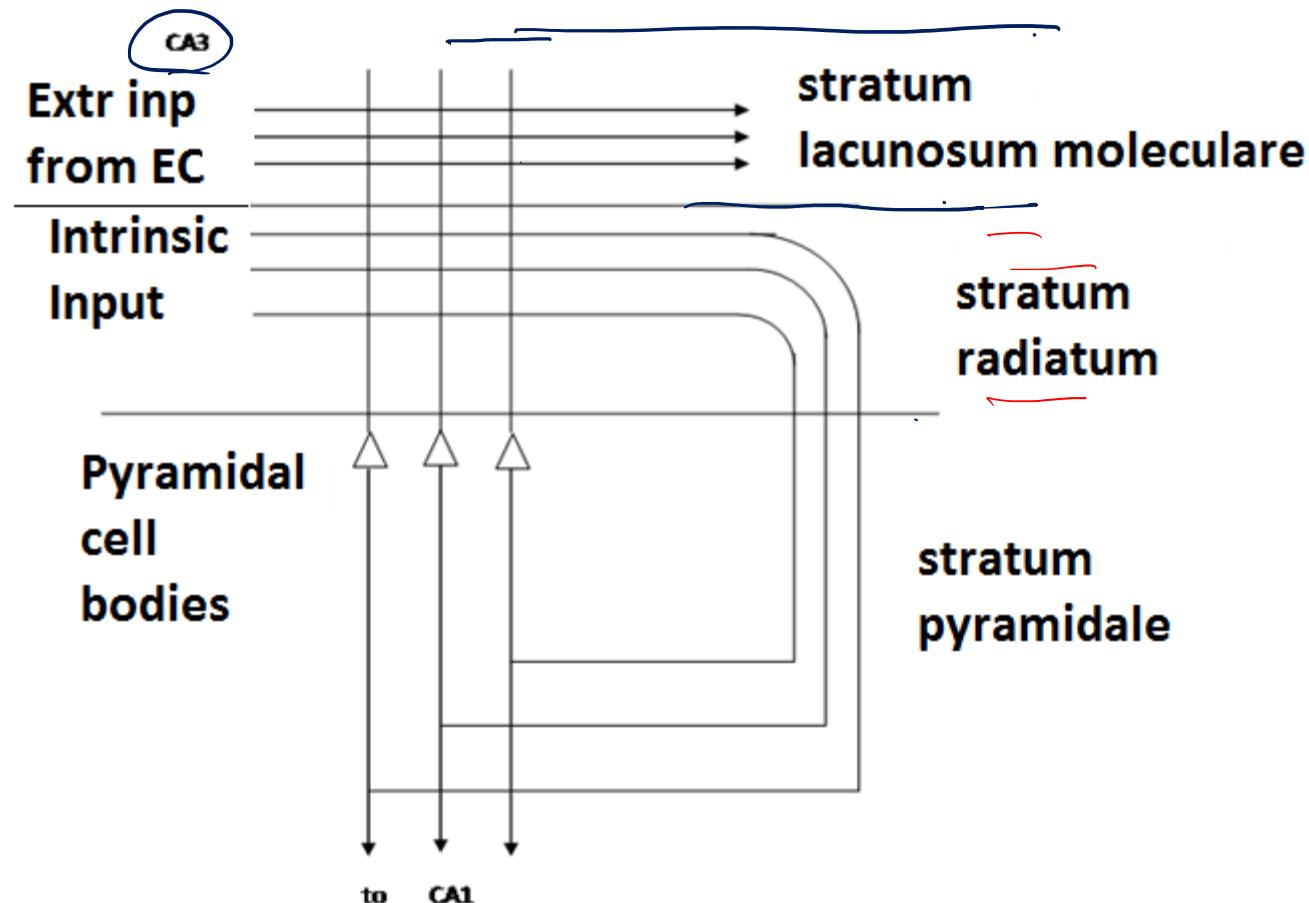
- At low levels of ACh,
 - Input is weakened $\downarrow A$
 - Recurrent connections of CA3 are strengthened $\uparrow B$
 - Decreased plasticity in recurrent connections of CA3 $\downarrow C$



Differential action of ACh in CA3

- Ach has different actions on different parts of CA3.
- Inputs to CA3 pyramidal neurons from different sources are anatomically segregated in CA3.
 - Inputs from EC to CA3 pyramidal neurons are located in a CA3 layer known as **stratum lacunosum-moleculare**.
 - Recurrent inputs from other CA3 pyramidal neurons are located in a CA3 layer known as **stratum radiatum**.

CA3 connectivity

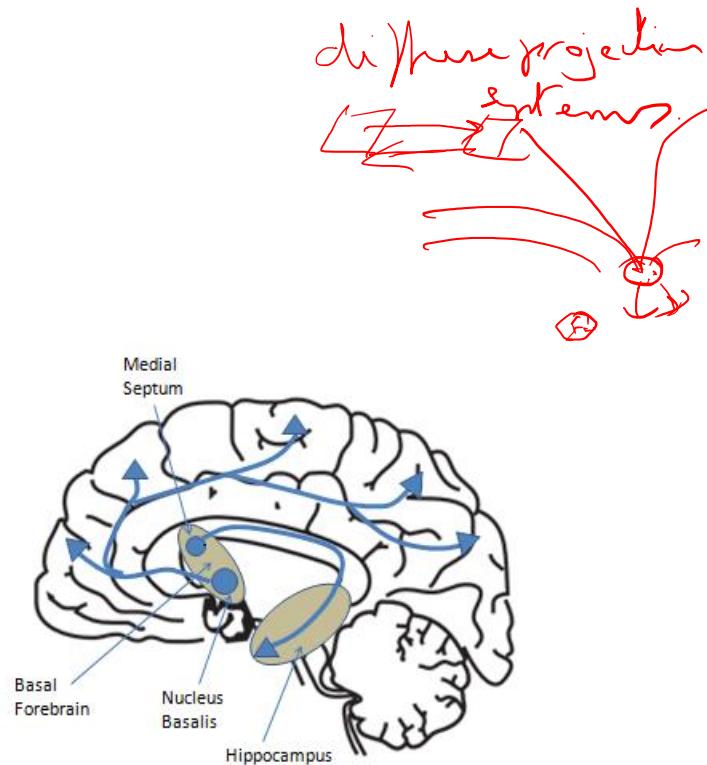


Action of ACh on CA3

- High Ach:
 - it suppresses the synapses in stratum radiatum much more than it does to synapses in stratum lacunosum-moleculare.
- High ACh suppresses recurrent connections more than inputs coming from EC.
- High Ach also increases LTP in CA3 recurrent connections.

What modulates the modulator?

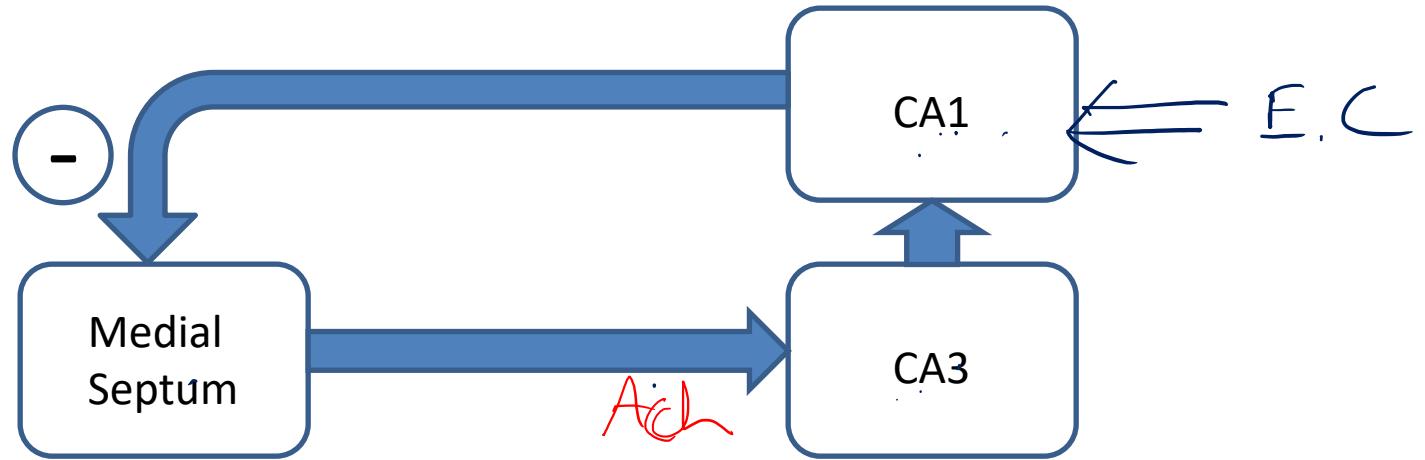
- What controls ACh release to HC?
- ACh is released by two neuronal clusters located in the basal forebrain.
- **Nucleus Basalis**: projects to widespread cortical targets.
- **Medial septum**: projects to HC.



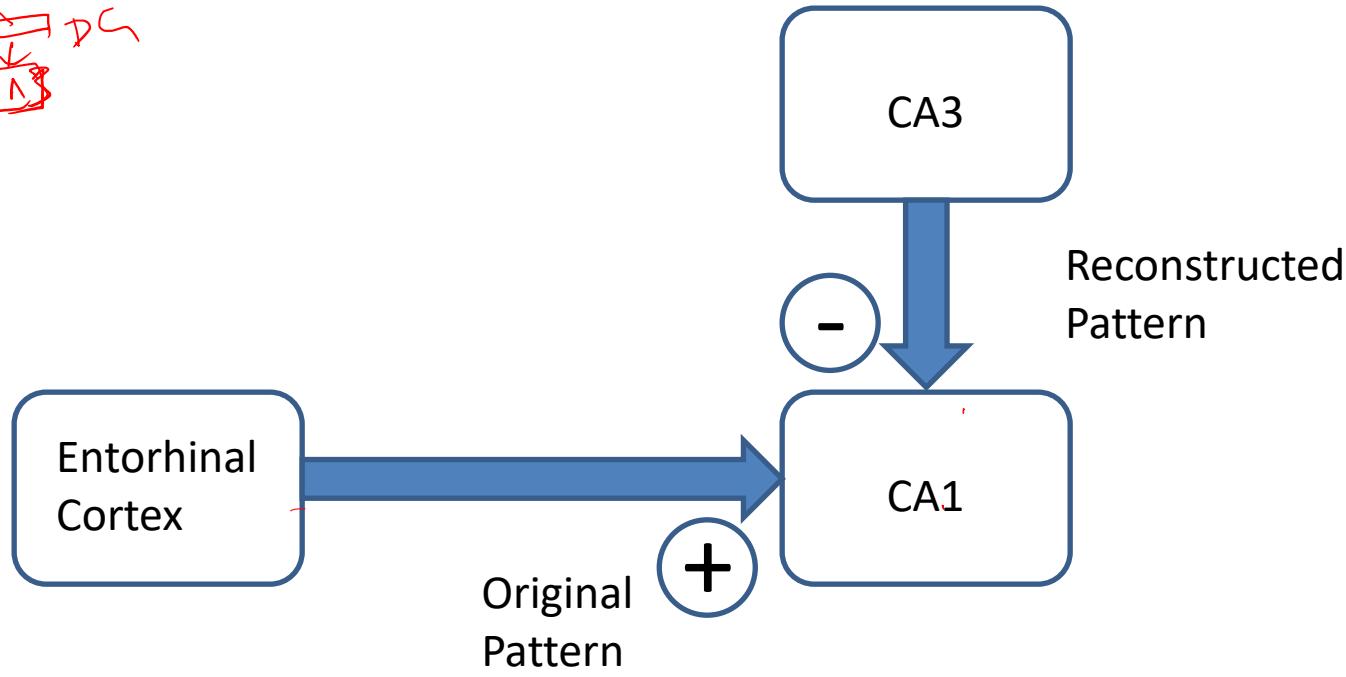
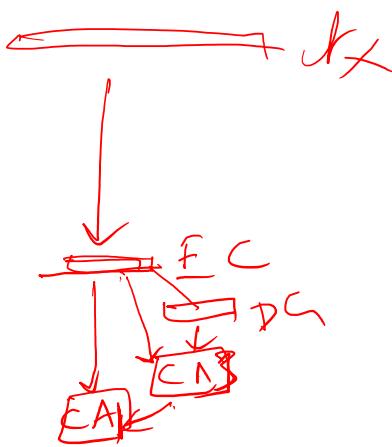
The Septohippocampal loop

- When this branch is stimulated it decreases the neural activity in medial septum.
- Thus, the loop from medial septum to HC and backwards acts as a self-regulating system for HC.
- Particularly it is known that the feedback exists from CA1 field of HC.

Septohippocampal Loop



Hasselmo and colleagues proposed that the above loop is capable of making sure that ACh is released just when it is required.



If the Original and Reconstruction match,
 Do NOT activate Medial Septum → no Ach release

If the Original and Reconstruction do not match,
 Activate Medial Septum → Ach release → Store pattern in CA3

Summary

- Hippocampus as a short term memory store
- CA3 as some sort of a Hopfield network
- The two modes of operation: storage and retrieval
- The role of Ach as a neuromodulator
- The role of the septohippocampal network
- Does not address:
 - The issue of saliency as a criterion for storage ✓
 - The transfer from short term to long term storage

Reference

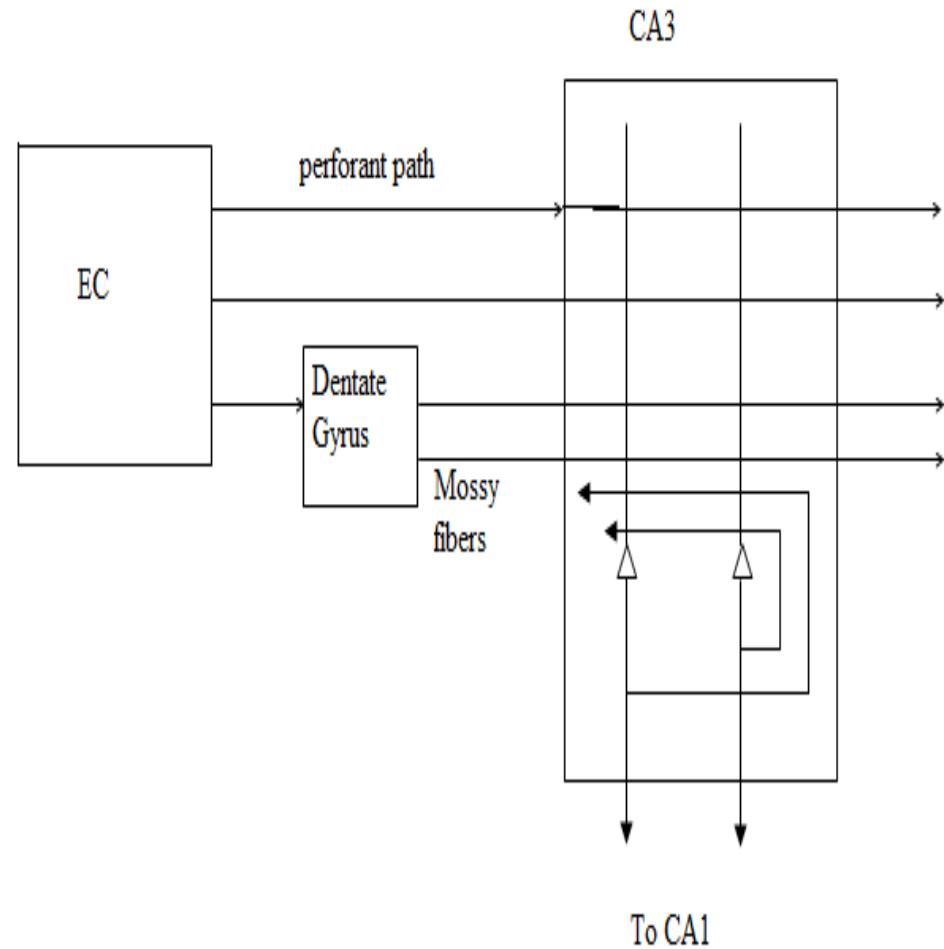
- *Gateway to Memory* by Gluck and Meyers, MIT Press.
- Demystifying the Brain by VSC, Springer.

- The last two areas project to Entorhinal cortex which is considered the gateway to HC.
- Information from EC propagates in that sequence via Dentate Gyrus, CA3, CA1 and subiculum before returning to EC. EC projects back to the neocortical targets via parahippocampal gyrus and perirhinal cortex.
- Thus HC receives compressed representation of information arising out of many sensory modalities, and applying the criteria of saliency which depends on the associated reward, sends the information back to the neocortex for long-term storage.

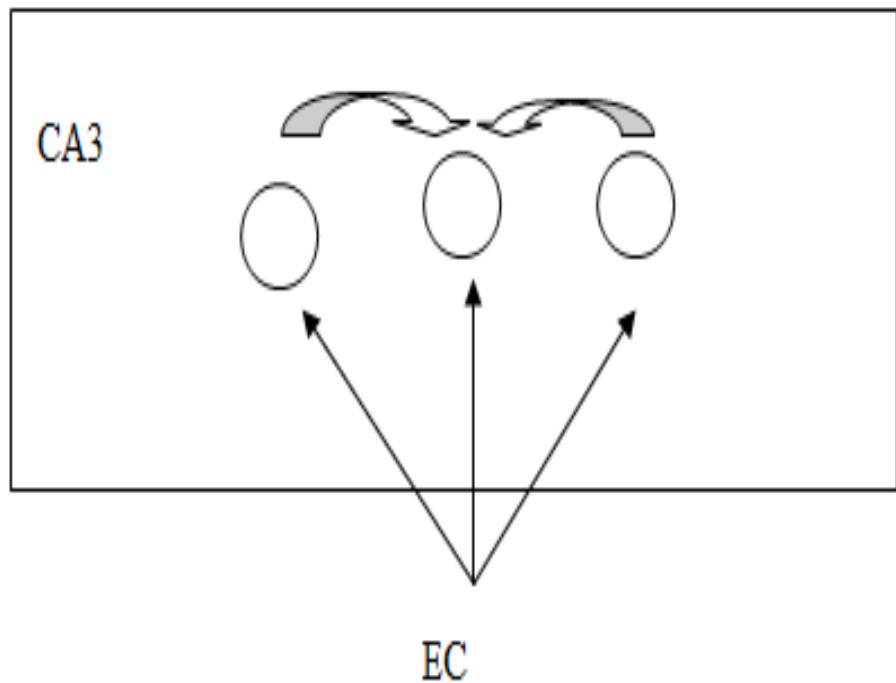
- The flow of information within the HC described above is rather simplistic.
- While it is true that information arising out of EC circulates through the HC and returns to EC, the path it follows is not a simple, singular loop as described above.
- Information from EC flows through multiple parallel pathways before returning to EC

- One branch of EC output into the HC goes to Dentate Gyrus, which in turn projects to CA3 via fibers known as mossy fibers.
- EC also projects directly to Dentate gyrus via the perforant pathway.
- The mossy fibers are known to have strong synapses, while the perforant pathway has a weaker influence on CA3.

- This difference in the relative strengths of the two afferents to CA3 seems to be significant, as we shall see shortly, to memory storage and recall operations of HC. CA3 neurons have extensive recurrent connections among themselves



- As mentioned above, CA3 neurons have extensive recurrent connections.
- In rat there are about 300,000 pyramidal neurons in CA3, with each of them receiving inputs from about 12,000 other CA3 pyramidal neurons. In contrast, each CA3 pyramidal neuron receives only 4000 inputs from EC.



- Effectively, each CA3 neuron receives inputs from about 4% of all CA3 pyramidal cells, a level of recurrence that is high compared to any other brain region.
- Furthermore, these recurrent connections are modifiable by LTP.
- Therefore, though the level of recurrence in CA3 is much lesser than that in the Hopfield network, it suggests the strong possibility of CA3 acting as some sort of a Hopfield network in which patterns are stored as attractors.

- We have seen that the memory capacity, P , of a fully-connected Hopfield network, in which the binary state (1/0) of each neuron is 1 or 0 with equal probability, equals $0.14 N$, where N is the number of neurons in the network.
- But it has been found that capacity is greater when sparseness conditions are imposed on the stored patterns.

- A pattern is considered to be sparse when instead of half the neurons being active (+1), only a small fraction, a , of neurons are active. There is evidence that only a small fraction of CA3 neurons are active at any given instant (evidence regarding this matter is reviewed in (Treves and Rolls, 1994)).

- Storage capacity of CA3 system in rat, which depends on sparseness, a , is estimated by Treves and Rolls (1994). Sparseness is defined as:

$$a = \frac{(E[r])^2}{E[r^2]}$$

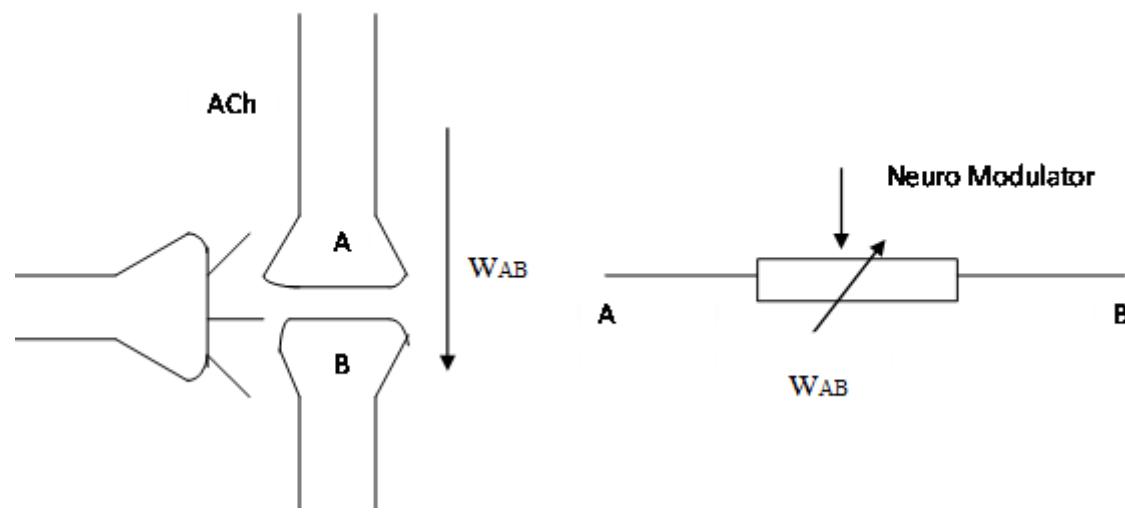
- Storage capacity, P, is estimated as,
- $P = \frac{C^{RC} k}{aln(1/a)}$
- C^{RC} - is the number of recurrent connections per CA3 neuron. (=12,000 in rat)
- k – is a parameter that depends on the neural activity distribution, connectivity pattern etc (Treves and Rolls 1994)
- a – sparseness, which is taken to be 0.02 based on experimental data
- Accordingly, P is calculated to be 36,000 patterns.

Storage and Retrieval in CA3:

- We have considered evidence that CA3 has features of an autoassociative network – high recurrence, sparseness and plastic synapses.
- But these features in themselves are not sufficient, since for the network to be useful as a memory, it must be operable in two modes – storage and retrieval.
- Very different conditions must prevail in the network when it operates in these two modes.

- Recall the two modes of operation of the Cohen-Grossberg model during storage and retrieval.
- During storage:
 - External input is strong
 - Recurrent connections are weak
 - Recurrent connections are plastic
- During retrieval:
 - External input is weak
 - Recurrent connections are strong
 - Recurrent connections are not plastic

- Let us consider if the above features are available in and around CA3.
- CA3 has two external inputs: one from EC by perforant pathway, and the other from EC via Dentate Gyrus by mossy fibers. There is evidence that perforant pathway inputs are weaker than mossy fibers.
- Strength and plasticity of CA3 recurrent connections: Acetylcholine (Ach) is a neuromodulator that is capable of controlling the strength and plasticity of recurrent connections in CA3.

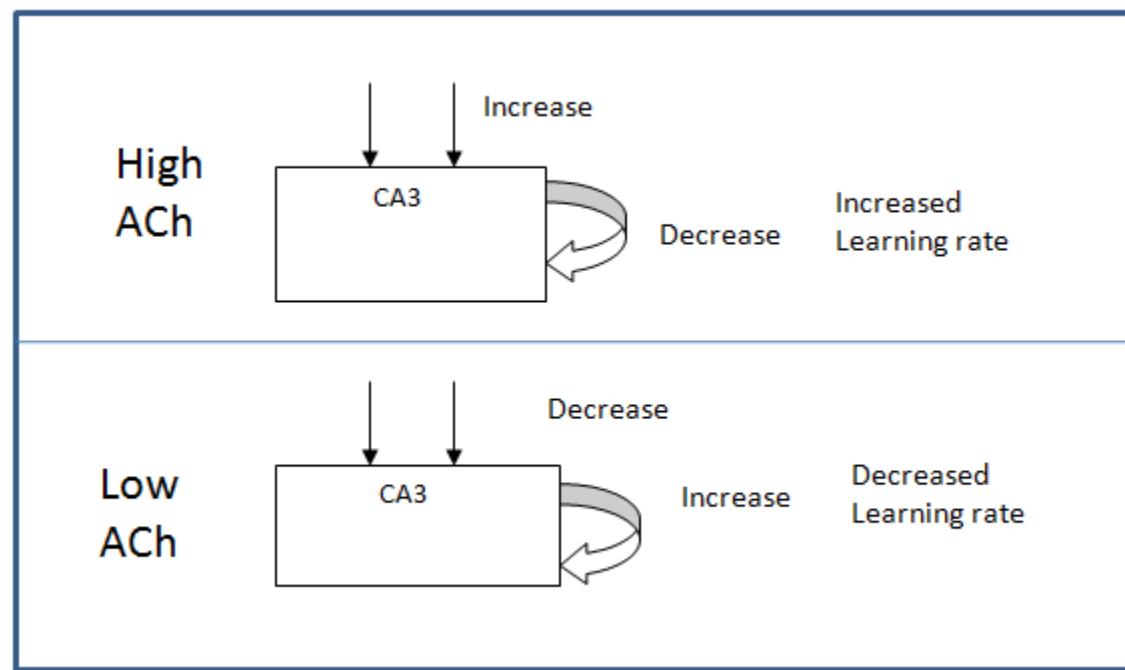


- A neuromodulator alters the strength of a synapse. Some common neuromodulators in the brain are:
- Acetylcholine, dopamine, norepinephrine and serotonin
- Blockage of ACh transmission in HC is known to interfere with memory storage.
- Scopolamine (Ach antagonist) → temporary amnesia

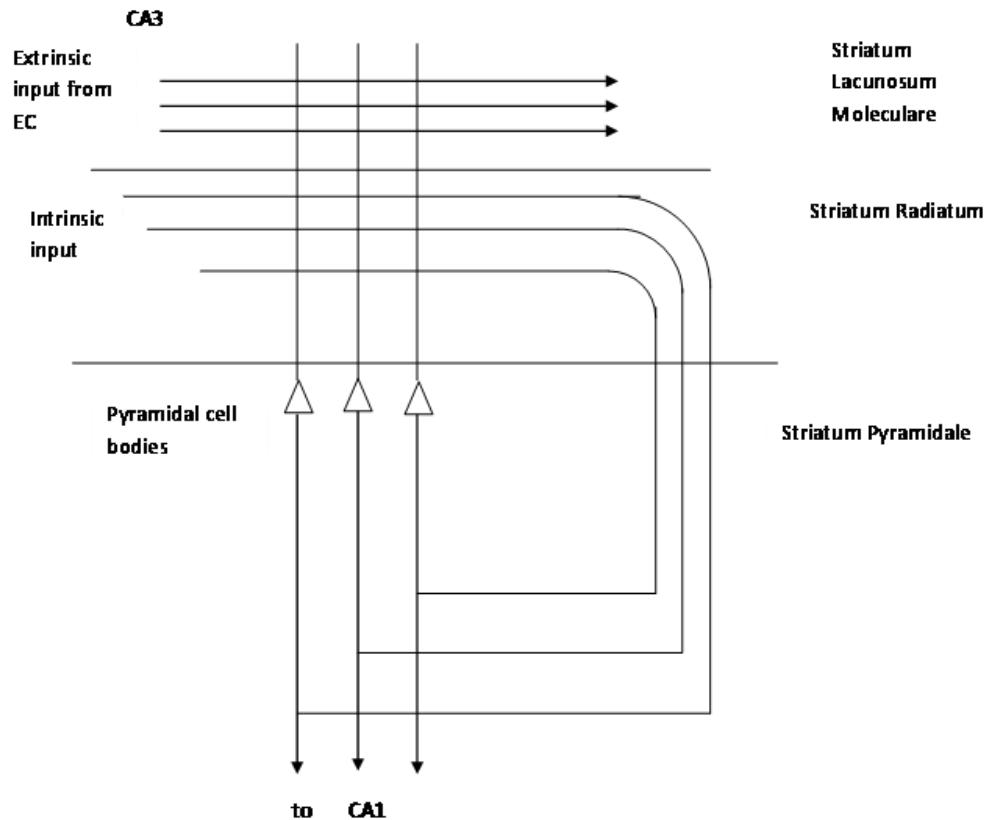
- Volunteers were given a dose of scopolamine and were asked to memorize some information.
- They had little or no memory of the information presented during the test
- Some volunteers even forgot about the trial.
- Scopolamine before experiment:
- Controls = 45 out of 128 words
- Scopolamine subjects = 6 out of 128 words

- When subjects were given scopolamine after the words were presented, no impairment in performance.
- Specifically, there is also evidence that Ach controls LTP in the recurrent connections of CA3.
- Michael Hasselmo and coworkers have suggested that Ach satisfies the criteria necessary to switch between storage and retrieval as follows:

- At high levels of Ach,
 - Input is strengthened
 - Recurrent connections of CA3 are weakened
 - Increased plasticity in recurrent connections of CA3
- At low levels of Ach,
 - Input is weakened
 - Recurrent connections of CA3 are strengthened
 - Decreased plasticity in recurrent connections of CA3

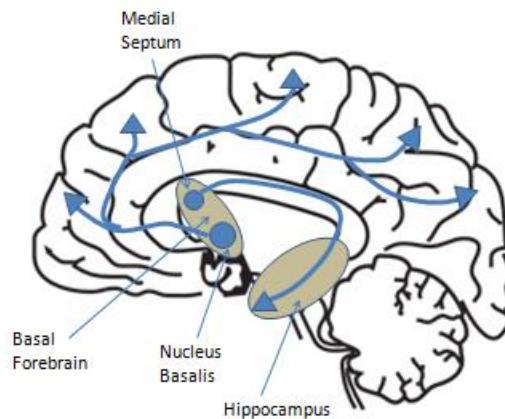


- Such complex differential action of Ach is possible because, Ach has different actions on different parts of the neuron.
- Furthermore, inputs to CA3 pyramidal neurons from different sources are anatomically segregated in CA3.
 - Inputs from EC to CA3 pyramidal neurons are located in a CA3 layer known as stratum lacunosum-moleculare.
 - Recurrent inputs from other CA3 pyramidal neurons are located in a CA3 layer known as stratum radiatum.



- When ACh or an agonist of ACh is applied to CA3, it suppresses the synapses in stratum radiatum much more than it does to synapses in stratum lacunosum-moleculare.
- Thus at higher concentrations ACh suppresses recurrent connections more than inputs coming from EC. At the same time increased ACh increases LTP in CA3 recurrent connections.

- What controls ACh release to HC?
- ACh is released by two neuronal clusters located in the basal forebrain.
- One cluster called the nucleus basalis, projects ACh to widespread cortical targets. The other cluster called medial septum projects to HC.



- When this branch is stimulated it decreases the neural activity in medial septum.
- Thus, the loop from medial septum to HC and backwards acts as a self-regulating system for HC.
- Particularly it is known that the feedback exists from CA1 field of HC.

- Hasselmo and colleagues proposed that the above loop is capable of making sure that ACh is released just when it is required.
- Basically ACh must be released when the input to HC, arriving at EC, needs to be stored.
- If the input is already stored, ACh release must be blocked.

- CA1 receives direct inputs from EC and indirectly via CA3. Hasselmo and coworkers proposed that CA1 acts as a comparator, that compares the original pattern in EC, and its reconstructed version from CA3.

- If the two copies are the same, the pattern must have already been stored in CA3. Then CA1 sends a signal to medial septum inhibiting it, and blocking further release of ACh.
- If the two copies are not the same, CA1 inhibition of medial septum ceases, thereby causing of release of ACh to CA3, which in turn induces pattern storage.

Competitive Learning

Like Hebbian learning, Competitive learning is an example of unsupervised learning.

- We have seen two ways in which Hebbian learning can be used:
 1. The Hopfield network, in which weights trained by Hebbian learning, serves as an associative memory. In such a memory, patterns can be stored and retrieved by cueing the network with an incomplete pattern. Thus a whole neighborhood of incomplete patterns (the basin of the attractor) is mapped onto the stored pattern. In that sense a Hopfield network partitions the input space, wherein each stored pattern represents a partition.

2. When Hebbian learning was used in a feedforward network, the weight vector converged to the eigenvector corresponding to the highest eigenvalue of the autocorrelation matrix of the input data. When Sanger's or Oja's rule, extensions of the basic Hebb's rule, were used, the network was able to extract the top K eigenvectors of the autocorrelation matrix of the input data.

- Thus in case 1, Hebbian learning performs a kind of clustering and in case 2, Hebbian learning extracts the principal components of the input data.
- Competitive learning is a form of unsupervised learning which performs clustering over the input data.

- In a competitive learning network with n-output neurons, each output neuron is associated with a cluster.
- When a data point from a cluster is presented to the network, only the neuron corresponding to that cluster responds, while all other neurons remain silent.
- The single neuron that responds is often called a “winner” and therefore a competitive learning network of the kind just described is also known as a “winner-take-all” network.

- It is easiest to introduce CL mechanism as a slight variation of Hebb's rule.
- For a single neuron, whose input-output relationship is given as, $y=w.x$
- Weight update by Hebb's rule is defined as,

$$\Delta w = \eta yx$$

- We have seen that as per the above rule, which simply aims to maximizing squared output, y^2 , the weight vector is rotated towards the input pattern.
- Since the weight vector is normalized, it is only allowed to rotate.
- The response of the neuron, y , increases as the weight vector aligns itself more and more in the direction of the input vector.

- Now consider a data set constrained such that all data exists on the unit sphere, $\|x\|=1$.
- Now that lengths are constrained, angular deviation and distance are uniquely related, at least within the hemisphere that is centered around the data point, x .
- Response is lesser when w is farther from x , and is the highest when $w = x$

- It is possible to present a different neuron model, for which the response $y = f(x; w)$, is the highest for $w = x$, and decreases with increasing distance between w and x .
- A simple example of such a function is the Gaussian.

$$y = \exp(-\|x - w\|^2 / \sigma^2)$$

- Just as in case of Hebbian learning, weight update simply increases the squared output, y^2 ; in case of the above “Gaussian neuron” too, we can formulate a weight update rule that increases the neuron’s response.
- Thus, the weight vector, w , is changed in the direction of gradient of y , w.r.t. w .

$$\Delta w = \eta \nabla_w y = 2\eta y(x - w) / \sigma^2$$

- Since y is always positive, it can be considered as a part of the learning rate h . Thus, we have the following simpler learning rule:

$$\Delta w = \eta(x - w)$$

- We thus have a ‘Gaussian’ neuron (fig. ***) whose response is given by eqn. (**). The neuron shows high response for inputs, x , close to the weight vector, w .
- Neurons which respond selectively to a small part of the input space are said to have ‘tuned responses.’

- Such neurons are known to exist in various sensory systems:
 - Neurons in the primary visual cortex that respond to a line of a given orientation presented in their receptive fields (Hubel D.H. and Wiesel T.N. (1974) Sequence regularity and geometry of orientation columns in the monkey striate cortex. *J. Comp. Neurol.*, 158:267-294)
 - Neurons in the auditory cortex that respond narrowly to single tones (Philips D et al (1995) Factors shaping the tone level sensitivity of single neurons in posterior field of cat auditory cortex.)

Training: Single neuron case:

- Let us consider what happens when we train a single Gaussian neuron using the learning rule. What does w converge to?
- Let x be drawn from a distribution, $p(x)$.
- If we iterate the learning rule at convergence we have,

$$\langle \Delta w \rangle = 0 = \int (x - w) p(x) dx$$

$$\int wp(x)dx = \int xp(x)dx$$

or

$$w = \int xp(x)dx$$

- Therefore, w is the mean of the data with probability density, p(x).
- Now let us consider the result of training with multiple neurons. Let us begin with 2 neurons.

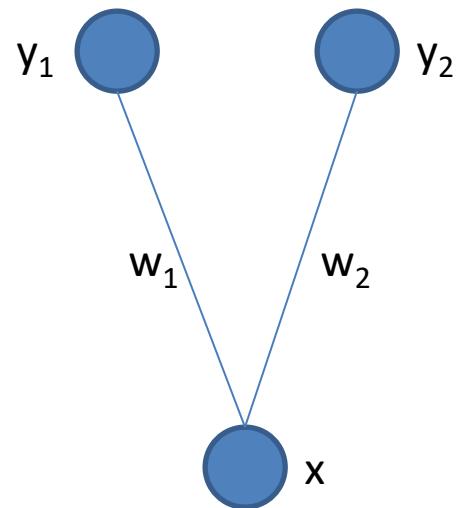
Two-neuron case:

- Let us consider two neurons with response functions given as,

$$y_1 = \exp(-\|x - w_1\|^2 / \sigma^2)$$

$$y_2 = \exp(-\|x - w_2\|^2 / \sigma^2)$$

where w_1 and w_2 are the weight vectors of the two neurons respectively.



- A straightforward extension of training to 2-neuron case would be to train both the weight vectors using the training rule as follows,

$$\Delta w_1 = \eta(x - w_1)$$

$$\Delta w_2 = \eta(x - w_2)$$

- But that gives us a trivial result since both w_1 and w_2 converge to the same point - the mean of the input data.

- To obtain a more interesting outcome, let us introduce the element of *competition* into learning.
- Since the Gaussian neurons only respond to a specific part of the input space (defined by the centroid w and width s), let us train the neurons such they respond to non-overlapping portions of the input space.
- Thus either neuron “specializes” over a certain part of the input space.

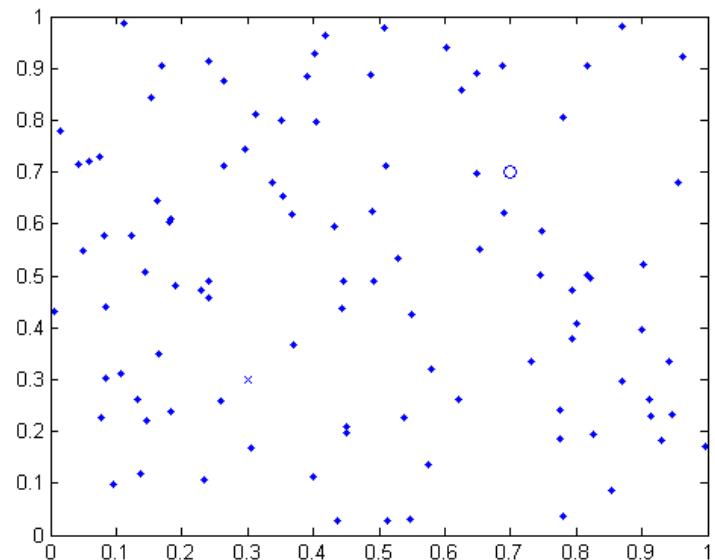
- This intuitive idea can be formalized in the following learning rule,

Learning rule:

- For a given x ,
 - If $y_1 > y_2$, $\Delta w_1 = \eta(x - w_1)$
 - Else $\Delta w_2 = \eta(x - w_2)$

- In the above training process, the two neurons are said to “compete” with each other to “win” over a certain input vector. Thus for a given input, the neuron with higher response is said to be the “winner.”
- The above learning rule is known as competitive learning.
- Let us consider the consequences of the above learning rule using a simple example.

- Consider a data set, S , consisting of points uniformly distributed over the unit square. Let the weights w_1 and w_2 be initialized to $(0.5, 0.1)$ and $(0.5, 0.8)$.
- At the end of training, neuron 1 is the winner for data from the upper half of the unit square, while neuron 2 is the winner for the lower half.



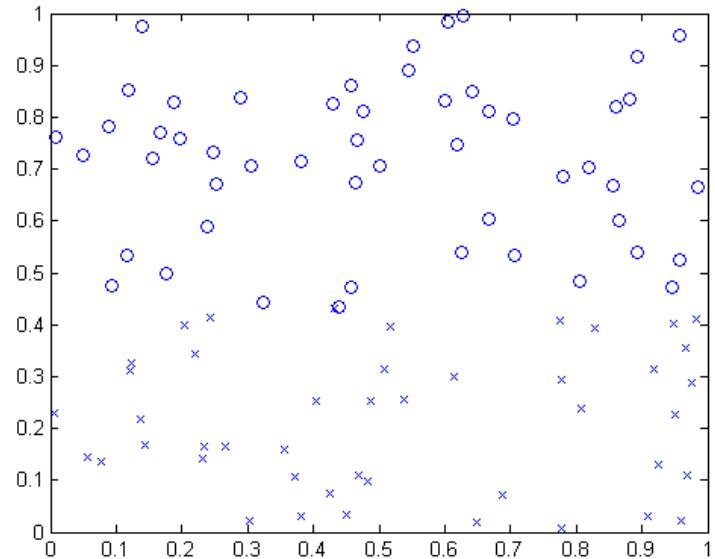
$w_1:(0.5, 0.1)$
 $w_2:(0.5, 0.8)$

- Thus the two neurons partition the input space. Two important lessons can be drawn from this exercise.

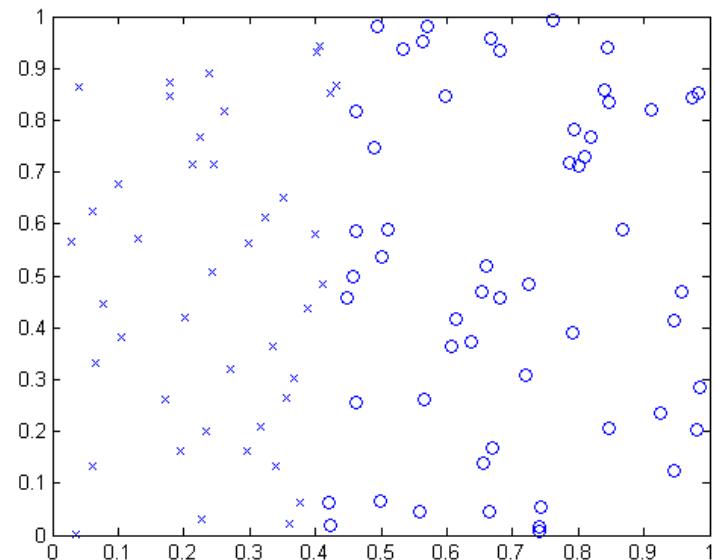
1. The border between the two partitions is the perpendicular bisector of the line joining w_1 and w_2 . The responses of the two neurons are the same on the border between the two partitions. Or,

$$\exp(-\|x - w_1\|^2 / \sigma^2) = \exp(-\|x - w_2\|^2 / \sigma^2)$$

$$\|x - w_1\|^2 = \|x - w_2\|^2$$



2. The final partitions obtained are sensitive to the initial conditions of w_1 and w_2 . Consider the final weights obtained with the following initial conditions $w_1 = (0.1, 0.5)$ and $w_2 = (0.8, 0.5)$. In this case, neuron 1 wins over the left half of the input space, while neuron 2 over the right half.



N-neuron case

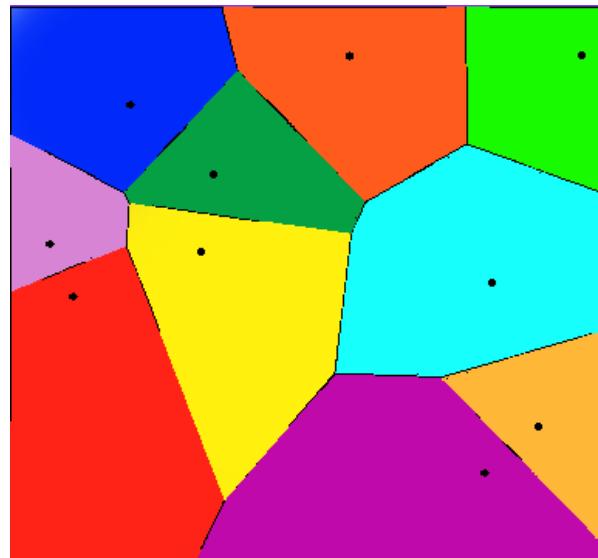
- In a network with ‘n’ neurons, with responses given as,

$$y_i = \exp(-\|x - w_i\|^2 / \sigma^2)$$

- the winner i^* is the one for which $y_{i^*} > y_i \quad \forall i \neq i^*$
- Only the weight of the winner is updated as usual:

$$\Delta w_{i^*} = \eta(x - w_{i^*})$$

- In this case, the network partitions the input space into n partitions, where each partition is bounded by hyperplanes.
- This partitioning is known as Voronoi tessellation.
- Thus competitive learning gives a way of partitioning a given data set which brings us to the problem of data clustering.



10 shops in a flat city and their Voronoi cells (euclidean distance).

Data Clustering

- The problem of data clustering consists of associating every data point, x , in a data set, S , to a cluster, c . This mapping from x to c is often done so as to optimize certain constraints.
- Two commonly used criteria for clustering are:
 - points within a cluster are close to each other.
 - points in different clusters are not so similar/close to each other.

- Data clustering is a vast area and cannot be discussed at any length here.
- Our present aim is only to show that a certain kind of clustering, known as K-means clustering, is closely related to competitive learning.

- In K-means clustering, the clusters are defined by a representative vector or a mean vector, w_i .
- A data point x_p , is associated to a mean vector, w_i , only if w_i is the nearest mean vector.
- We presently use Euclidean distance as the distance measure, though more sophisticated distance measures are also used.

- Criterion (i) can be satisfied mathematically in this fashion.
- Consider the cost function,

$$E = \sum_p \| x_p - w_{i(p)} \|^2$$

x_p – p'th data point

$w_{i(p)}$ – mean vector nearest to x_p .

- The mean vectors, w_i , can be updated so as to minimize E , by performing gradient descent on E , as follows:

$$\Delta w_{i(p)} = \eta(x_p - w_{i(p)})$$

- The above update can also expressed as follows. If w_i is the nearest mean vector (the “winner”) to x_p , update w_i as,

$$\Delta w_i = \eta(x_p - w_i)$$

- The update rule is known as K-means clustering.
- Note that it is also identical to the competitive learning rule, which was inspired by Hebbian learning.

Self-organizing Map

- Information is often represented spatially in the two-dimensional neuronal sheets in the brain, in both the cortex and subcortical structures.
- We have learnt about the somatosensory, motor and visual maps in the corresponding sensory cortices in the brain.

- A map, in its ordinary sense, denotes a two-dimensional representation of a real-world domain, such that nearby points in the domain are mapped onto nearby points in the map.
- Due to this “adjacency-preserving” property, these maps are also called topographic maps.

- Self-organizing maps (SOM) are models of the topographic maps of the brain, first proposed by Teovo Kohonen.
- The SOM model can be presented as an extension of the competitive learning model described in the previous section.

- It is constructed by adding a biologically-relevant feature that is not originally present in the competitive learning network.
- A key property of the SOM is that nearby or similar inputs activate nearby neurons in the map. The competitive learning network does not have this property.

- Consider a hypothetical competitive learning network with 3 output neurons.
- The input space is two-dimensional.
- The weight vectors w_1 , w_2 , w_3 lie on a line as shown with w_1 in between w_2 and w_3 .

- Note that such an arrangement is possible since there is no relation between the spatial position of the weight vectors and their indices.
- Now consider an input vector, x , that is continuously displaced from x_2 on the left to x_3 on the right extreme.
- The output neurons are activated in the following order: 2, 1, 3.

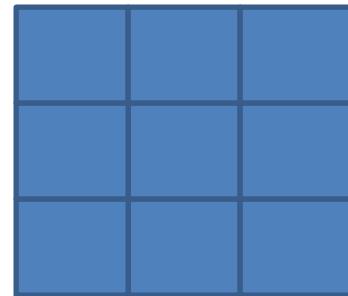
- If we imagine that the three output neurons are on a 1-dimensional grid, and their indices represent cardinal spatial positions on the grid, we now have a violation of the adjacency principle.
- Continuous changes in the input result in big jumps in the location of the neuron that is activated.
- Nearby inputs do not activate nearby neurons in the output layer.

- In order to achieve the adjacency principle, the competitive learning must be altered. In the original learning rule, each weight vector moves independently.
- There is nothing that constrains them to be ordered in any fashion.
- The essence of the modification proposed in the SOM model, is a mechanism that ensures that the weight vectors remain spatially ordered, while they also move towards the data points that activate them maximally.

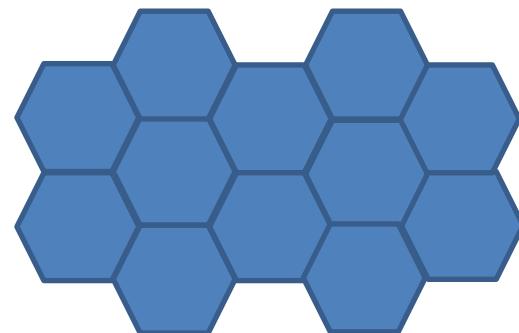
Learning rule for SOM

- Unlike a competitive learning network, which consists of a single row of output neurons, a SOM consists of a m-dimensional grid of neurons.
- Usually two-dimensional SOMs are studied since SOMs were originally inspired by the two-dimensional maps in the brain.

- The topology of the grid is usually rectangular, though sometimes hexagonal topologies are also considered.



Rectangular Topology



Hexagonal Topology

- As in the competitive learning network, the input, x , is presented to every neuron in the SOM.
- In a SOM with rectangular topology, the response of a neuron at location (i,j) may be expressed as,

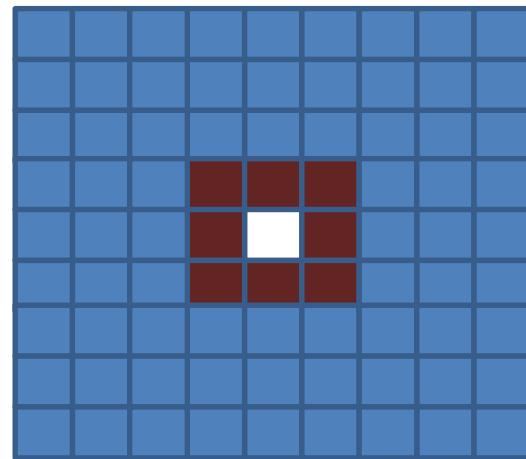
$$y_{ij} = \exp(-\|x - w_{ij}\|^2 / \sigma^2)$$

Where w_{ij} is the weight vector that connects the input vector, x , to the neuron at (i,j)

- The weights are initialized either randomly, or by drawing from the input data set itself.
- Training loop proceeds as follows:
- Loop over input data, x :
 1. Present x to all neurons in the SOM and calculate outputs using eqn. (*)
 2. Find the winner (i^*, j^*) such that,
$$y_{i^*j^*} > y_{ij} \quad \forall (i, j) \neq (i^*, j^*)$$

3.

- As in the case of competitive learning, the weight vector of the winner is moved towards the input, x .
- But in addition, neurons close to the winner in the SOM are also moved towards the input, x , but with a lesser learning rate.
- Neurons that are nearby in the SOM are defined by a neighborhood N



For the neuron in white(center) the neurons in red represent the neighborhood if we consider the neighborhood radius to be 1

- Weights corresponding to all the neurons in the neighborhood, N , of the winner, are moved towards the input as follows:

$$\Delta w_{i(p)} = \eta(x_p - w_{i(p)})$$

$$\Delta w_{ij} = \eta \Lambda(i, j; i^*, j^*)(x - w_{ij})$$

- Where, $(i, j) \in N$ and

$$\Lambda(i, j; i^*, j^*) = \exp(-(i - i^*)^2 - (j - j^*)^2)$$

- Note that is the highest for $(i,j)=(i^*,j^*)$ and decreases gradually with increasing distance between (i,j) and (i^*, j^*) .
 - Neighborhood size is large in the early stages, and is decreased gradually as training progresses.
4. Loop over data until all weights converge.
- End loop

- Let us consider the stages in SOM training with the help of an elementary example.
- Let us train a 1-dimensional SOM consisting of 20 neurons on 1-dimensional input data drawn from a Gaussian distribution of mean 0 and standard deviation 1.
- The weights, $(w_1, \dots w_i, \dots w_{20})$ also 1-dimensional, are randomly drawn from the data set. Note that there is no ordering among the initial weights.
- The ordering is expected to emerge out of SOM training.

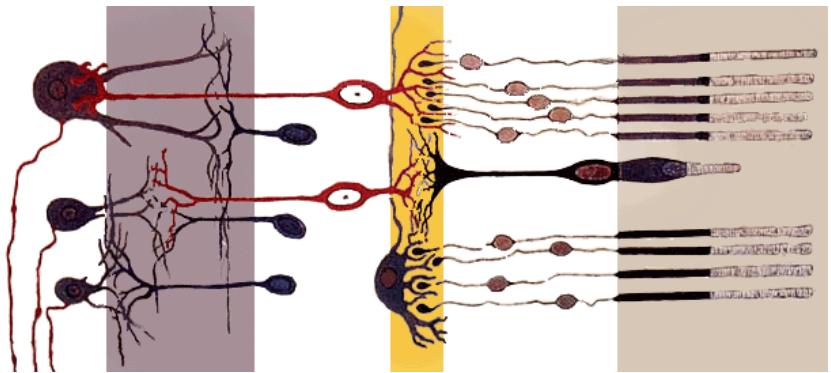
- Consider the distribution of weights in various stages of training.
 - 1) Ordering phase:
 - 2) Settling phase:

Applications of SOM model in neuroscience:

1. Modeling orientation-sensitive maps in the visual cortex:

- There are neurons in the primary visual cortex that respond specifically to oriented bars, lines or edges presented in their receptive fields. This response property suddenly emerges at the level of visual cortex and does not exist in lower stages of the visual system.

- Light that enters the eye through the pupil is converted into electrical signals by an array photoreceptors called rods and cones located in the retina.
- These signals are then propagated through two layers of neurons – bipolar layer and ganglion cell layer – both located in the retina

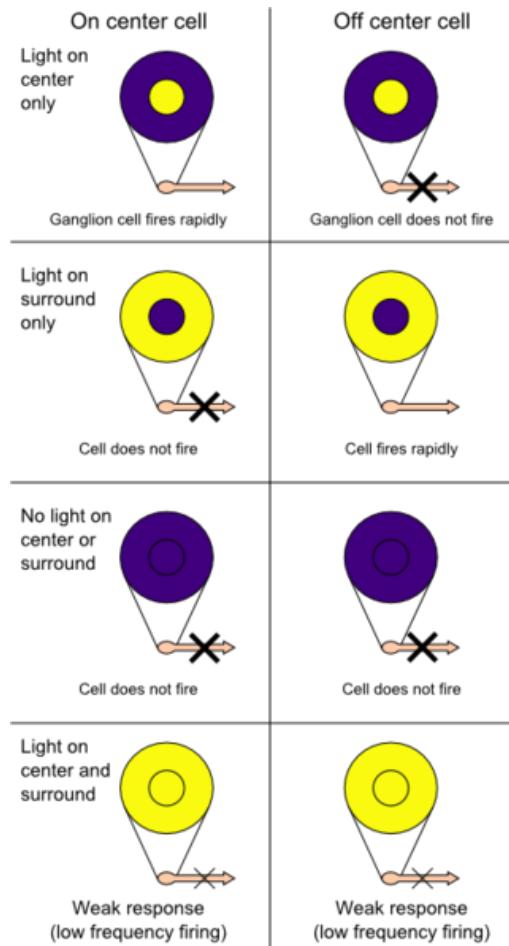


- The output of the ganglion cell layer projects to a nucleus known as the Lateral Geniculate Nucleus (LGN) located in the thalamus.
- LGN neurons in turn project to the primary visual cortex, the first cortical stopover of visual information entering the brain.
- A neuron in a given layer is connected to only a small “window” of neurons in the previous layer. This localized, or ‘pyramidal’ connectivity gives rise to the concept of a Receptive Field.

- Each neuron is able to receive information only from a small part of the external visual space.
- A photoreceptor receives information only a small visual angle.
- Same is the case with a bipolar cell since it receives inputs from a small set of photoreceptors.
- We have the same situation with the ganglion cell which has a RF of size 1° . As you higher in the visual hierarchy, naturally, the RF size increases.

- In addition to have larger RFs, neurons in higher stages of visual hierarchy also respond to more complex patterns compared to neurons in lower layers.
- For example, bipolar cells respond simple dot-like features – a bright dot with a dark background, or a black dot with a white background.

- Neurons that respond to circularly symmetric, dot-like patterns are thought to have circularly symmetric, “center-surround” RFs
- Neurons that respond to a bright dot with a dark background have “ON-center, OFF-surround” RFs, while those that respond to a black dot have “OFF-center, ON-surround” RFs.



- Neurons in bipolar and ganglion layers of the retina, and those of the many layers of LGN are known to have center-surround type of RFs.
- But in the primary visual cortex have RFs which do not have this simple circular symmetry.
- An important class of such cells consists of neurons that respond to oriented lines, a feature that can be easily modeled by a SOM.

- Showing how orientation sensitivity arises in a network naturally by unsupervised learning is only part of the answer.
- The Linsker's model, which we encountered in the last chapter, achieves it.
- A more challenging question is: how is the orientation sensitivity spatially distributed in the cortex. If every neuron in the visual cortex is mapped onto an orientation, q , (varying between 0 and π), what is the shape of that map?

- Starting with the pioneering work of Huber and Wiesel who discovered orientation maps in the visual cortex, a lot of subsequent experimental work was directed towards study of orientation maps.
- A summary of some of the salient properties of orientation maps is as follows:
 - The maps of orientation sensitivity are highly repetitive
 - Orientation changes continuously as a function of cortical location except at isolated points.

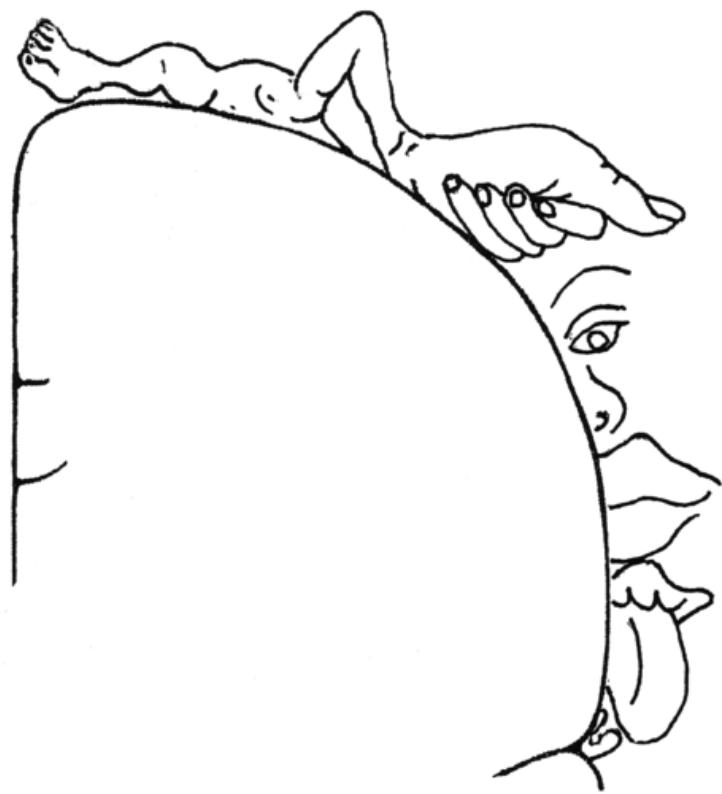
- These isolated points where orientation changes drastically are mathematical singularities. They are also known as ‘pinwheels’ in the jargon of visual science.
- Orientation changes by 180 deg around the pinwheels
- Both types of pinwheels appear in equal numbers
- There exist line-like regions (fractures), across which orientation preferences change rapidly with distance.

- Simple SOM-based models can be used to model all the above properties of orientation maps.
- A natural way to generate orientation maps with a SOM is to train it on images consisting of oriented lines.

- An even simpler model of orientation map is possible, by simplifying the training data. Orientation data is parameterized by a single parameter – orientation, q , - with periodicity of p . Instead of presenting whole images of oriented bars, we may present,
 $x = [\cos(2q), \sin(2q)]$, as input data.

A model of somatosensory map

- Located in the post-central gyrus, the somatosensory map is a tactile map of the entire body surface.
- Touching a point on the body surface activates a corresponding pool of neurons in the somatosensory map.

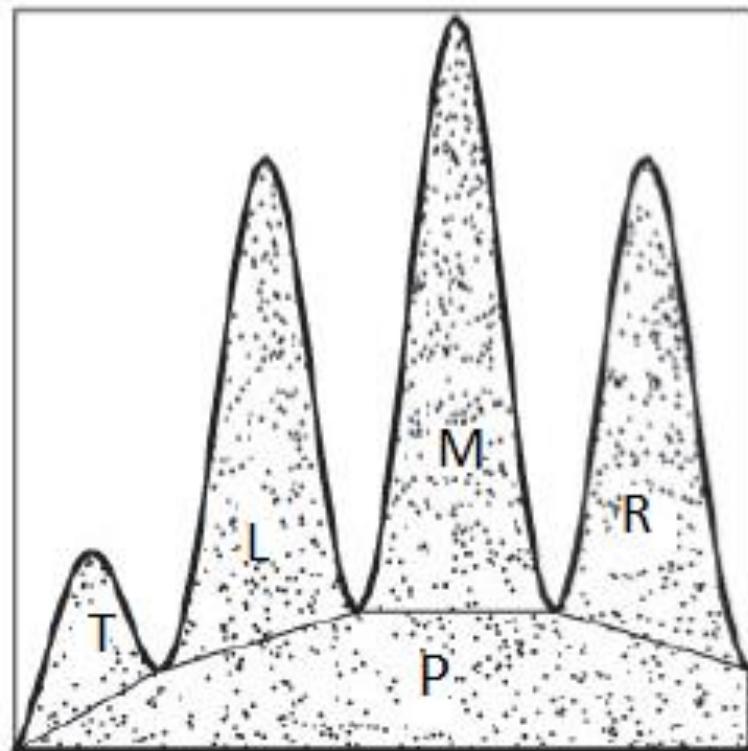


- Although the map satisfies for the most part, the adjacency principle, there are certain “fractures” in the map where the adjacency principle is violated.
- For example, note that the parts of the map that respond to the hand and the head are close to each other, though the hand and the head are not adjacent anatomically.

- The somatosensory map is not “drawn to scale.”
- There is a disproportionate allocation of cortical real-estate to the body surface.
- The map areas corresponding to the hands and mouth area are much bigger than the areas that process the abdomen.

- Thus more than the anatomical size of the body part that is serviced, the amount of tactile information that pours in from the said organ, seems to determine the cortical area allocated to the body part in the somatosensory map.

- Kaas and colleagues studied the dynamic nature of somatotopic map in an adult ape.
- Using electrophysiological recordings, they mapped the five fingers of one hand of the animal.
- The map reveals a nearly linear ordering of the five regions corresponding to the five fingers



- The middle finger of the animal was then amputated and the resulting changes in the map were observed.
- Several weeks after the amputation, the region of the map that earlier corresponded to middle finger, started responding to the adjacent fingers – index and ring fingers.

- Since the middle finger was missing, the region numbered 3 had nothing to respond to.
- Due to dynamic remapping and rewiring, neurons of area 3 now respond to adjacent fingers.
- Therefore, regions 2 and 4 now expand and encroach into what was region 3 earlier

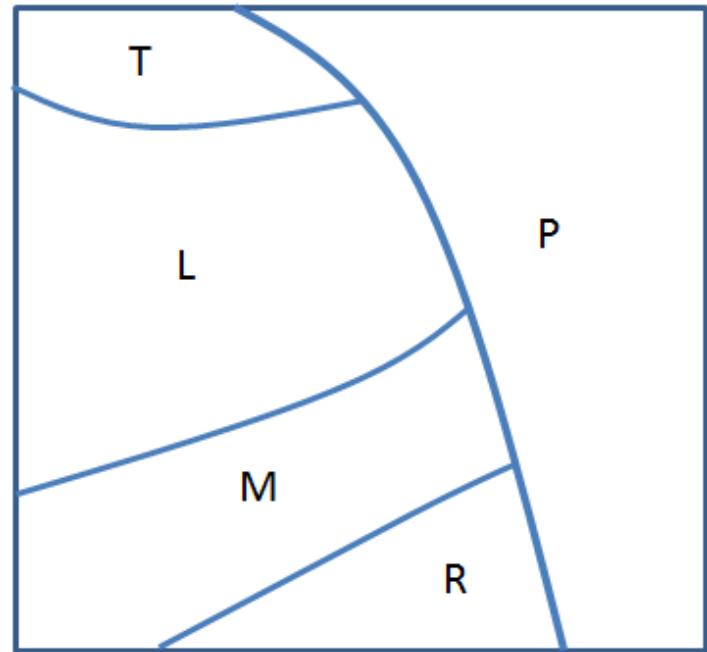
- Such dynamic reorganization of somatotopic map has been modeled by Ritter and Schulten used the SOM model to explain the dynamic remapping observed.
- The map model consists of a 30×30 array of neurons.
- Inputs: Points from the image of a “hand”, a two-dimensional area, parts of which designated as the five fingers

- Neurons are Gaussian neurons and therefore exhibit tuned responses.
- But the weights are not ordered. Adjacency principle is not preserved.
- Therefore nearby neurons do not respond to the same finger or to nearby fingers.

Map organization after 500 iterations:

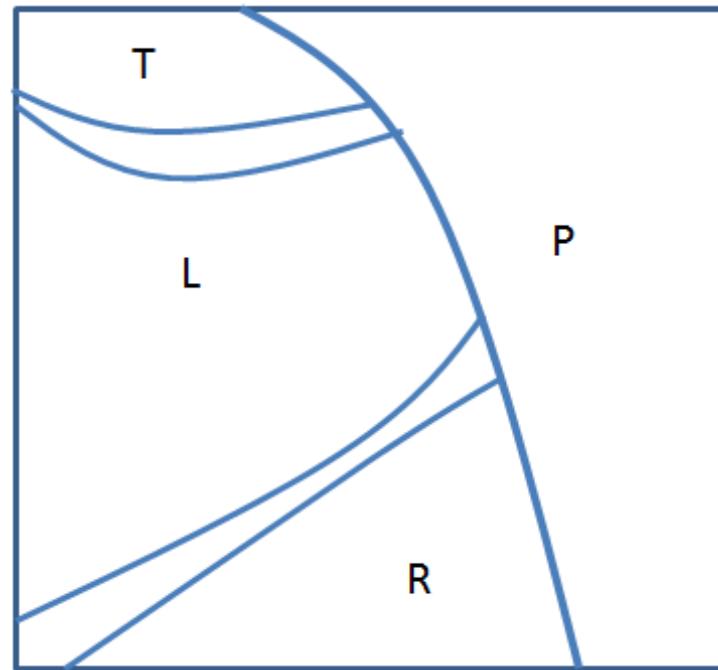
- A coarse map can be seen to have formed.
- Continuous stretches of neurons in the map now seem to respond to the same finger.
- However, it may be noted that some neurons at this stage do not respond to any point on the hand surface (dots in the map).

- A well-ordered map can be observed after 20000 iterations.
- The top right part of the map has neurons that respond to the palm.
- Below the palm region, there are four nearly parallel strips which correspond to the four fingers. Nearby neurons now respond to the same or adjacent finger quite consistently



Map organization after 20,000 iterations

- Simulating amputation: inputs from the middle finger are omitted, and the map training is continued.
- Map organization after another 50,000 iterations of training: the regions corresponding to the left and right fingers may be seen to encroach into what was earlier the “middle finger region.”



Inputs from middle finger omitted after
20000 iterations.

Map organization after 50,000 iterations

Linear neuron model: Hebbian learning

- The output of a linear perceptron can be given:

$$y = \sum_{i=1}^n w_i x_i$$

$$y = w \bullet x$$

- Applying Hebb's rule

$$\Delta w_i = \eta y x_i$$

$$\begin{aligned}\Delta w &= \eta y x \\ &= \eta (w^T x) x = \eta x (x^T w) = \eta (x x^T) w\end{aligned}$$

- For multiple patterns the learning rule is,

$$\Delta w = \eta \sum_p (x(p)x(p)^T)w$$

$$R = \sum_p x(p)x(p)^T$$

- R is the autocorrelation matrix of training data. It is:
 - symmetric,
 - positive semi-definite

- Proof:
 - Symmetric:
$$R_{ij} = \sum_p x_i(p)x_j(p)^T = R_{ji}$$
- Positive semi-definite:
 - Consider the quadratic form associated with R , where u is a non-zero real vector. $\frac{1}{2}u^T Ru$
$$\begin{aligned} \frac{1}{2}u^T Ru &= \frac{1}{2}u^T \left(\sum_p x(p)x(p)^T \right) u \\ &= \frac{1}{2} \left(\sum_p (u^T x(p))(x(p)^T u) \right) \\ &= \frac{1}{2} \sum_p (u^T x(p))^2 \geq 0 \end{aligned}$$

- In differential equation form
- Maximize

$$\Delta w = \eta R w$$

$$\dot{w} = R w$$

- Therefore, hebbian learning of a linear neuron
- Maximizing the quadratic form, E, of R

$$\begin{aligned} E(w) &= \frac{1}{2} w^T R w \\ &= \frac{1}{2} \left(\sum_p (w^T x(p))(x(p)^T w) \right) \\ &= \frac{1}{2} \sum_p (w^T x(p))^2 = \frac{1}{2} \sum_p (y(p))^2 \end{aligned}$$

- Therefore, Hebbian learning of a linear neuron
- Maximizing the quadratic form of R

$$\frac{1}{2} w^T R w$$

- Maximizing the average squared output of the neuron.
- We also note that if the data is ‘zero mean’ ($E[x] = 0$), Hebbian learning also maximizes output variance. Since the neuron is linear, for zero-mean input data, mean squared value of the output equals output variance. i.e.,

$$E[y^2] = \sigma_y^2$$

- But if R is positive semi-definite, $E(w)$ does not have a maximum.
- Therefore, E must be constrained. A simple constraint is to make w a unit norm vector. The unit norm constraint can be added as a cost to $E(w)$, yielding the new E' as follows:

$$E(w) = \frac{1}{2} w^T R w - \frac{1}{2} \lambda (\|w\|^2 - 1)$$

- Calculating the gradient, $\nabla_w E(w) = Rw - \lambda w = 0$

$$Rw = \lambda w$$

which is an eigenvalue equation.

- Therefore, when trained by Hebbian learning, the weight vector of a linear neuron converges to the eigenvectors of the autocorrelation matrix, R.
- But since a symmetric real matrix of size ‘ $n \times n$ ’ has n eigenvectors, it is not clear which of them w tends to.
- We will show that w tends to the eigenvector corresponding to the highest eigenvalue.

- Proof:

Let Q be a orthogonal, diagonalizing matrix Q such that,

$$Q^T R Q = \Lambda$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_i & 0 \\ 0 & 0 & \lambda_n \end{bmatrix}$$

$Q = [q_1 | \dots | q_i | \dots | q_n]$ where q_i are the eigenvectors of R.

- Now consider the linear transformation, $w = Qx$, and express $E(w)$ in terms of x , as follows,

$$\begin{aligned} E(w) &= \frac{1}{2}(Qx)^T R Q x \\ &= \frac{1}{2} x^T Q^T R Q x = \frac{1}{2} x^T \Lambda x = \frac{1}{2} \sum_{i=1}^n \lambda_i x_i^2 \end{aligned}$$

Since Q is also a rotational transformation, maximum of the new function $E(x)$ is the same as the maximum of the older function $E(w)$. Let us consider the maximum of $E'(x)$,

- Let the eigenvalues of R be ordered such that,

$$\lambda_1 \geq \dots \geq \lambda_i \geq \dots \lambda_n$$

$$E(x) = \frac{1}{2} \sum_{i=1}^n \lambda_i x_i^2$$

- We now impose the unit norm constraint on x as follows

$$= \frac{1}{2} \sum_{i=1}^n (\lambda_i x_i^2)$$

$$= \frac{1}{2} (\lambda_1 (1 - \sum_{i \neq 1} x_i^2) + \sum_{i \neq 1} \lambda_i x_i^2)$$

- The maximum of the above function can be found by solving the following differential equations,

$$\frac{dx_i}{dt} = (\lambda_i - \lambda_1) x_i$$

There are (n-1) such equations corresponding to (n-1) components, x_i $i = 2, \dots, n$.

λ_1 is the largest eigenvalue, in all the above differential equations, $x_i \rightarrow 0$, $i = 2, \dots, n$.

Since $\|x\| = 1$, the only remaining component $x_1 = 1$. Therefore the maximum of $E(x)$ occurs when,

$$x = [1 \ 0 \ 0 \ \dots \ 0].$$

Since $w = Qx$, we have, $w = q_1$.

- Thus the weight vector of the linear neuron of eqn

$$y = w \bullet x$$

converges to the eigenvector corresponding to the highest eigenvalue of E , when trained by Hebbian learning.

Oja's Rule

- Under the action of Hebbian learning, weight vector of a linear neuron converged to the first eigenvector of R only when the weights are normalized as $\|w\|=1$.
- But such a condition is artificial and not part of the Hebbian mechanism which is biologically motivated. Therefore, Oja (1982) proposed a modification of Hebbian mechanism in which the weight vector is automatically normalized without explicitly an explicit step like,
- $w \rightarrow w/\|w\|$
- The weight update according to Oja (1982) is as follows:

$$\Delta w_i = \eta y(x_i - yw_i)$$

- In vector form, the update rule can be written as,

$$\Delta w = \eta y(x - yw)$$

Let us prove that the above rule does the following:

- Maximizes $\frac{1}{2} w^T R w$
- $\|w\| = 1$

Consider the average update in w for the entire data set S , when the weight vector converges.

$$E[\Delta w] = \eta E[y(x - yw)] = 0$$

$$= \eta E[(yx - y^2 w)]$$

$$= \eta E[((w^T x)x - (w^T x)^2 w)]$$

$$= \eta[(E(xx^T)w - E(w^T(xx^T)w)w)]$$

$$= \eta[(Rw - (w^T R w)w)] = 0$$

- The last equation is the eigenvalue equation in R.

$$Rw - \lambda w = 0$$

- Thus w is an eigenvector of R , where $\lambda = w^T R w$

or, $\lambda = w^T (\lambda w)$

$$\|w\|=1.$$

Like Hebbian learning, an advantage of Oja's rule is that it is local: update for the i 'th component, w_i , of the weight vector, w , is dependent on quantities that are locally available at the presynaptic or postsynaptic ends of the synapse that is represented by w_i .

- Example: Long term potentiation in hippocampal neurons of brain

Principal Component Analysis (PCA)

- Before we proceed to prove an interesting result relating Hebbian learning and principal component analysis (PCA) we state a result from linear algebra.
- Spectral Theorem: If R is a real symmetric matrix, and Q is an orthogonal, diagonalizing matrix such that,
- Where

$$Q^T R Q = \Lambda$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_i & 0 \\ 0 & 0 & \lambda_n \end{bmatrix}$$

- Then $Q = [q_1 | \dots | q_i | \dots | q_n]$

$$R = \sum_{i=1}^n \lambda_i q_i q_i^T$$

- Proof:

Since $Q^T R Q = \Lambda$

$$\begin{aligned} R &= Q \Lambda Q^T \\ &= \sum_{i=1}^n \lambda_i q_i q_i^T \end{aligned}$$

To derive the last result, we used the following,

$$q_i q_j^T = \delta(i, j)$$

- Where $\delta(i, j)$ is the Kronecker delta, defined as,

$$\delta(i, j) = 1, \text{ if } (i = j)$$

$$= 1, \text{ otherwise}$$

- We have shown earlier that, Hebbian learning of a linear neuron
- Maximizing the quadratic form of $R = \frac{1}{2}w^T R w$
- Maximizing the average squared output of the neuron.
- We also note that if the data is ‘zero mean’ ($E[x] = 0$), Hebbian learning also maximizes output variance. Since the neuron is linear, for zero-mean input data, mean squared value of the output equals output variance. i.e., $E[y^2] = \sigma_y^2$

Hebbian learning for Data Compression

- A vector, x , of dimension, n , can be transformed into another vector, y , of dimension, m , ($m < n$), such that x can be reconstructed from y , with minimum reconstruction error.
- To enable such compression, we assume, for the moment, the following:
 - Hebbian learning extracts the eigenvector corresponding to the largest eigenvalue of the autocorrelation matrix, R .
 - But we assume that it is possible to extract all the eigenvectors of R , by some sort of an extension of Hebbian learning (*??*).
 - But for now we assume such an extension, and describe how data compression can be achieved by Hebbian learning.

Let x , be a data point drawn from a data set S . R is the autocorrelation matrix associated with S . Assume that the data is zero-mean ($E[x]=0$). Q is a matrix constructed out of the eigenvectors of R as follows,

$$Q = [q_1 | \dots | q_i | \dots | q_n]$$

Consider the following linear transformation,

$$y = Q^T x$$

Note that the components of y , are the projections of x onto the first m eigenvectors of R .

$$y_i = q_i^T x$$

Let us calculate the variance of y_i , which we will use shortly.

Since the data set, S , is zero-mean,

$E[x] = 0$. Therefore, from linearity of eqn. above, we have $E[y_i] = 0$. Therefore,

$$\sigma_{y_i}^2 = E[y_i^2] = E[(q_i^T x)(q_i^T x)]$$

$$= q_i^T E[(x^T x)] q_i$$

$$= q_i^T R q_i = q_i^T \lambda_i q_i = \lambda_i$$

Thus the variance of the i 'th component, y_i , is the corresponding eigenvalue.

'x' can be reconstructed from y, by simply inverting the transformation of eqn. $y = Q^T x$

Which is $x = Qy$

In the last equation, x can be expressed as a weighted sum eigenvectors as,

$$x = \sum_{i=1}^n q_i y_i$$

Eigenvectors are ordered such that the corresponding eigenvalues are in the descending order.

$$\lambda_1 \geq \dots \geq \lambda_i \geq \dots \lambda_n$$

Now consider a reconstruction of x, denoted by \hat{x} , produced by taking only a partial summation of the expression above.

$$\hat{x} = \sum_{i=1}^m q_i y_i$$

- Reconstruction error,

$$e = x - \hat{x} = \sum_{i=1}^n q_i y_i - \sum_{i=1}^m q_i y_i = \sum_{i=m+1}^n q_i y_i$$

- Root Mean Square (RMS) value of the reconstruction error is,

$$\begin{aligned}
 E[e^T e] &= E\left[\left(\sum_{i=m+1}^n q_i y_i\right)^T \left(\sum_{i=m+1}^n q_i y_i\right)\right] \\
 &= E\left[\left(\sum_{i=m+1}^n q_i^T q_i y_i^2\right)\right] = E\left[\left(\sum_{i=m+1}^n y_i^2\right)\right] = \\
 &= \sum_{i=m+1}^n E[y_i^2] = \sum_{i=m+1}^n \lambda_i
 \end{aligned}$$

Thus the reconstruction error is the sum of lower (corresponding to larger i) eigenvalues of R . As m increases, and approaches n , error reduces since there are fewer terms in the expansion of eqn above.

But error decreases also because the eigenvalues are sorted and lower eigenvalues are smaller in magnitude than higher ones (larger i).

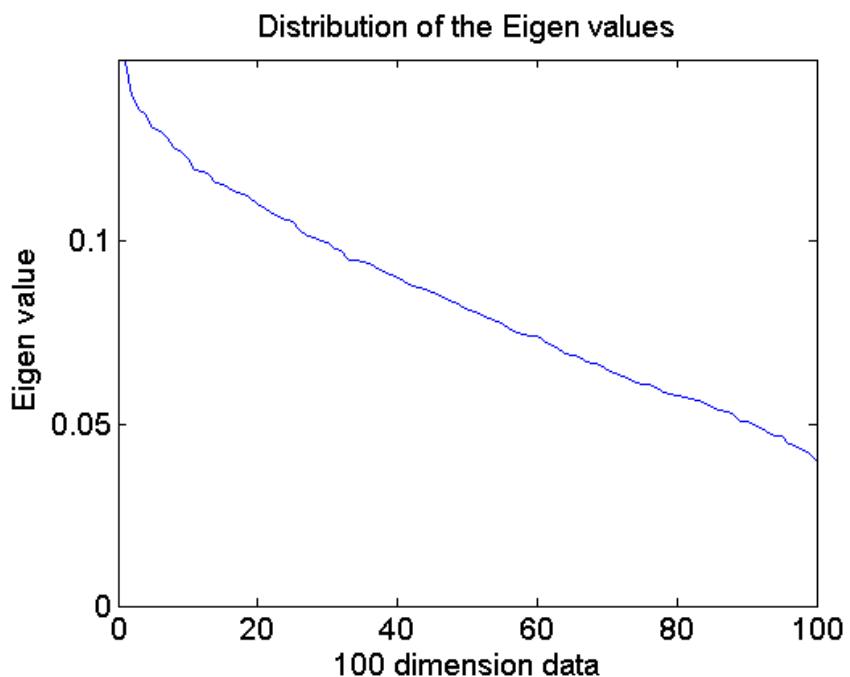
If ‘ m ’ is chosen such that the larger eigenvalues are included, x can be expressed in a compressed form, y , and reconstructed again, \hat{x} , with minimal loss.

Extending to ‘m’ principle components

- Hebb’s rule gives the weight vector that is the first eigenvector of the autocorrelation matrix R.
- Oja’s rule also essentially provided the same result, with the distinction that it achieved normalization naturally.
- ***It would be desirable to extend these results to the case of m-principal components.***
- Two such extensions are available
 - Sanger’s rule for extension (1989) and
 - Oja’s rule for extension(1989).

Example:

- Take random 100 dim data. Show the distribution of eigenvalues.
- Taken a random 100 dim data distributed between 0 and 1. The Eigen value is distributed is shown in the figure:



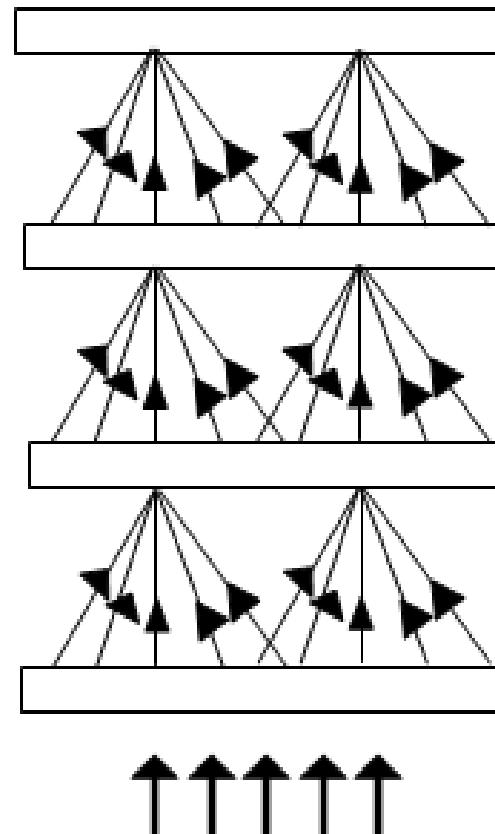
Sanger's model

- Since M-principal components need to be discovered in this case, we have a network with 'm' output neurons.
- The neurons are linear as before.
- The output of the i'th output neuron can therefore be expressed as,
$$y_i = \sum_{j=1}^m w_{ij} x_j$$
- In matrix form:
$$y = w_i^T x = x^T w_i$$
- Weight training:

$$\Delta w_{ij} = \eta y_i (x_i - \sum_{k=1}^i w_{kj} y_k)$$

Linsker's model

- It is a model of the visual system that consists of a multilayered network trained by Hebbian learning.
- The layers in the network are two-dimensional, analogous to the sheets of neurons in various stages of the real visual system.



Linsker's model

- Neurons are all layers are linear.
- Each neuron in a given layer receives inputs from a local neighborhood in the previous layers, a feature that is also inspired the connectivity patterns in the visual system.
- *The training is done in stages, one layer at a time.* Due to such layer-wise training, weights in each weight stage evolve differently.
- Rich response patterns like orientation sensitivity is exhibited as a result

- Consider the response, y , of a neuron in one of the layers of Linsker's model.

$$y = a + \sum_{j=1}^K w_j V_j$$

Where V_j could be the input pattern, x_j , or the response of a neuron in the previous layer.

- A variation of Hebb's rule is used to train the weights, w_i :

$$\Delta w_i = \eta(V_i y + bV_i + cy + d)$$

The first term on the right-hand side (the product) is the product term that appears in the original Hebb's rule.

The parameters b , c and d can be tuned appropriately for various response properties.

- The weights are prevented from blowing up by clipping them as follows,

$$w_- < w_i < w_+$$

- To compute the final values to which the weights converge, let us consider the average change in weights, which must be zero at convergence.

$$0 = E[\Delta w_i] = \eta(E[V_i y] + bE[V_i] + cE[y] + d)$$

- Let \bar{v}_i where v_i is the deviation from the mean. Then,

$$\begin{aligned} E[\Delta w_i] &= \eta(E[(\bar{V} + v_i)(a + \sum_j^K w_j(\bar{V} + v_j))] + b\bar{V} + cE[(a + \sum_j^K w_j(\bar{V} + v_j))] + d) \\ &= \eta(aE[(\bar{V} + v_i)] + E[\sum_j^K w_j(\bar{V} + v_j)(\bar{V} + v_i)] + b\bar{V} + ac + cE[\sum_j^K w_j(\bar{V} + v_j)] + d) \\ &= \eta(a\bar{V} + \sum_j^K w_j\bar{V}^2 + E[\sum_j^K w_j v_j v_i] + b\bar{V} + ac + cE[\sum_j^K w_j(\bar{V} + v_j)] + d) \\ &= \eta(a\bar{V} + \sum_j^K w_j\bar{V}^2 + \sum_j^K w_j C_{ij} + b\bar{V} + ac + c \sum_j^K w_j \bar{V} + d) \end{aligned}$$

Where C_{ij} is the the (i,j) element of the covariance matrix $E[v_i v_j]$.

- Terms in the last equation can be regrouped as,

$$\begin{aligned}
 &= \eta \left(\sum_j^K w_j \bar{V}^2 + \sum_j^K w_j C_{ij} + a\bar{V} + b\bar{V} + ac + d + c \sum_j^K w_j \bar{V} + d \right) \\
 &= \eta \left(\left(\sum_j^K w_j C_{ij} \right) + (a\bar{V} + b\bar{V} + ac + d) + (\bar{V}^2 + c\bar{V}) \left(\sum_j^K w_j \right) \right) \\
 &= \eta \left(\left(\sum_j^K w_j C_{ij} \right) + \lambda(\mu - \sum_j^K w_j) \right)
 \end{aligned}$$

where l and m are functions of the constants a, b, c, d and .

- The above weight dynamics can be interpreted as gradient descent over a cost function:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

- Hence,

$$E = -\frac{1}{2} w^T C w + \frac{\lambda}{2} (\mu - \sum_j w_j)^2$$

The first term is similar to standard Hebbian learning that maximizes quadratic form associated with the autocorrelation matrix of the input data. The second term is a Lagrangian multiplier that imposes the condition $\sum_j w_j = \mu$ on the weights.

Response on training

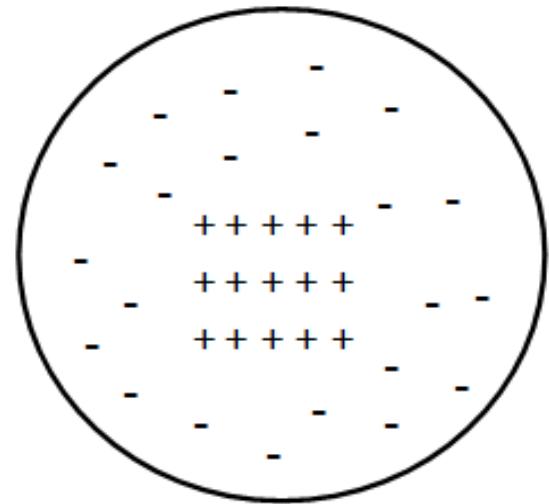
- On 7 two-dimensional layers of neurons, A to G, where each layer projects to the subsequent layer in a ‘pyramidal’ fashion.
- Training is done in a sequential fashion.
- Weights of a stage are first trained to saturation, before the next stage is trained. Note that all training in this model is unsupervised governed by the model eqns.
- In this kind of multilayer model, trained by simple Hebbian mechanism, neurons
 - Center-surround kind of receptive fields in lower layers,
 - Orientation sensitivity in higher layers.

Response of first two layers

- Independent random noise is given as input to the first layer, A.
 - The autocorrelation matrix is an identity matrix.
 - For a range of parameters all the weight saturated to w_+ (since all the weights are the same).
- The response of Layer B is simply a local average, or smoothed version of the image presented to layer A.
 - Neural activation turned out to have high local correlation.
 - Beyond a small radius of high correlation, layer B neurons had low correlation.

Centre surround receptive fields

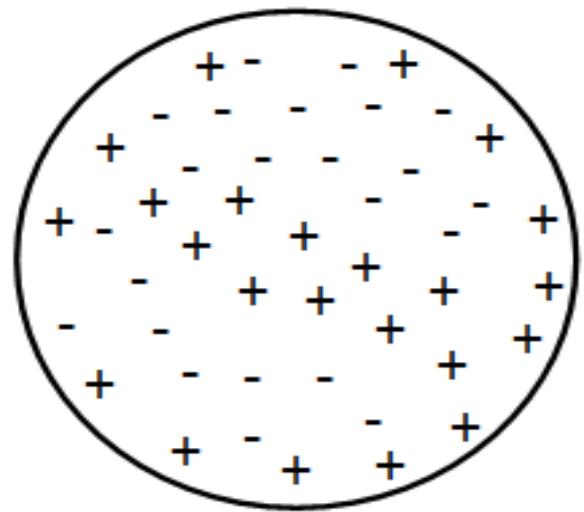
- Neurons in layer C developed center-surround kind of receptive fields.
- They responded strongly either to a bright dot with a dark background, or to a black dot with a white background.
- This trend continued to all the way to layer F where neurons had center-surround receptive fields.



The Center-Surround receptive field in lower layers

Response of higher layers

- In Layer G the parameter values were changed.
- It produced a variety of weight patterns on training, many of them were asymmetric.
- Some neurons had receptive fields with alternating bands of positive and negative weights. Such cells had orientation sensitivity. Some other cells had a central positive region surrounding by several islands of negative regions.



The orientation sensitivity in higher layers