1956

1966

1969

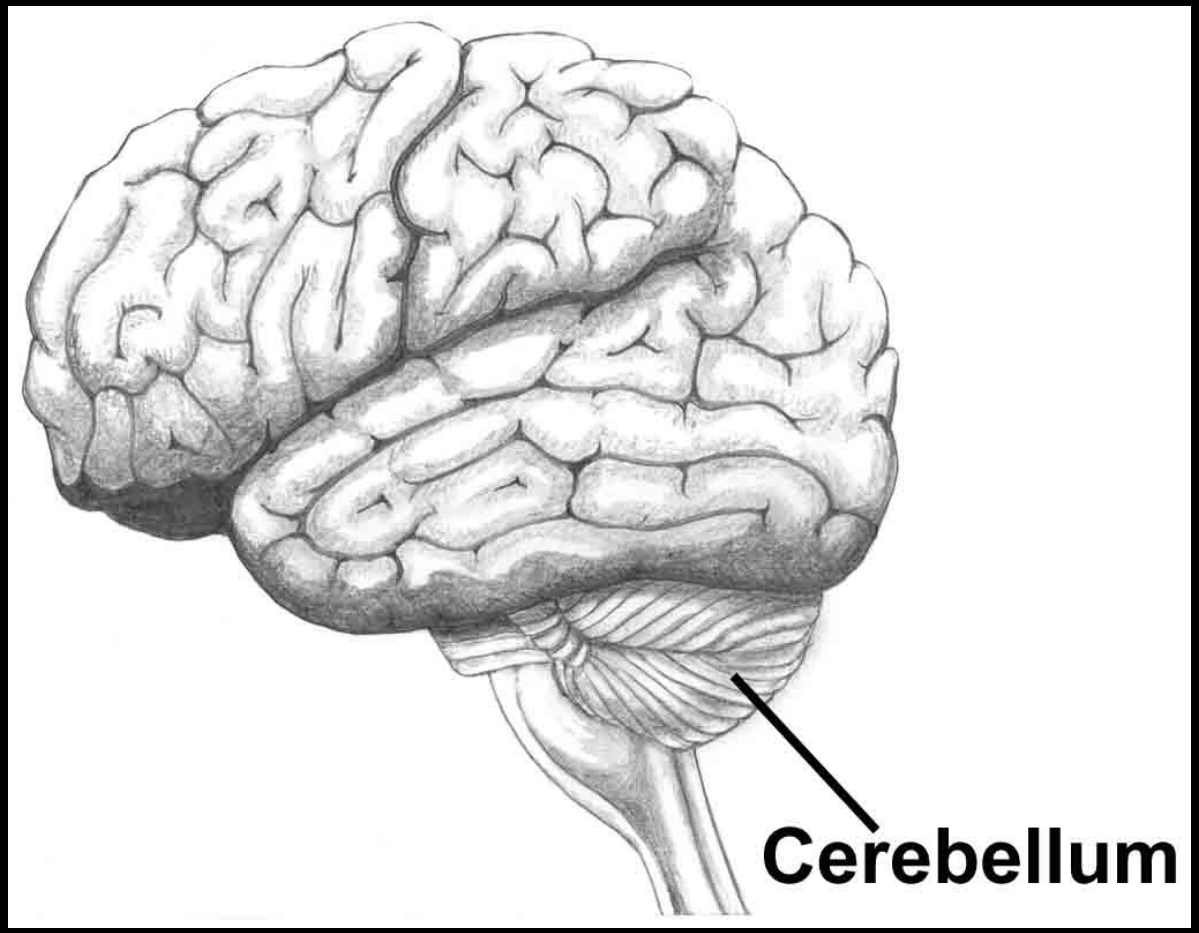
A THEORY OF CEREBELLAR CORTEX

By DAVID MARR*

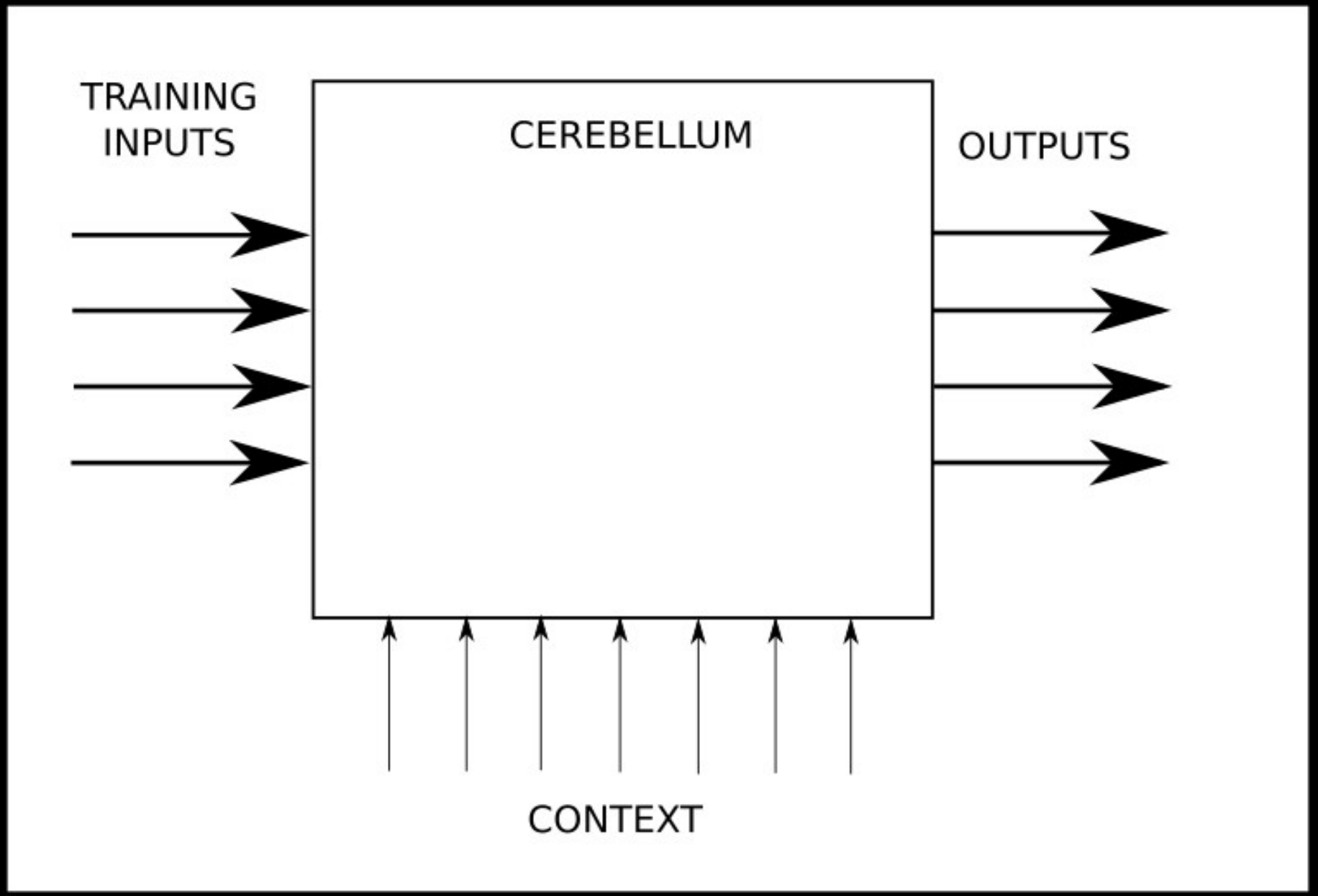
From Trinity College, Cambridge

(*Received* 2 *December* 1968)

Cerebellum



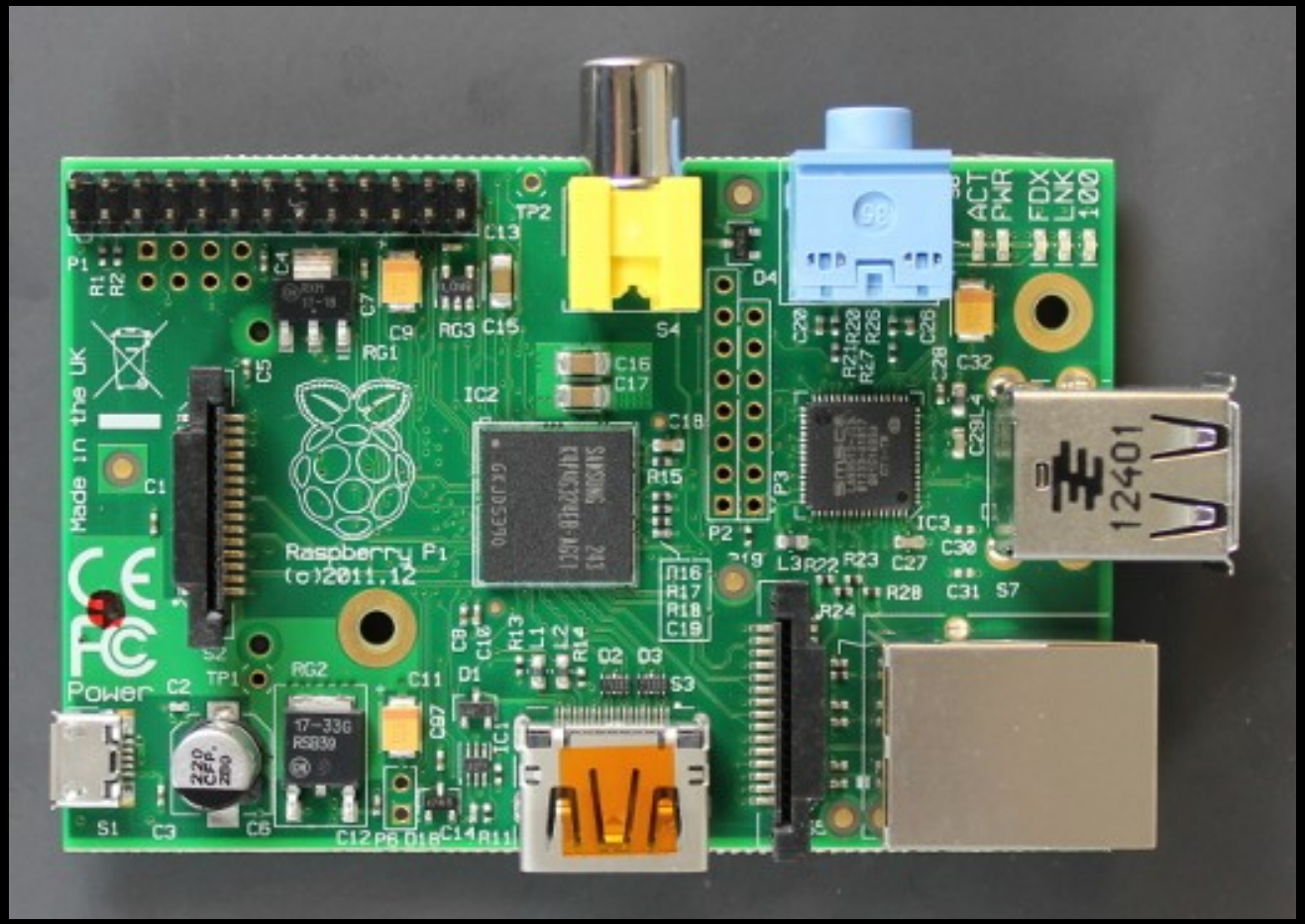
Marr's Model



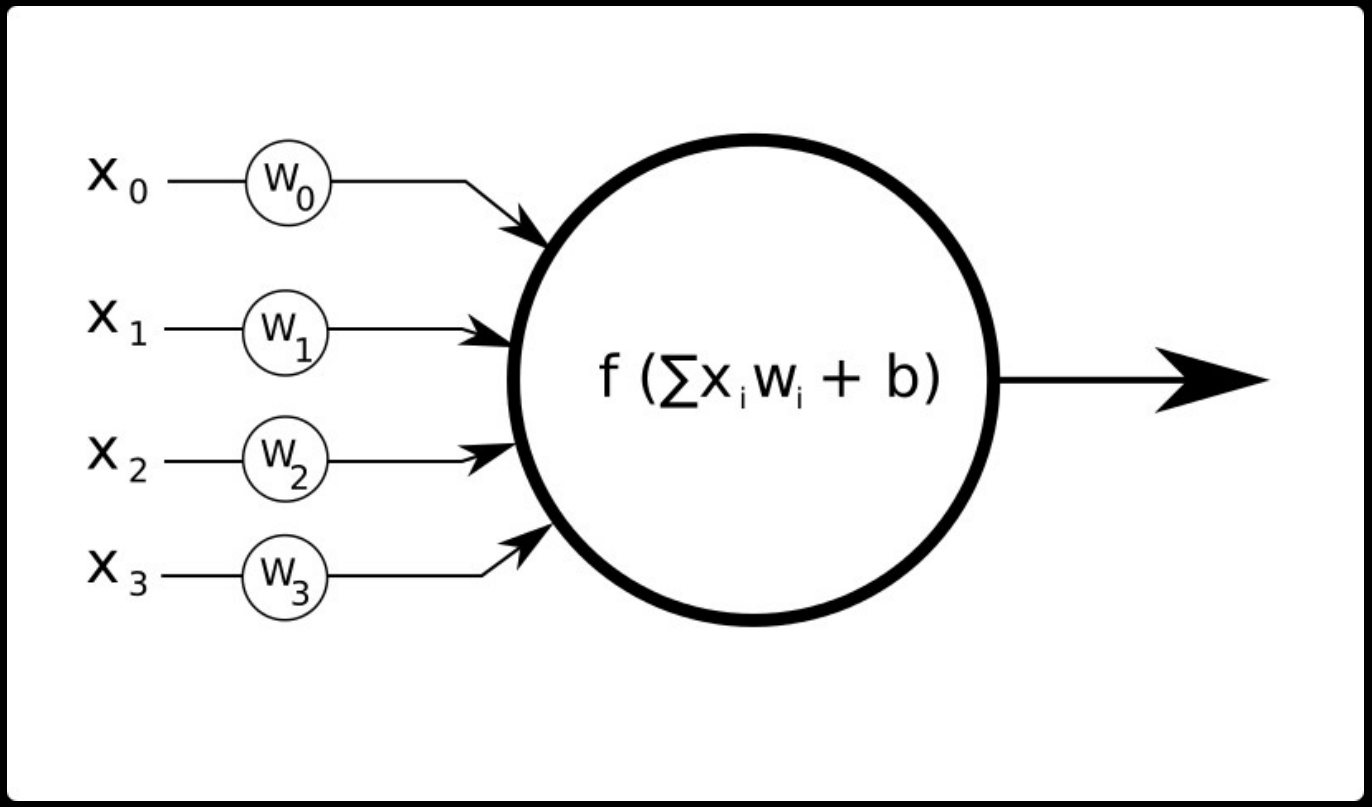
1974


BIRKBECK COLLEGE
1823 1951 1885
IN NOCTE CONSILIVM
NO PARKING IN THIS AREA
CCTV

2012

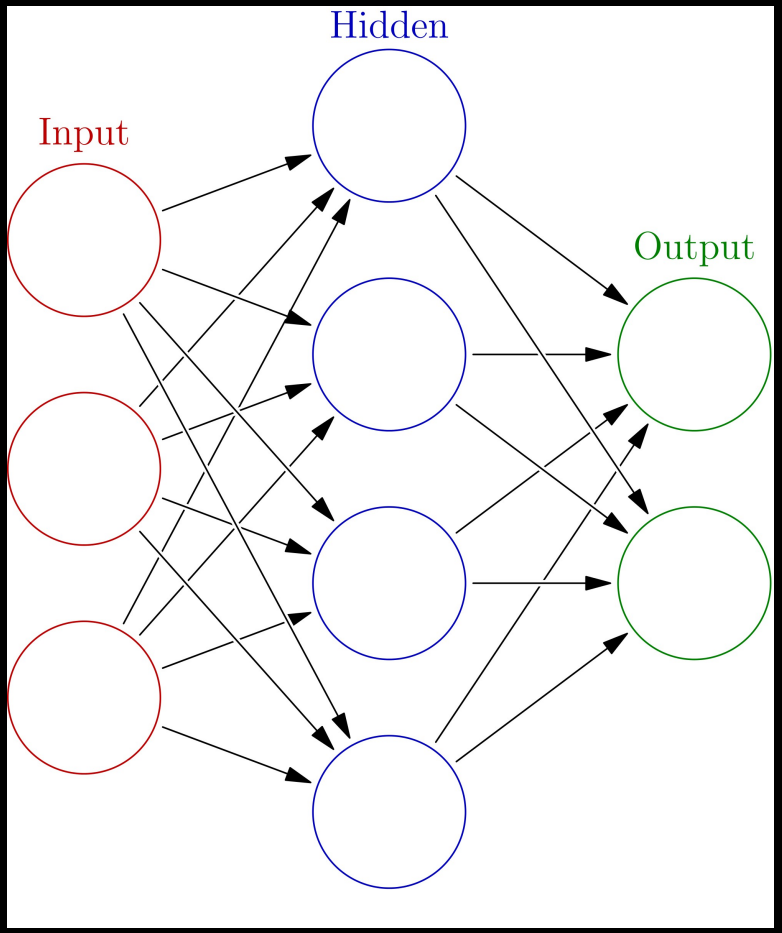

Made in the UK
Raspberry Pi
(c)2011.12
Power
ACT
PWR
FDX
LNK
100
12401

2014



x – inputs; w – weights; b – bias
f – *transfer* function;
f∑(x . w) + b is the *activation* function

2016



Who uses ANNs?

Everybody's doing it...

Google

facebook

NETFLIX

Baidu 百度

HSBC

...including me :)

- ALGOL60 => Python => APL
- Blogging
- Stealing code from Gil and Phil
- Talk at Birkbeck

Marr's model in APL

```
cycle←{α granfire ω granuleThresholds α}
fireGolgis←{(0⌈+/,α)++/[1]+/[3]2 5 2 5⍴ω}
granfire←{ω<0⌈+⌿neighbours α}
granuleThresholds←{spread thresholds α fireGolgis ω}
neighbours←{ω⌿(1⊖ω)⌿(1⊖ω),[¯0.5]1⌿1⊖ω }
spread←{5 5\[0]5 5\[1]ω}
thresholds←{α←0 0.3 0.5 0.7 ◊ +/[0]α∘.≤ω÷25}
view←{0.2×α[;ω+⍳10]}
```

Current Tools

- PyBrain
- SciPy/numpy
- Numenta (HTM)
- GPU
  - Google TensorFlow
  - FB Torch
  - Theano
  - Caffe
- Amazon DSSTNE

Python

```
import random
from math import exp


def random_vector(cols):
    return list([random.random() for i in range(cols)])

def random_vov(rows, cols):
    return list([random_vector(cols) for j in range(rows)])

def dot_product(v1, v2):
    return sum((a*b) for (a,b) in zip(v1, v2))

def inner_product(vov, v2):
    return list([dot_product(v1, v2) for v1 in vov])

def sigmoid(x):
    return 1.0/(1.0+exp(-x))

def sigmoid_neuron(vov, v2):
    return list([sigmoid(x) for x in inner_product(vov, v2)])


mat = random_vov(3, 4)
vec = random_vector(4)
print sigmoid_neuron(mat, vec)
```

APL

```
mat ←0.01×?3 4ρ100
vec ← 0.01×?4ρ100
sn  ← {÷1+*-⍺+.×⍵}
mat sn vec
```

Speed

APL 100 x faster than Python
About the same as numpy
Better with multiple cores
100x slower than GPU-based code

ANN = APL Tipping Point?

- Lots of maths on big arrays
  - Some boolean, some sparse
- Parallel Computing
- And one day, GPU?