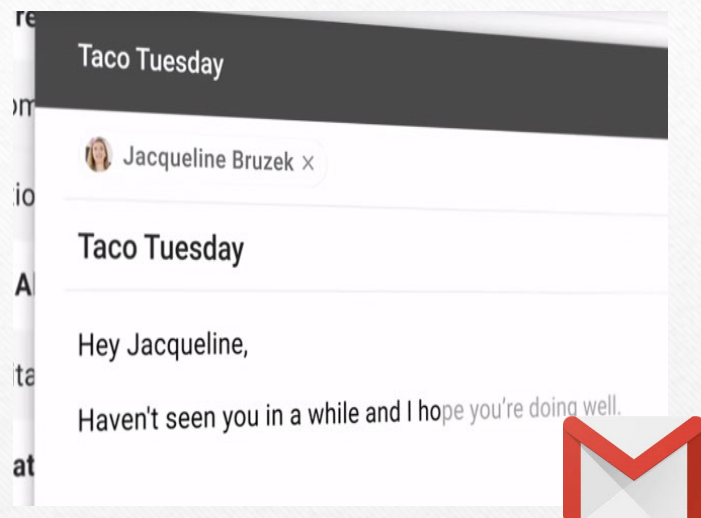


# Artificial Neural Network

---

ECON 4810



#### We've Suggested Tags for Your Photos

We've automatically grouped together similar pictures and suggested the names of friends who might appear in them. This lets you quickly label your photos and notify friends who are in this album.

#### Tag Your Friends

This will quickly label your photos and notify the friends you tag. [Learn more](#)



Who is this?



Who is this?



Who is this?



Who is this?



Who is this?



Who is this?

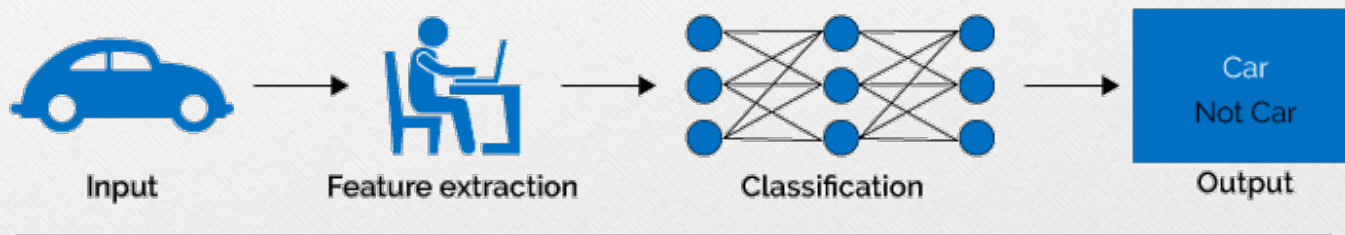




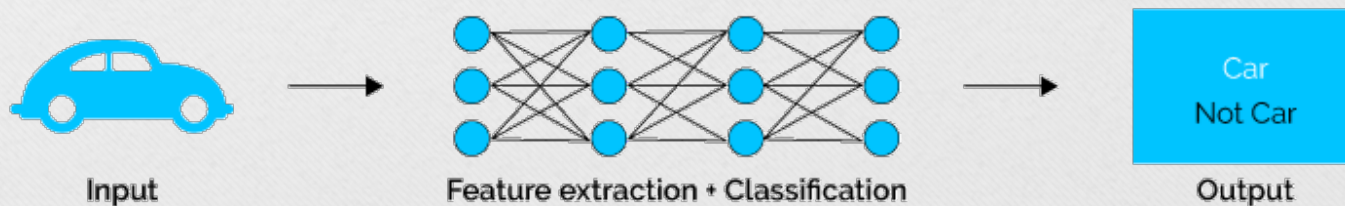
# Artificial Neural Network

- The most prominent A.I. systems today are powered by artificial neural networks (ANN).
- ANN is the pinnacle of model complexity

## Machine Learning



## Deep Learning



# Artificial Neural Network

Today:

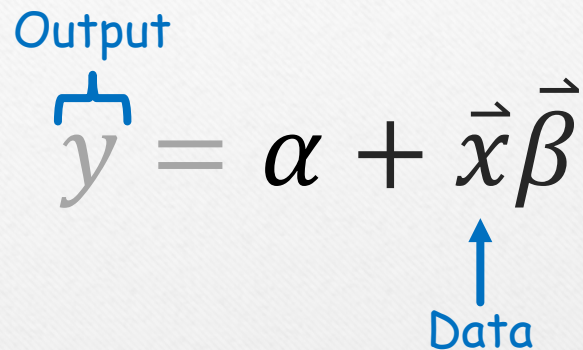
- What is ANN?
- What are the common types of ANN?
- How to write an ANN in Python?

# This is a Linear Regression

Output

$$\hat{y} = \alpha + \vec{x} \vec{\beta}$$

Data



Where  $\vec{x} = [x_1 \quad \dots \quad x_k]$  is one sample of features (a.k.a. one observation of independent variables)

$\vec{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}$  is a vector of weights (a.k.a. coefficients)

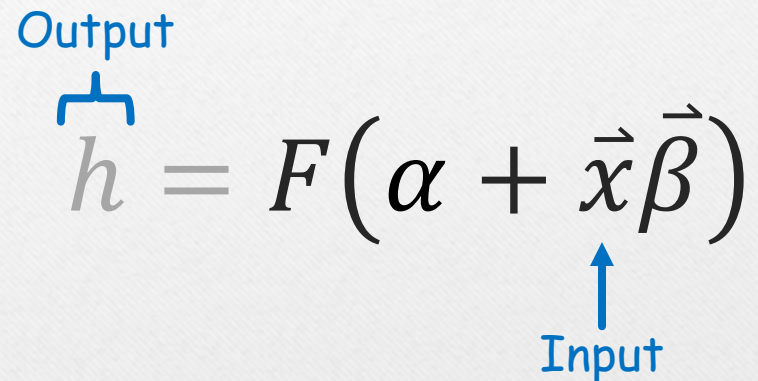


# This is a Logistic Regression

$$\overbrace{P(y = 1)}^{\text{Output}} = F\left(\alpha + \underbrace{\vec{x}\vec{\beta}}_{\text{Data}}\right)$$

$$\text{Where } F(z) = \frac{e^z}{1+e^z}$$

# This is a Neuron



The diagram shows the equation  $h = F(\alpha + \vec{x}\vec{\beta})$ . A blue bracket above the  $h$  is labeled "Output". A blue arrow points from the word "Input" below to the  $\vec{x}$  term in the equation.

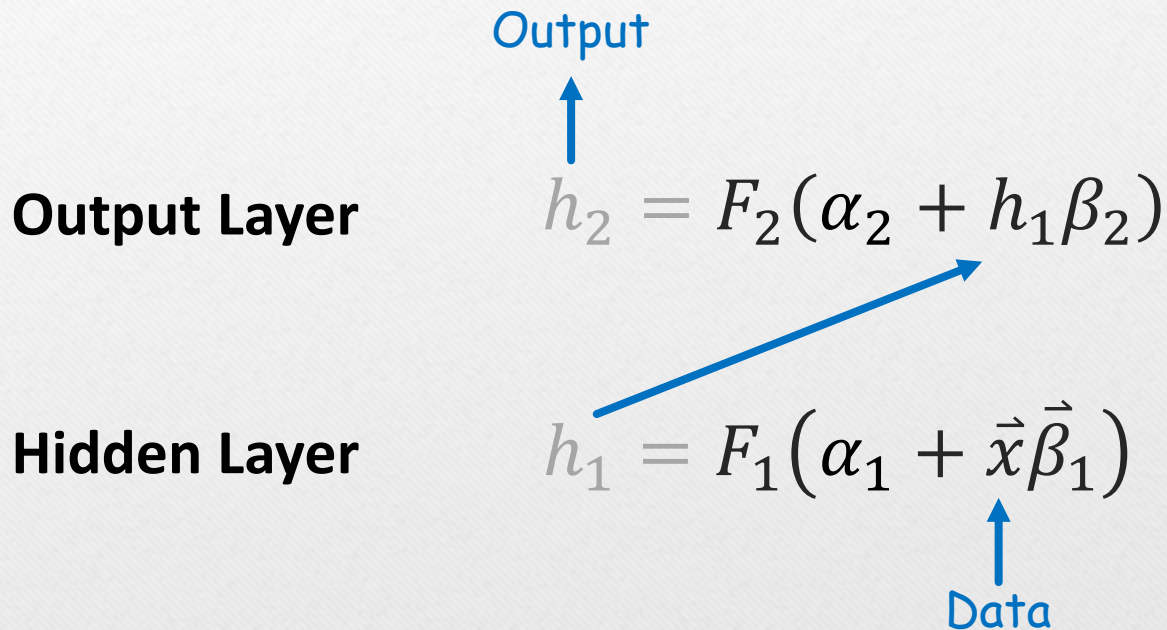
$$\overset{\text{Output}}{\underbrace{h}} = F\left(\alpha + \underset{\substack{\uparrow \\ \text{Input}}}{\vec{x}}\vec{\beta}\right)$$

Where  $F(z)$  is a non-linear function

$F(z)$  is called the **activation function** and  
 $h$  the neuron's **activation**

# Artificial Neural Network

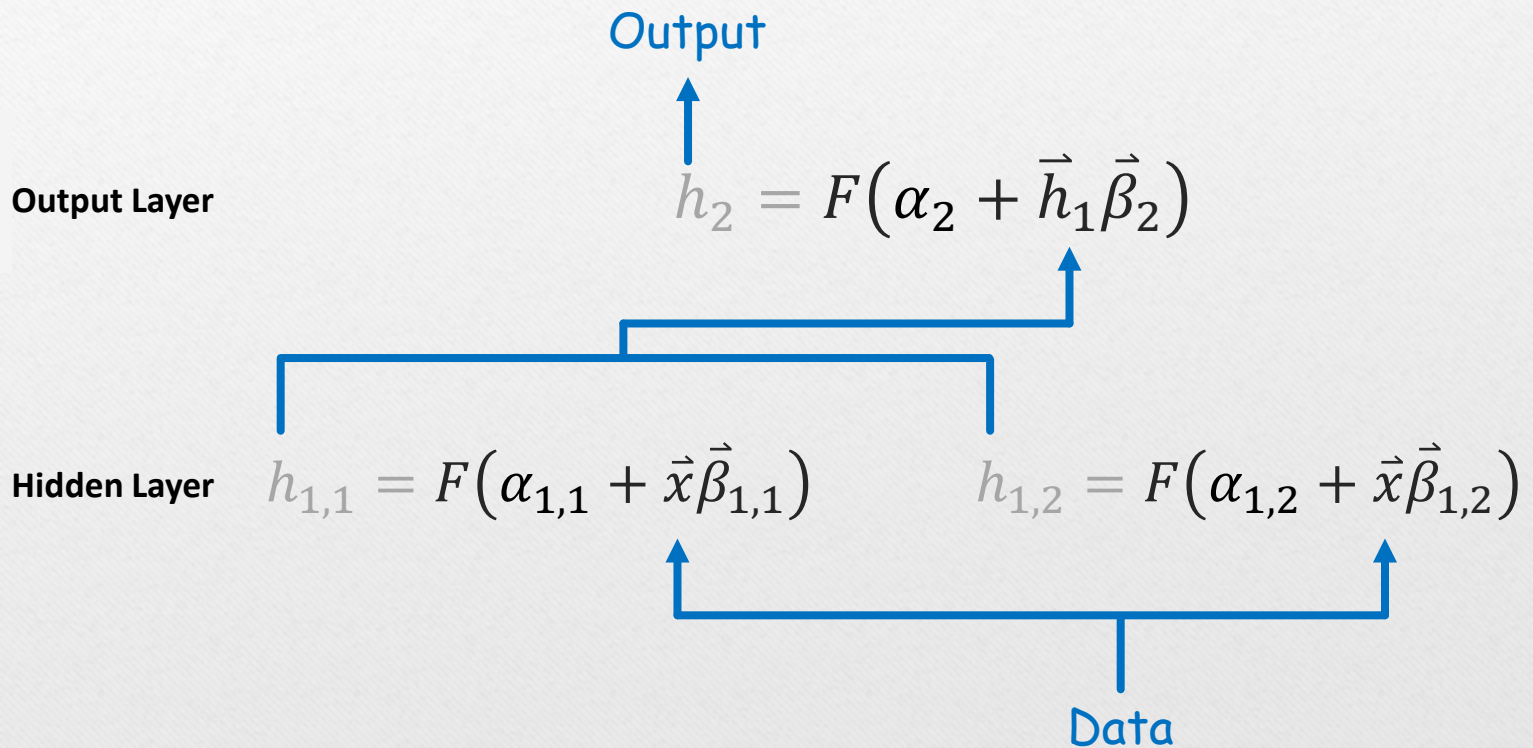
(A Very Simple One)





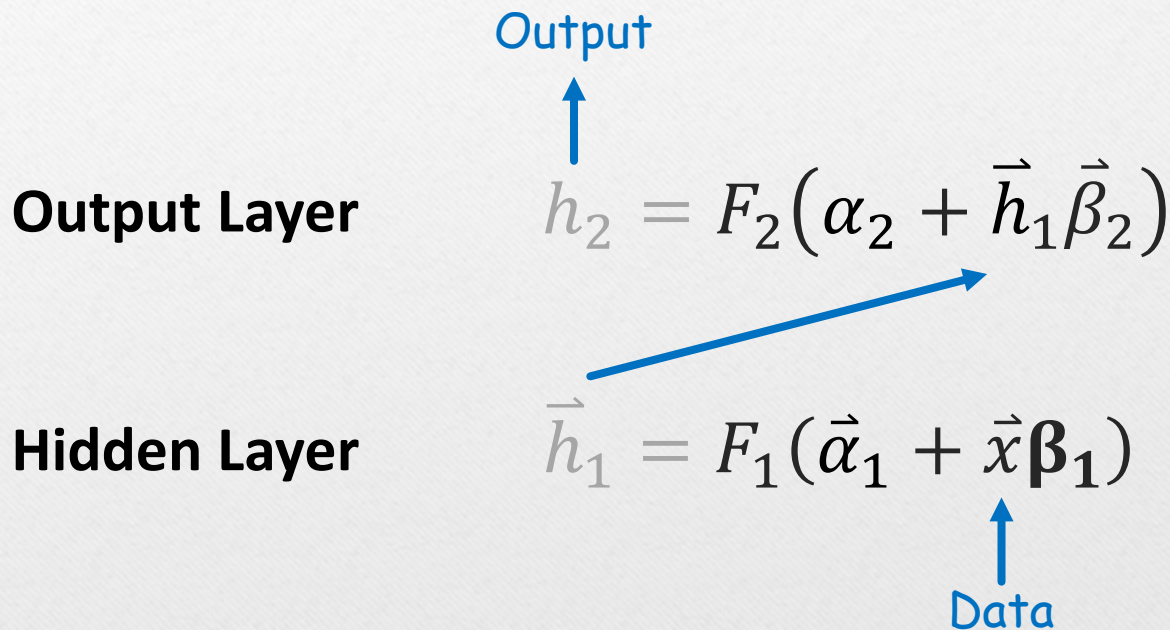
# Artificial Neural Network

(With Two Hidden Neurons)



# Artificial Neural Network

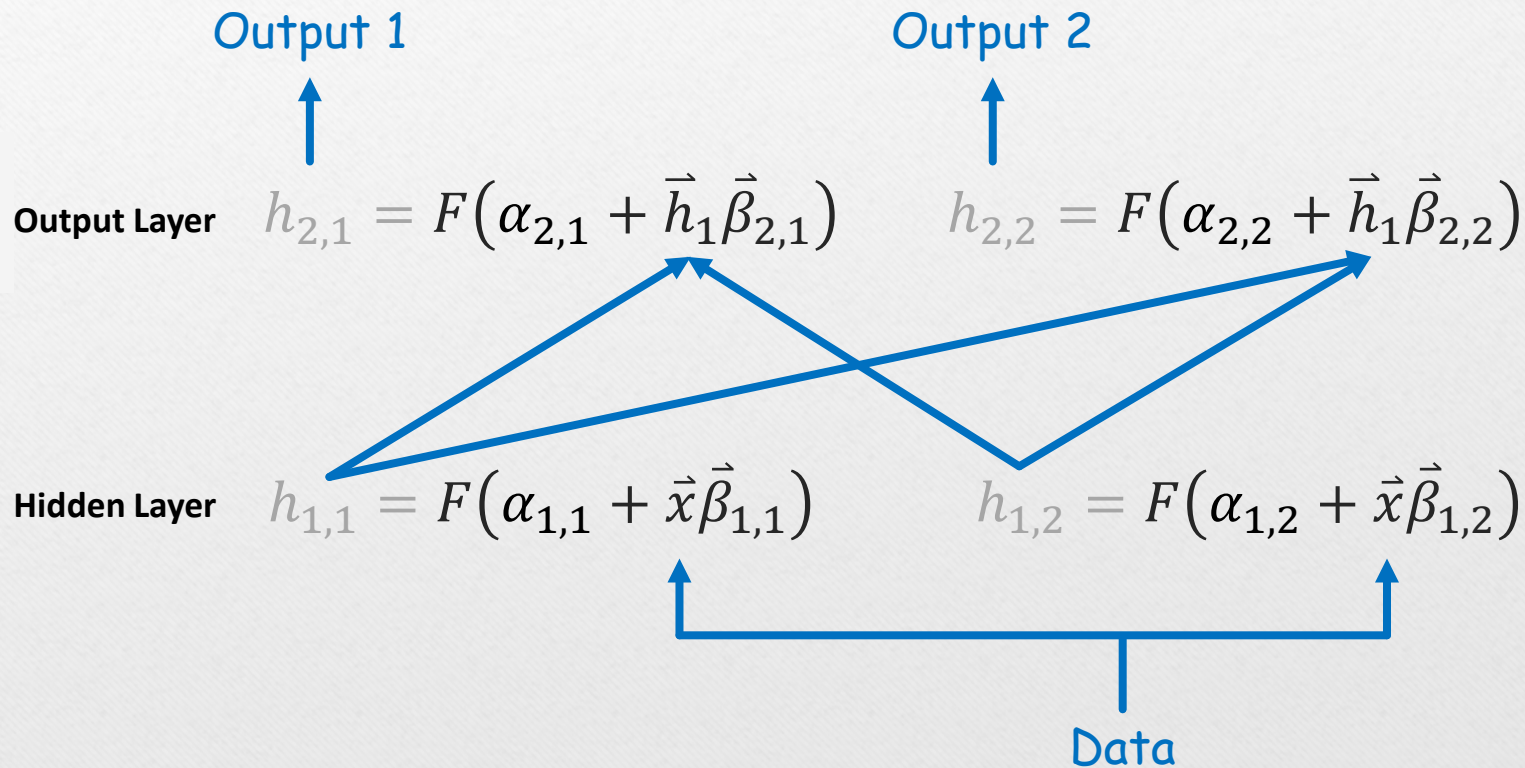
(With Multiple Hidden Neurons)



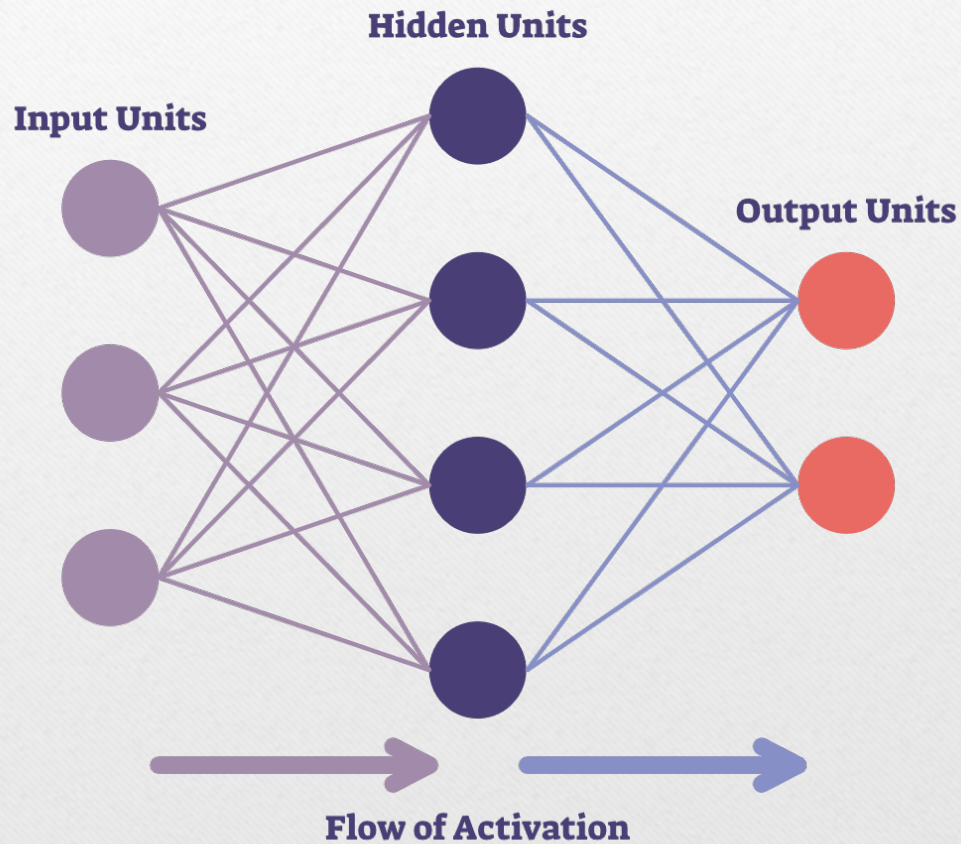


# Artificial Neural Network

(With Two Hidden Neurons and Two Output Neurons)



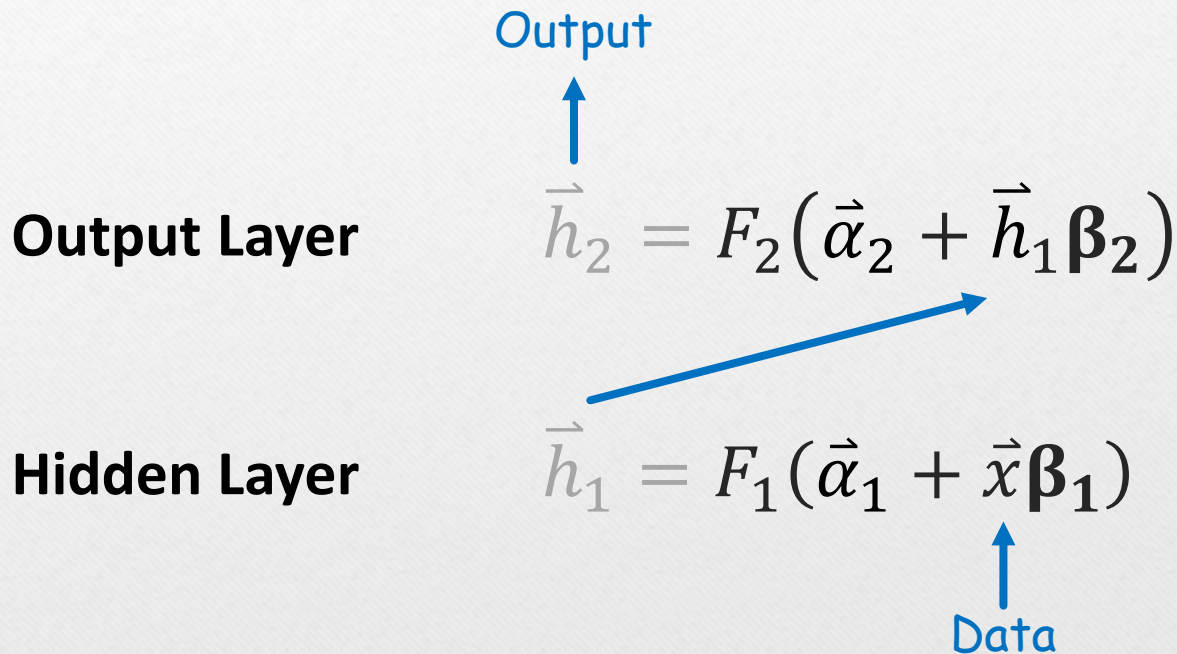
# AI Neural Networks



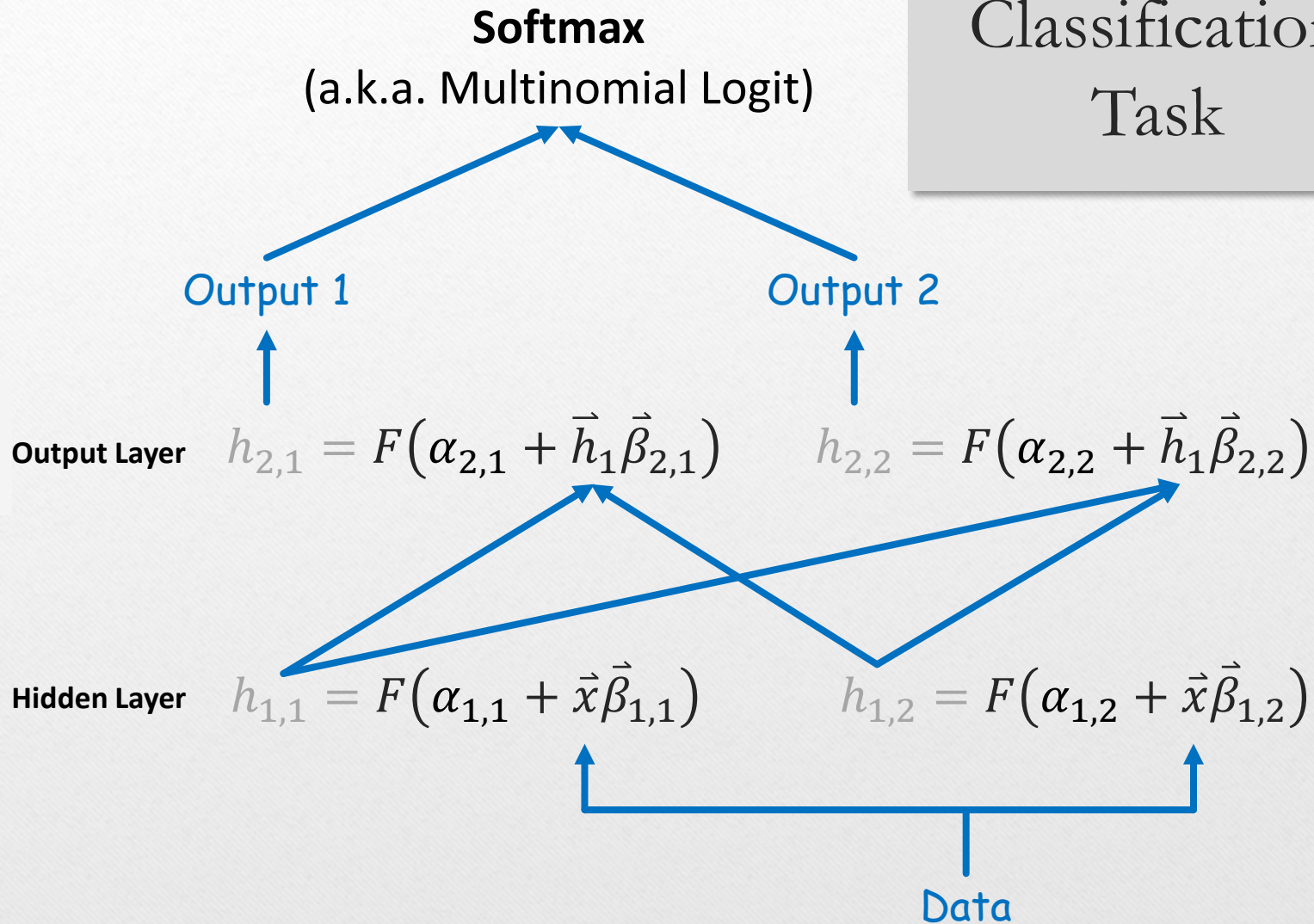


# Artificial Neural Network

(With Multiple Hidden Neurons and Outputs)



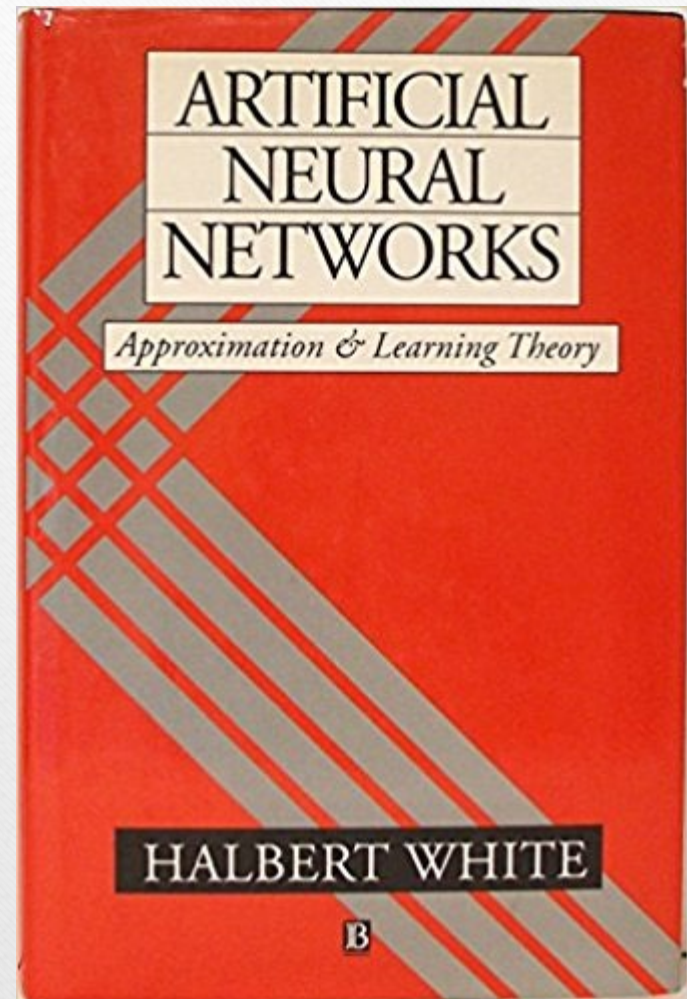
# Classification Task





An econometrician's view  
of artificial neural network:  
a bunch of regressions  
stacked together

- Halbert White made significant contribution to the theoretical foundation of ANN in the 1980s
- Application was rare because the lack of computational power



# Why Does Activation Have to be Non-Linear?

$$h_2 = F_2(\alpha_2 + \beta_2 h_1)$$

$$h_1 = F_1(\alpha_1 + \vec{\beta}_1 \vec{x})$$

If  $F_1$  is linear, you get

$$h_2 = F_2\left(\alpha_2 + \beta_2(\alpha_1 + \vec{\beta}_1 \vec{x})\right)$$

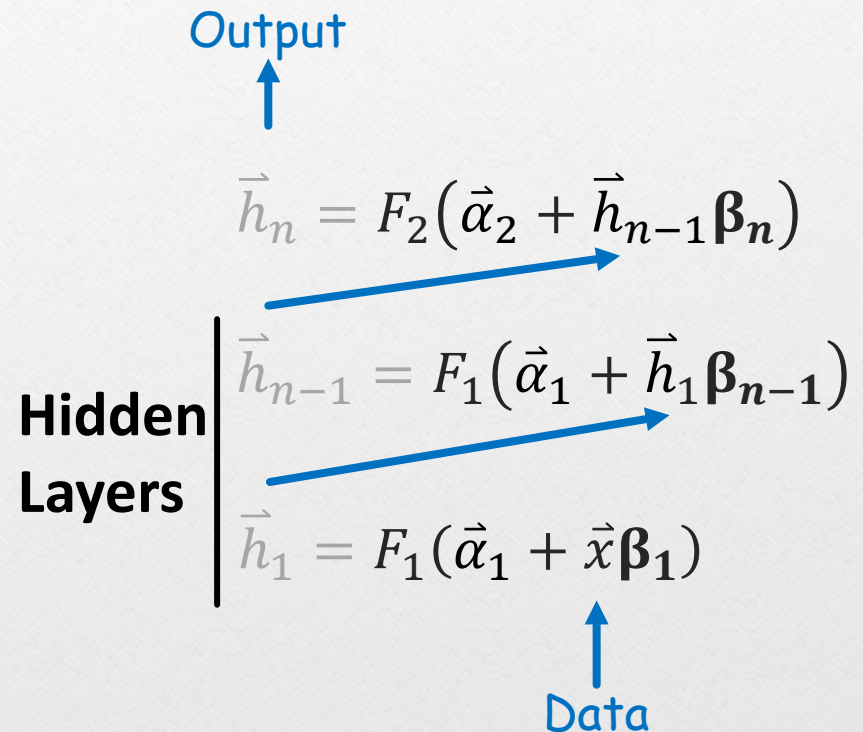
$$= F_2(\alpha'_2 + \vec{\beta}'_2 \vec{x})$$

So there is no point in having the hidden layer.



# Deep Learning

- Deep Learning refers to the stacking of multiple hidden layers
- Typical layer count is in the single digit but can go as high as a hundred.



# How Does an ANN Learn?

- **Gradient Descent**



# Stochastic Gradient Descent

- Learning is quite slow if we only update model weights after we go through all data.
- We could instead update every time after we gone through a given number of samples. This is **Stochastic Gradient Descent (SGD)**.
- Because we are not using all data, we could be updating towards the wrong direction sometimes, but on average the updates will be correct. Hence, *stochastic*.
- The stochastic nature is actually a good property because it helps the model escape from local optima.

# How Does an ANN Learn?

- **Gradient Descent**
- **Back Propagation**



# Back Propagation

- Because neural network have multiple layers, the gradient of lower layers needs be computed using the chain rule. This process is called **back propagation**.

Output



$$\vec{h}_2 = F_2(\vec{\alpha}_2 + \vec{h}_1 \beta_2)$$



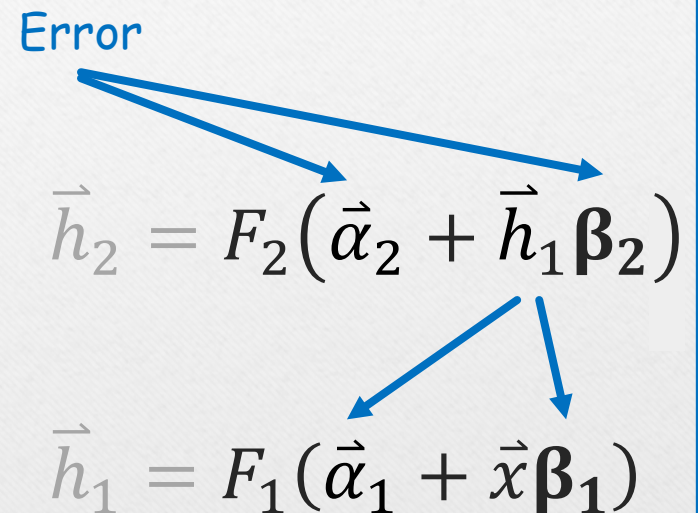
$$\vec{h}_1 = F_1(\vec{\alpha}_1 + \vec{x} \beta_1)$$

Data



# Back Propagation

- Because neural network have multiple layers, the gradient of lower layers needs be computed using the chain rule. This process is called **back propagation**.





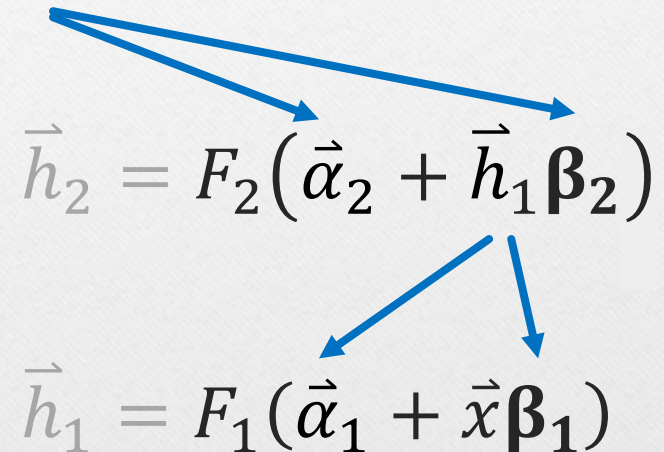
# Back Propagation

E.g. single target regression task, the gradient of the second weight of the first neuron is:

$$\begin{aligned}
 & \frac{\partial}{\partial \beta_1^{(1,2)}} (y - h_2)^2 \\
 &= -2(y - h_2) F_2'(\vec{\alpha}_2 + \vec{h}_1 \vec{\beta}_2) \frac{\partial}{\partial \beta_1^{(1,2)}} (\vec{\alpha}_2 + \vec{h}_1 \vec{\beta}_2) \\
 &= -2(y - h_2) F_2'(\vec{\alpha}_2 + \vec{h}_1 \vec{\beta}_2) \beta_2^{(1)} \\
 &\quad \cdot \frac{\partial}{\partial \beta_1^{(1,2)}} F_1(\alpha_1 + \vec{x} \vec{\beta}_1^{(1)}) \\
 &= -2(y - h_2) F_2'(\vec{\alpha}_2 + \vec{h}_1 \vec{\beta}_2) \beta_2^{(1)} \\
 &\quad \cdot F_1'(\vec{\alpha}_1 + \vec{x} \vec{\beta}_1^{(1)}) x_2
 \end{aligned}$$

An important feature of any neural network library is to automate the computation of gradient.

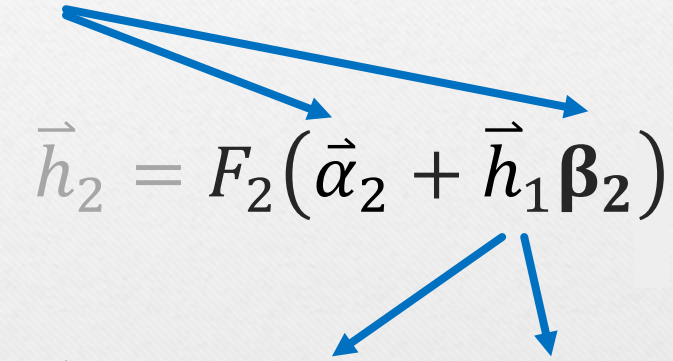
Error



# Back Propagation

- As the number of layers go up, the process becomes quite brittle.
- Remember the chain rule involves a lot of multiplication.
- Small times small means very small, resulting in **vanishing gradient**.
- Big times big means very big, resulting in **exploding gradient**.
- Neither is good for learning.
- A lot of research goes into finding ways to combat the issue: different activation functions, different randomization distributions...

Error


$$\vec{h}_2 = F_2(\vec{\alpha}_2 + \vec{h}_1 \beta_2)$$

$$\vec{h}_1 = F_1(\vec{\alpha}_1 + \vec{x} \beta_1)$$



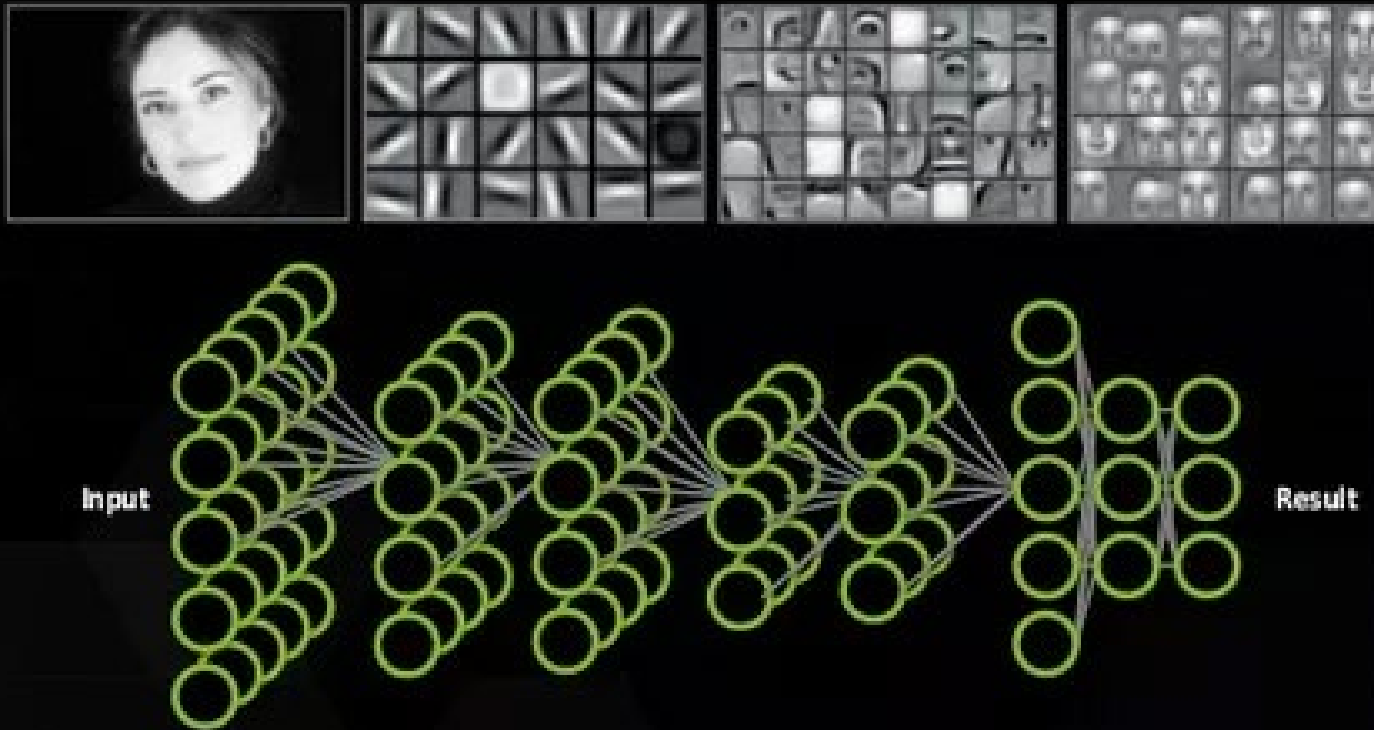
# Computation

---

- The idea of artificial neural network can be traced back to the 1940s.
- Due to the large number of parameters and large data size involved, effective use of ANN is prohibitive until recently.

# 2017 Big

## WHAT MAKES DEEP LEARNING DEEP?



Today's Largest Networks

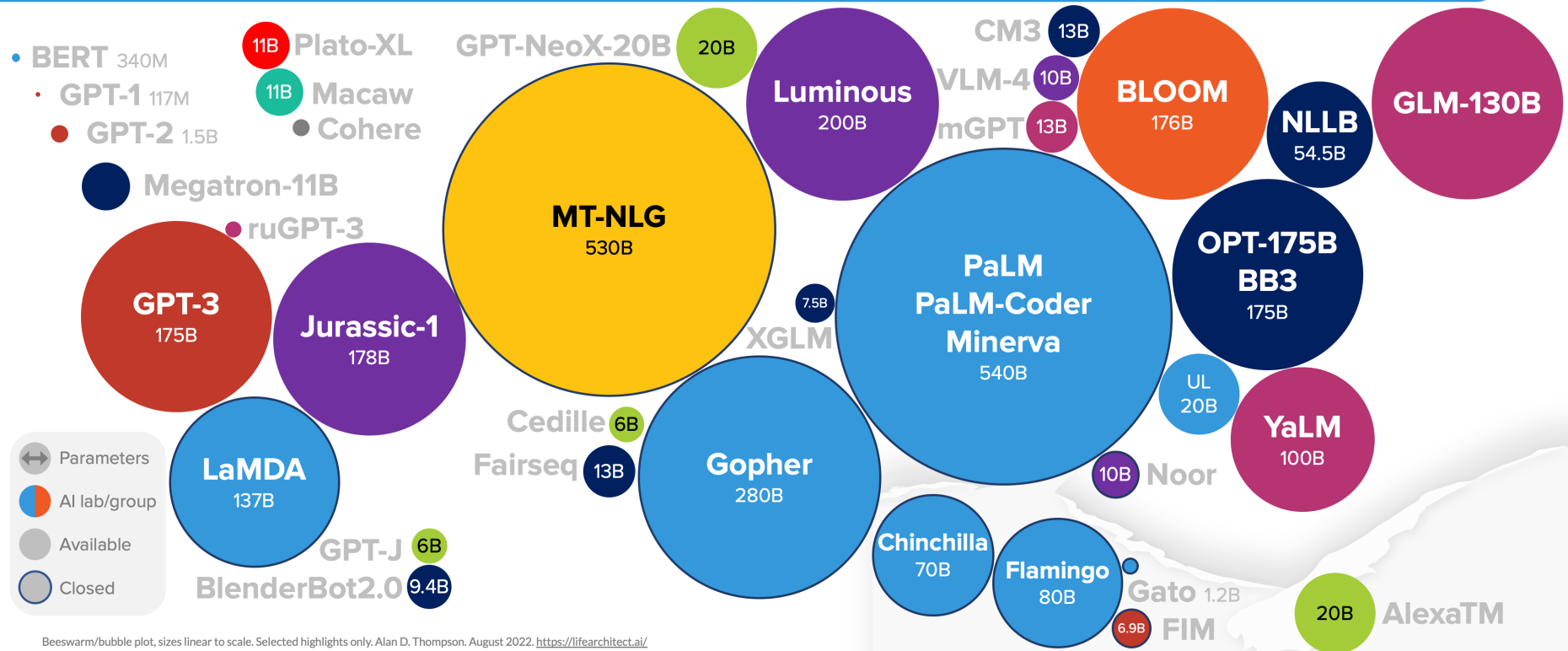
- ~10 layers
- 1B parameters
- 10M images
- ~30 Exaflops
- ~30 GPU days

Human brain has trillions of parameters - only 1,000 more.

Source: Nvidia



# LANGUAGE MODEL SIZES TO AUG/2022



# It Takes a Lot to Train These Models!

Model size	Hidden size	Number of layers	Number of parameters (billion)	Model-parallel size	Number of GPUs	Batch size	Achieved teraFLOPs per GPU	Percentage of theoretical peak FLOPs	Achieved aggregate petaFLOPs
1.7B	2304	24	1.7	1	32	512	137	44%	4.4
3.6B	3072	30	3.6	2	64	512	138	44%	8.8
7.5B	4096	36	7.5	4	128	512	142	46%	18.2
18B	6144	40	18.4	8	256	1024	135	43%	34.6
39B	8192	48	39.1	16	512	1536	138	44%	70.8
76B	10240	60	76.1	32	1024	1792	140	45%	143.8
145B	12288	80	145.6	64	1536	2304	148	47%	227.1
310B	16384	96	310.1	128	1920	2160	155	50%	297.4
530B	20480	105	529.6	280	2520	2520	163	52%	410.2
1T	25600	128	1008.0	512	3072	3072	163	52%	502.0

Source: Nvidia



# Computation

---

- ANN took off due to massive increase in computational capabilities, particularly in the use of graphic processing unit (GPU) for computation.

