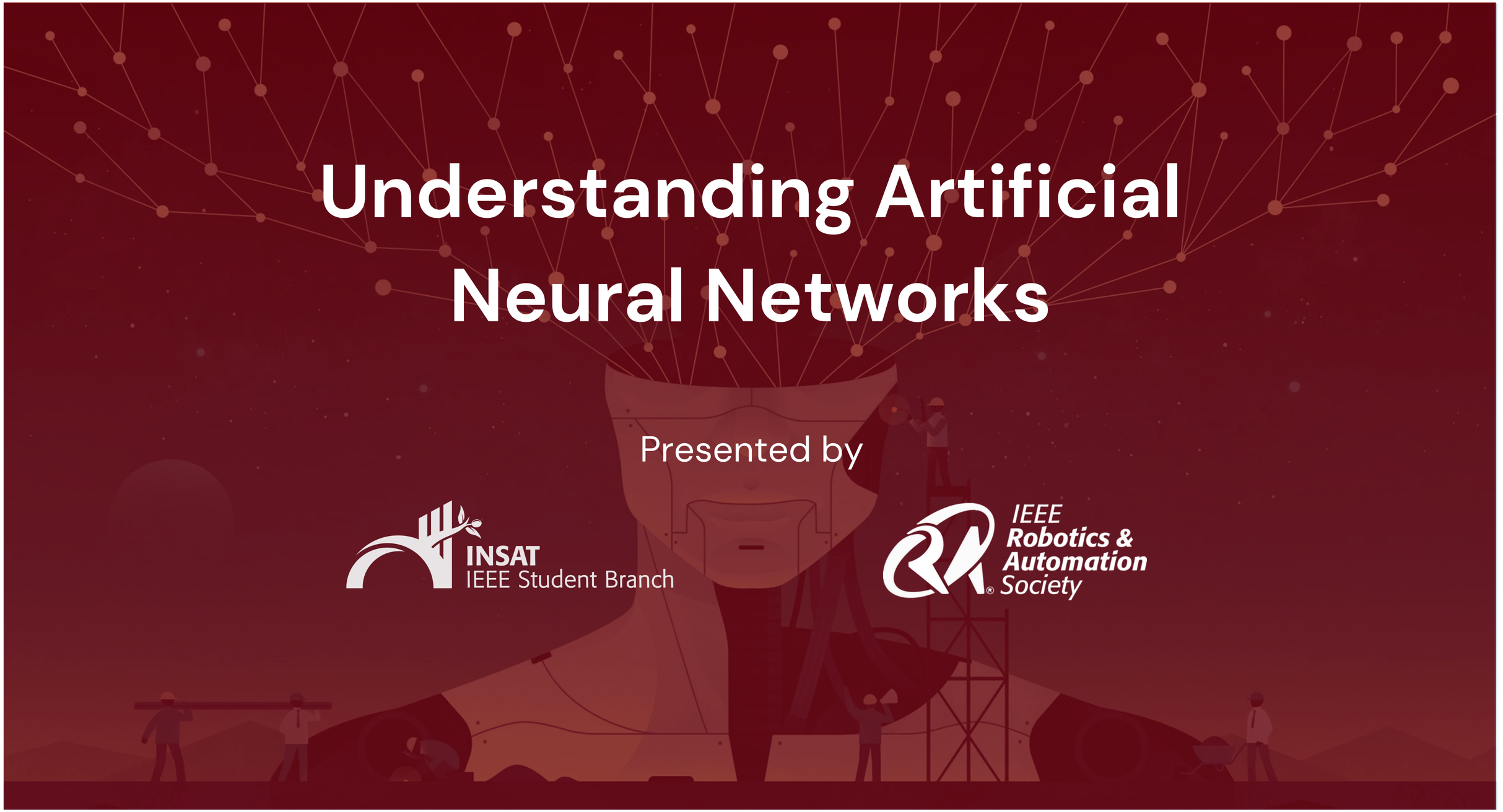
Understanding Artificial Neural Networks



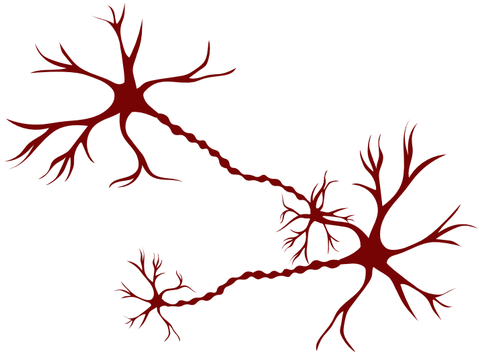
Presented by

INSAT
IEEE Student Branch

IEEE
Robotics & Automation Society

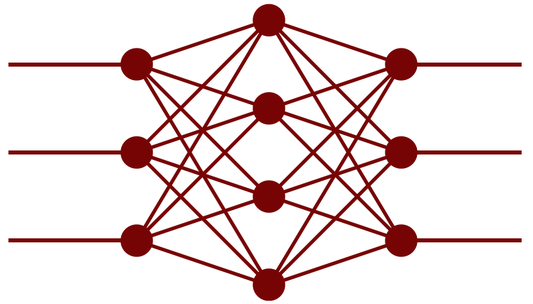
What Are Neural Networks?

- A neural network (ANN) is a machine learning model inspired by the human brain's neural structure.
- It consists of interconnected nodes, or artificial neurons, that mimic brain neurons.



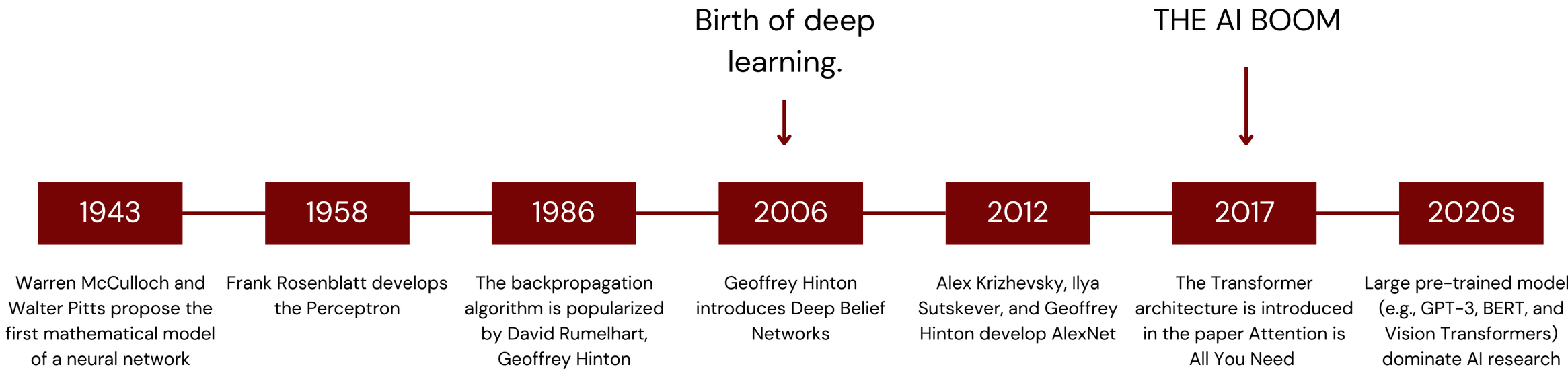
Biological Neurons

Modelisation



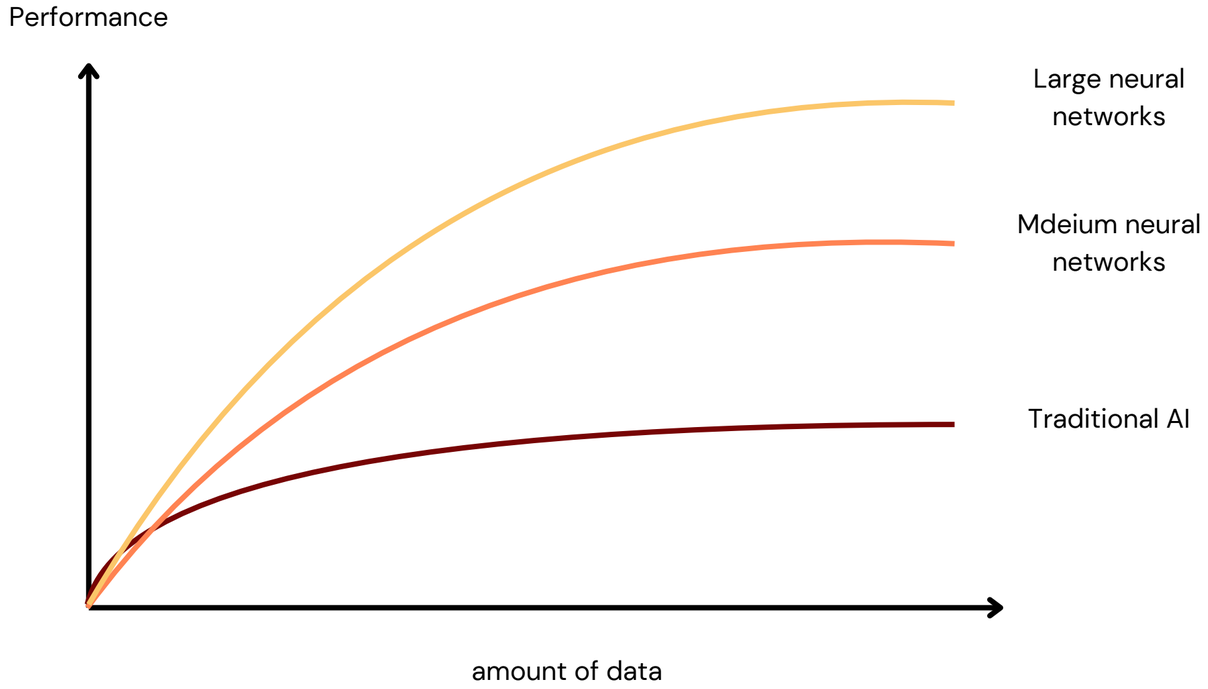
Artificial Neurons

History of Neural Networks



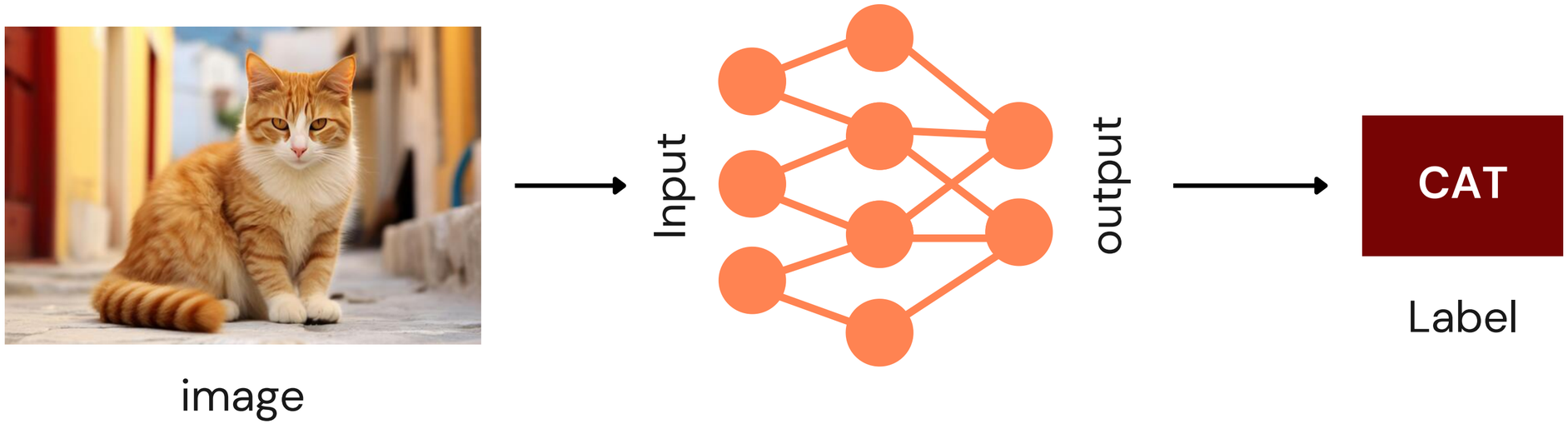
Why Are Neural Networks Relevant Today?

- Explosion of data: Neural networks thrive on large datasets.
- Advancements in computing power: GPUs and TPUs.



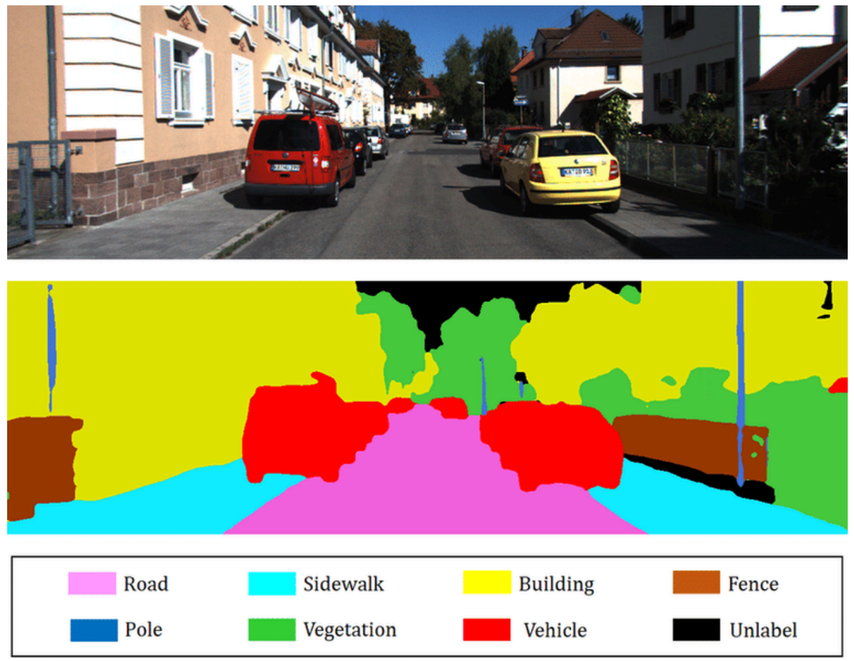
What Can We Do With Neural Networks?

Classification

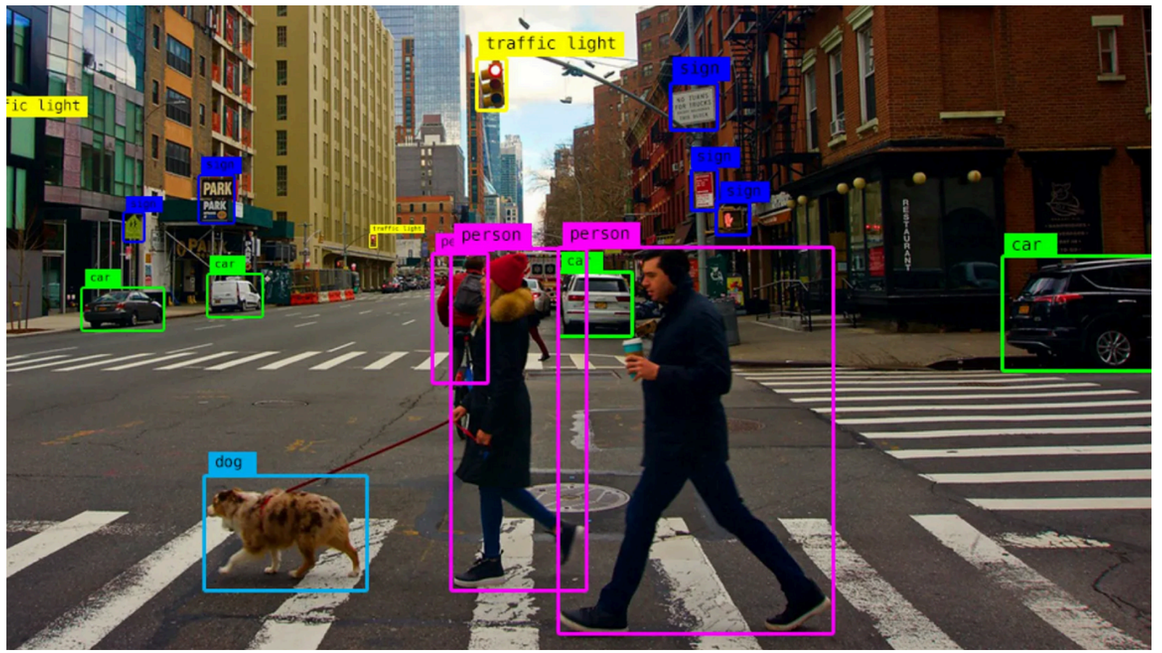


What Can We Do With Neural Networks?

Tasks beyond classification



Semantic segmentation

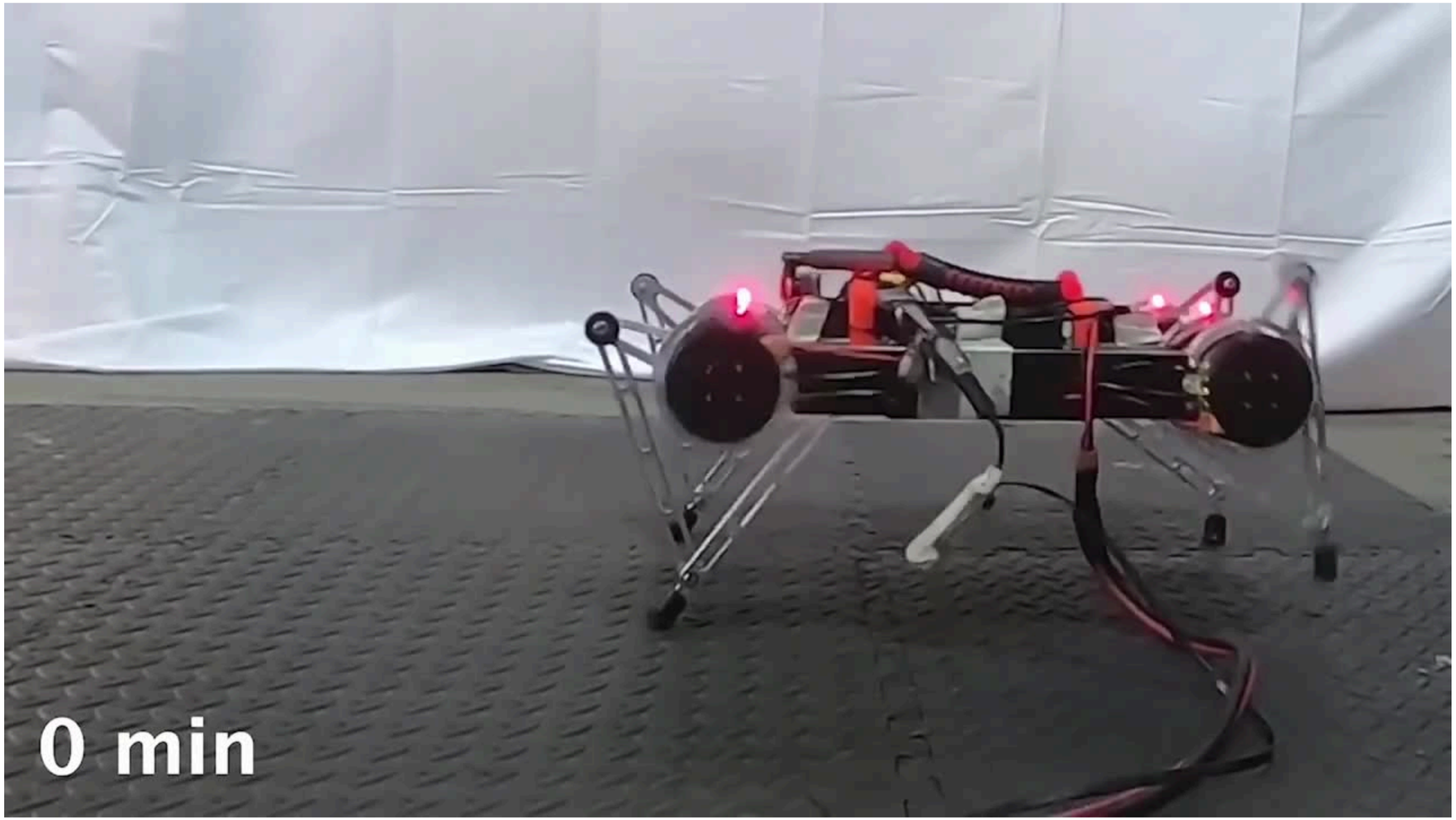


Object detection

What Can We Do With Neural Networks?

Video Understanding

What Can We Do With Neural Networks?



Controlling physical systems

What Can We Do With Neural Networks?

Style transfere

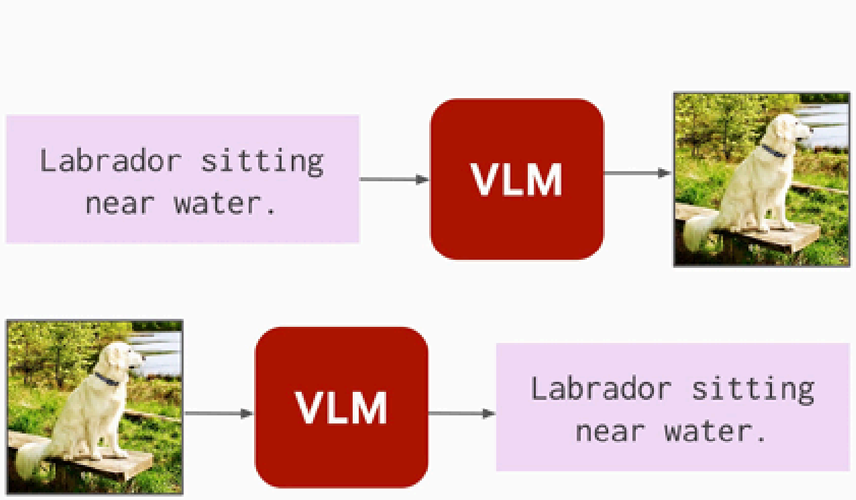
What Can We Do With Neural Networks?

"Teddy bears working on new AI research underwater with 1990s technology"

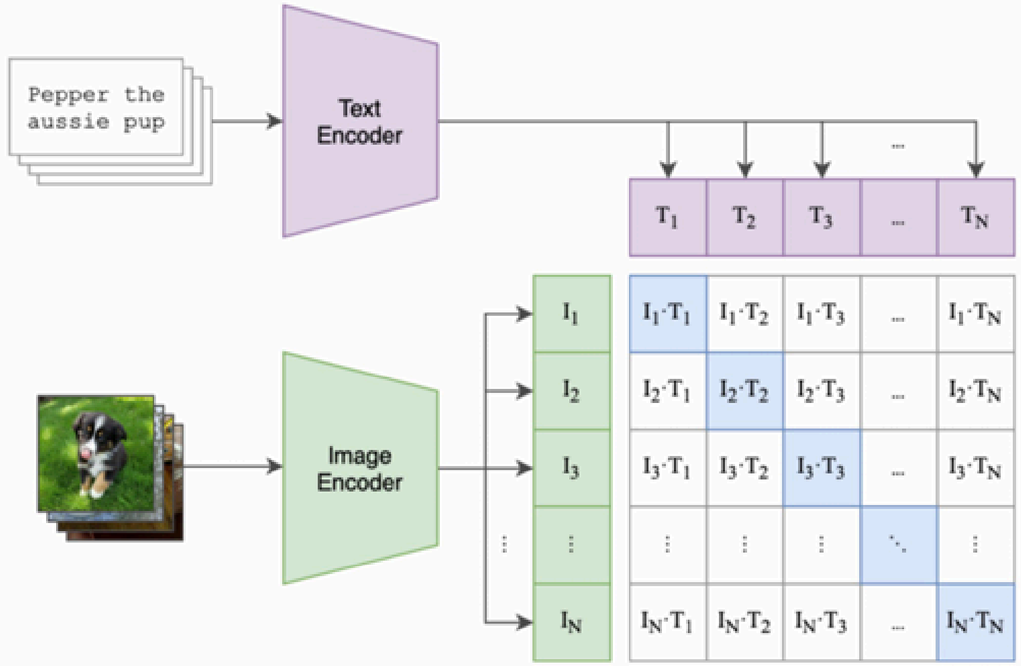
DALL-E 2

Generative modeling

What Can We Do With Neural Networks?



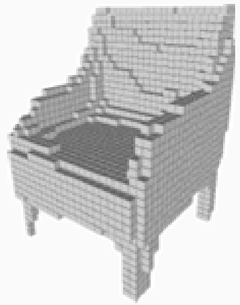
Yasunaga, Michihiro, et al. "Retrieval-augmented multimodal language modeling." arXiv preprint arXiv:2211.12561 (2022).



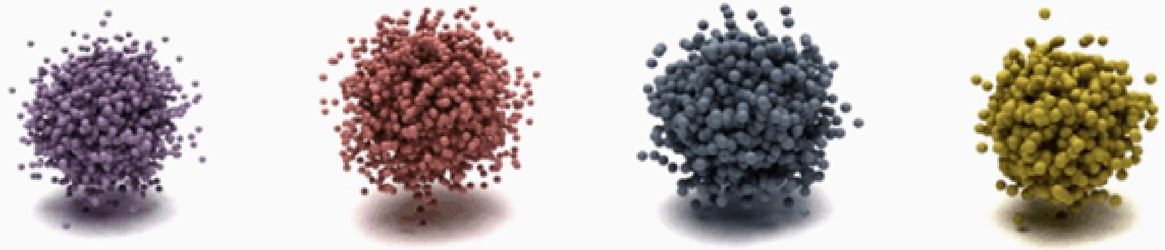
Contrastive pre-training in CLIP. The blue squares are the pairs for which we want to optimize the similarity. Image derived from https://github.com/openai/CLIP

Vision Language Modeling

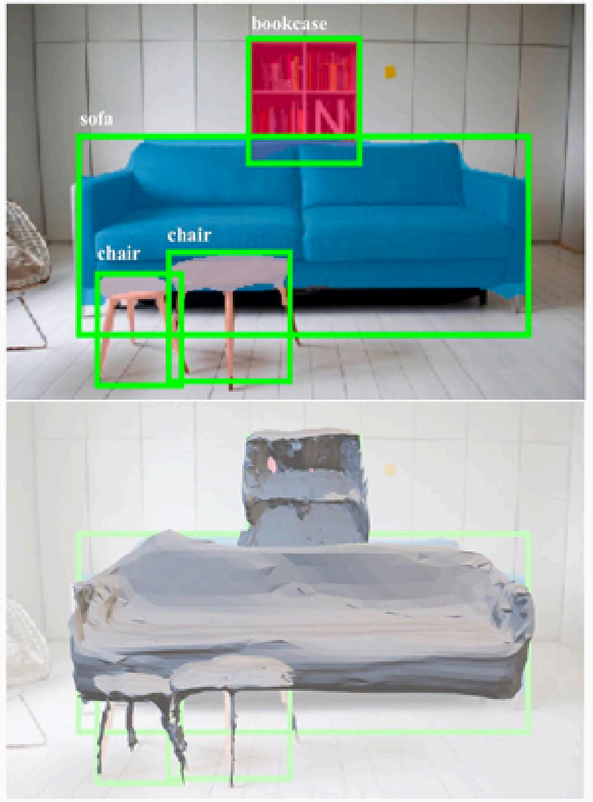
What Can We Do With Neural Networks?



Choy et al., 3D-R2N2: Recurrent Reconstruction Neural Network (2016)



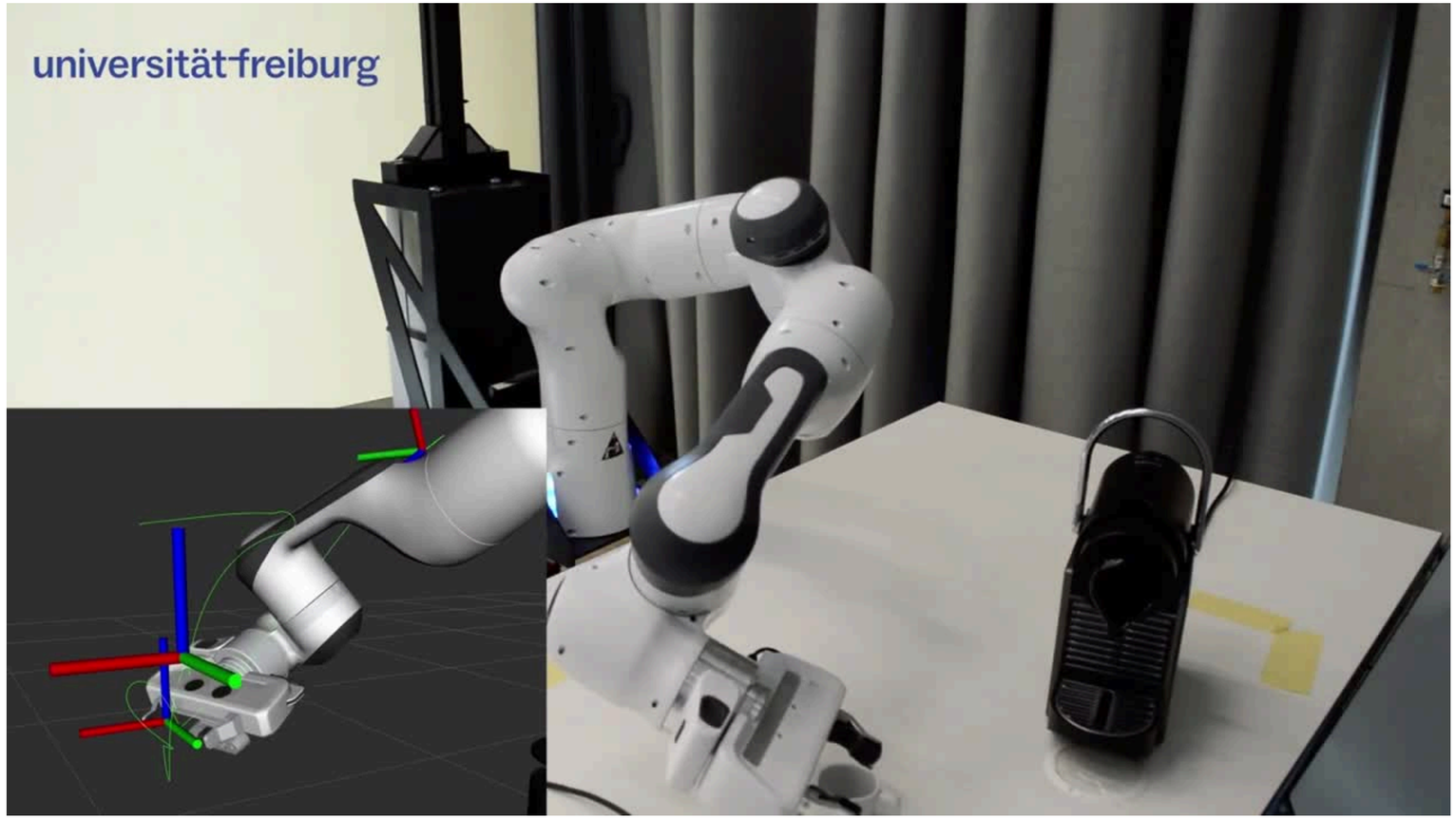
Zhou et al., 3D Shape Generation and Completion through Point-Voxel Diffusion (2021)



Gkioxari et al., “Mesh R-CNN”, ICCV 2019

3D Vision

What Can We Do With Neural Networks?

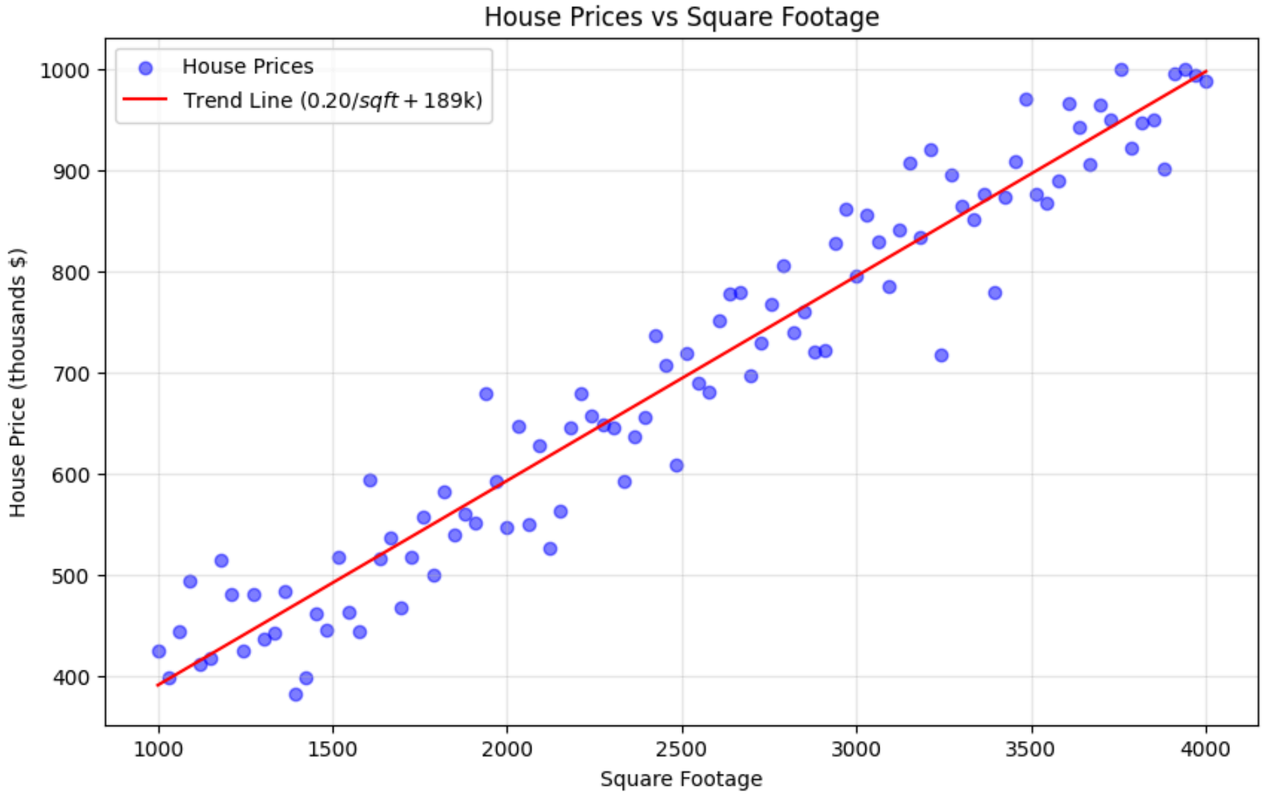


Multi-modal intelligence

Why Neural Networks ?

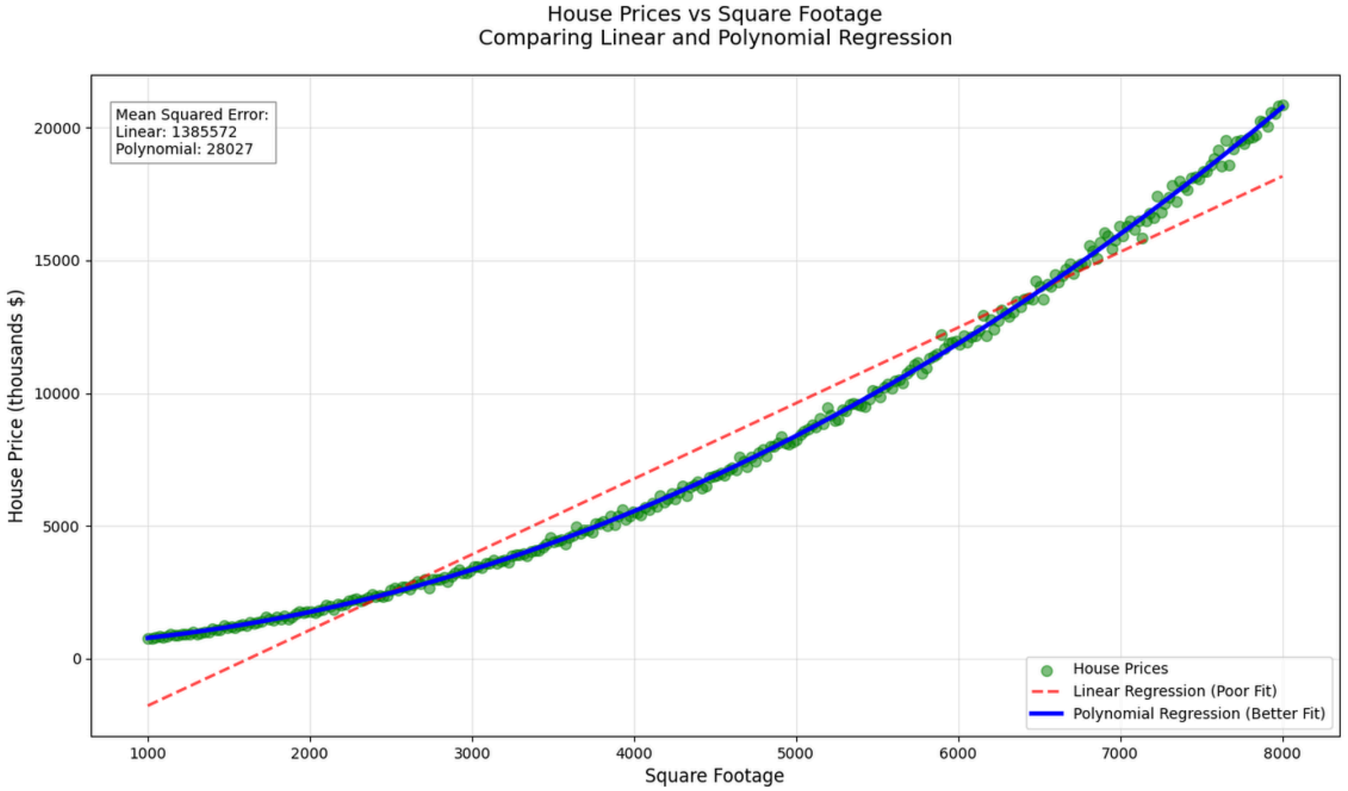
Linear regression

- Fits a straight line to data.
- It works well when the relationship between input and output is linear.



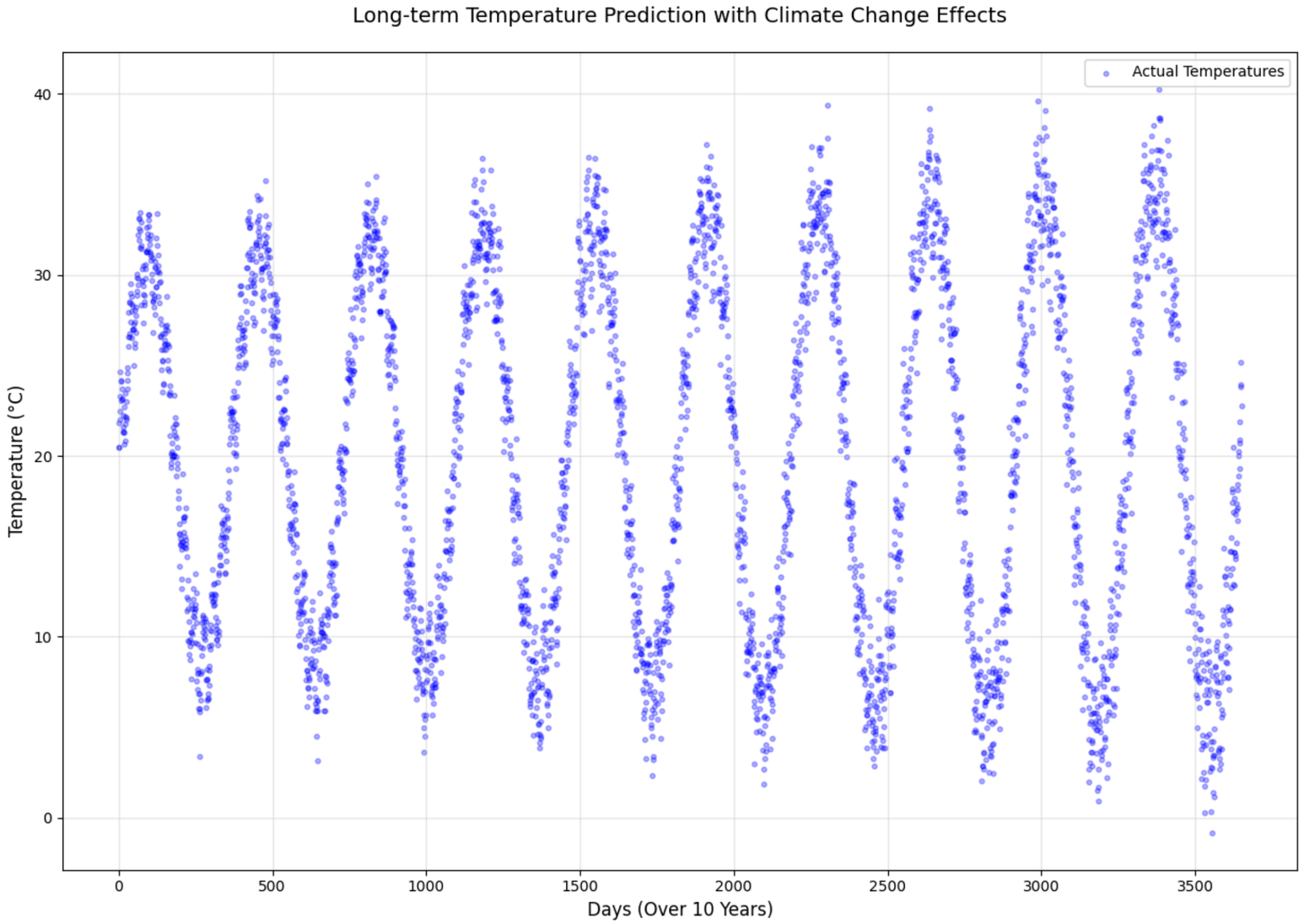
Polynomial Regression

- Fits data with a polynomial equation with degree greater than 1
- Works on more complex data.



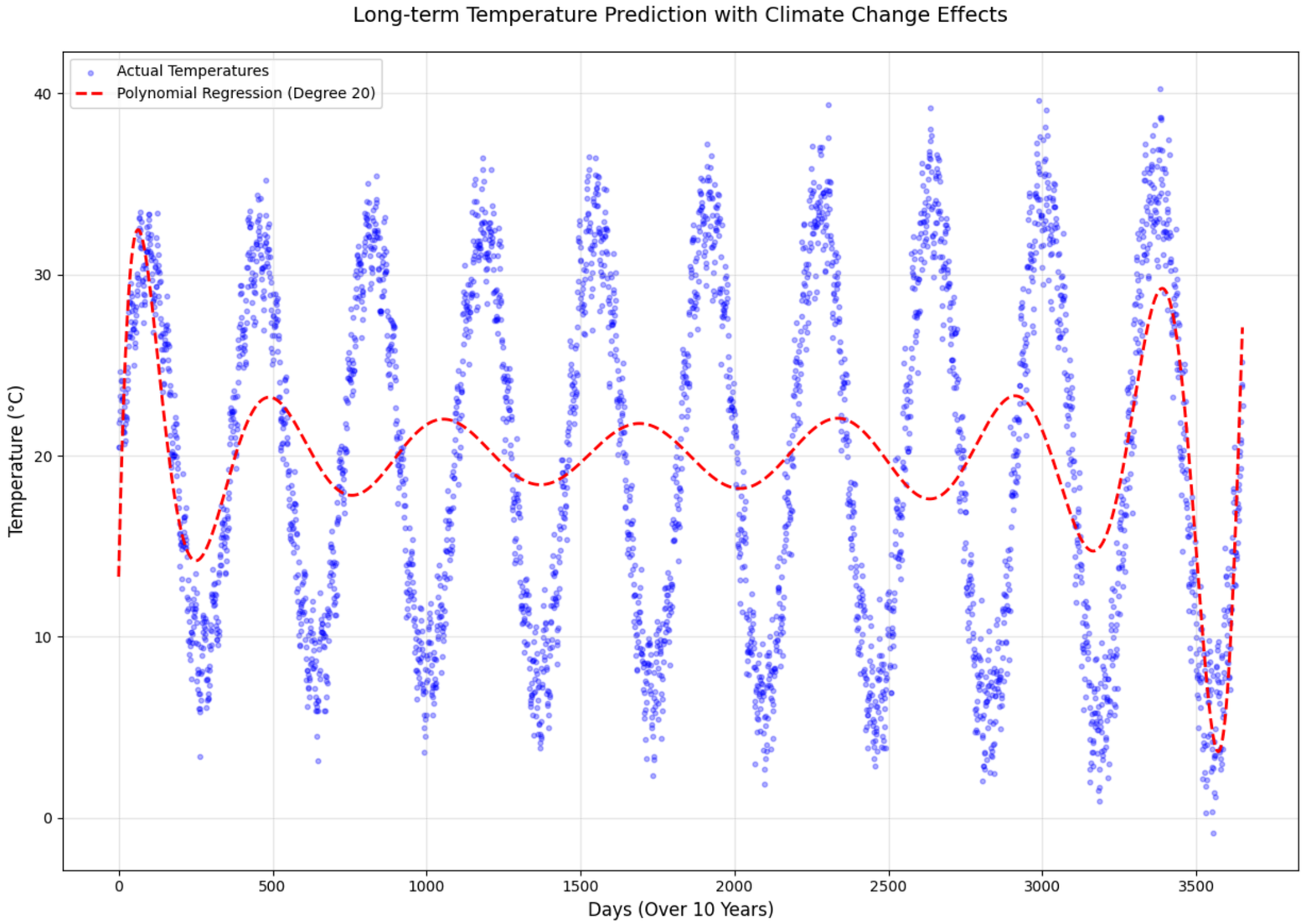
Can we fit this data?

- How about taking much more complex data?
- Can polynomial models fit the data?



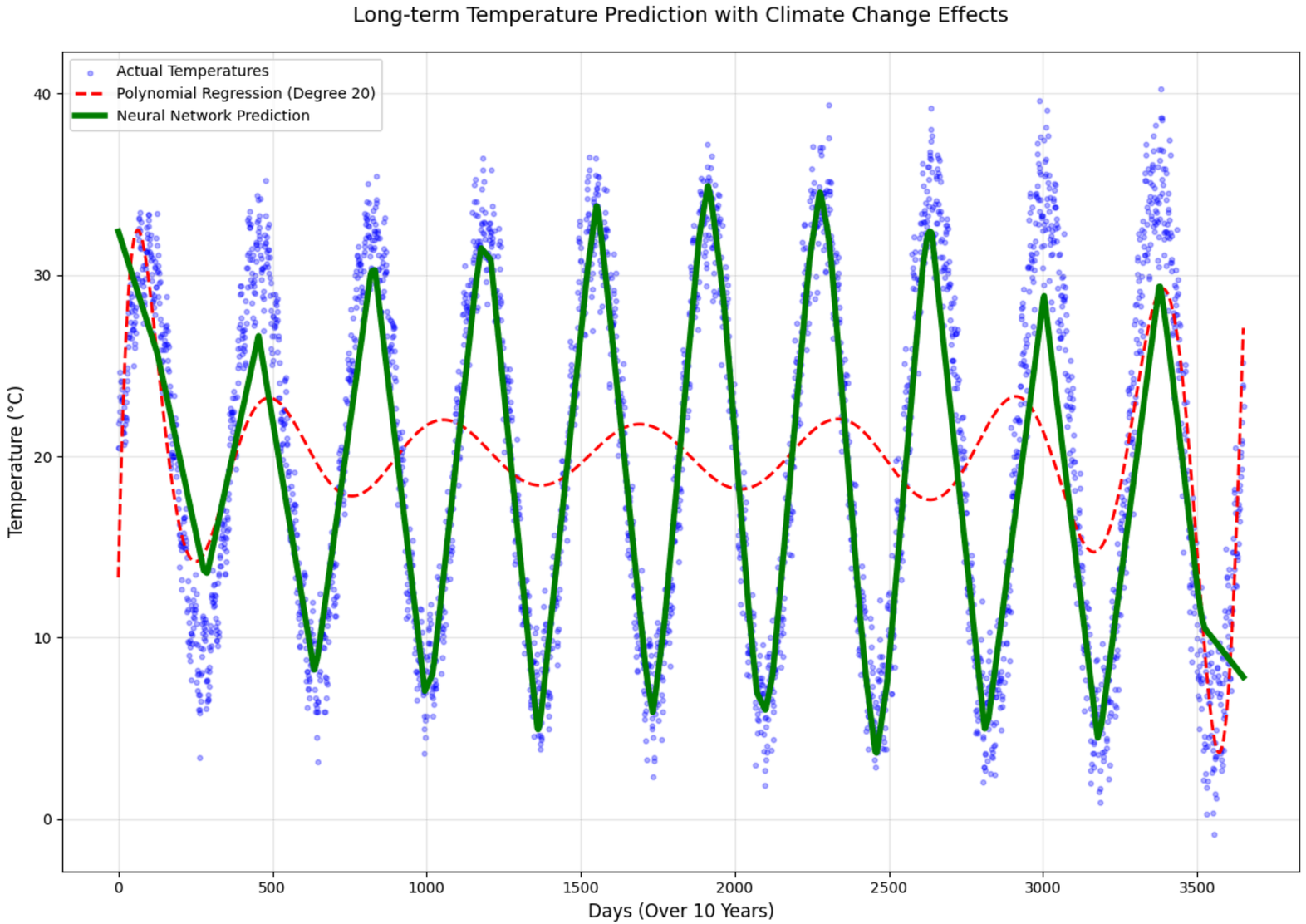
Can we fit this data?

- Let's try polynomial regression
- It seems like the model failed to fit the data



Can we fit this data?

- Let's try Neural Networks
- It appears that the neural network excels at adapting to this complex data!



Neural networks emerge as the champion.

- Neural networks can adapt to evolving patterns over time.
- They can automatically learn and model complex interactions between multiple factors

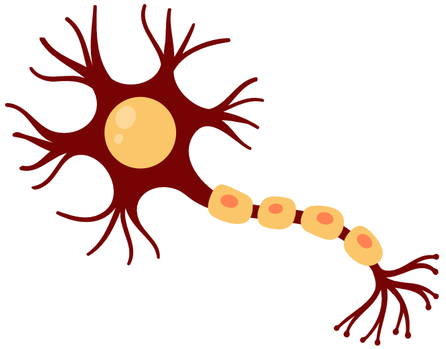
How Do Neural Networks Function?

A Single Neuron

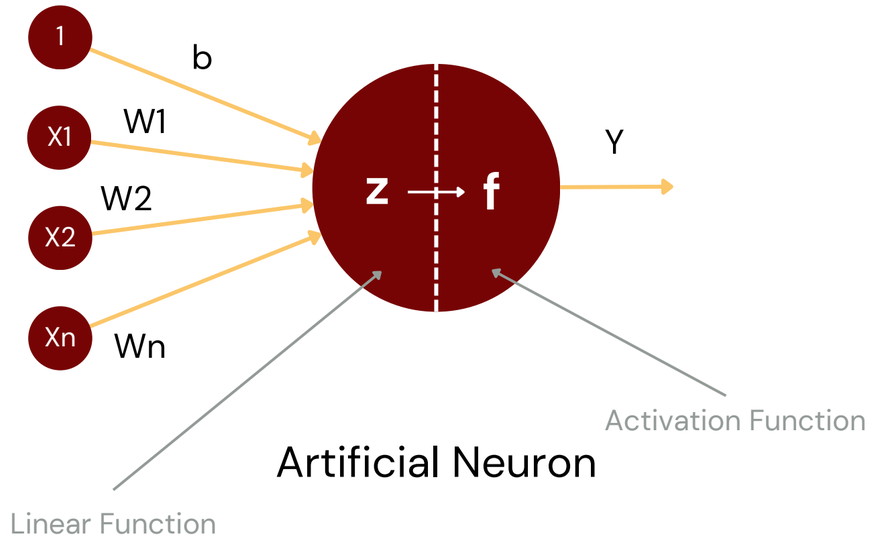
- A single artificial neuron is a mathematical model or function that takes an input, does a specific calculation based on weights and adds a bias

$$z = \sum_{i=1}^{n} w_i x_i + b$$

Mathematical formulation



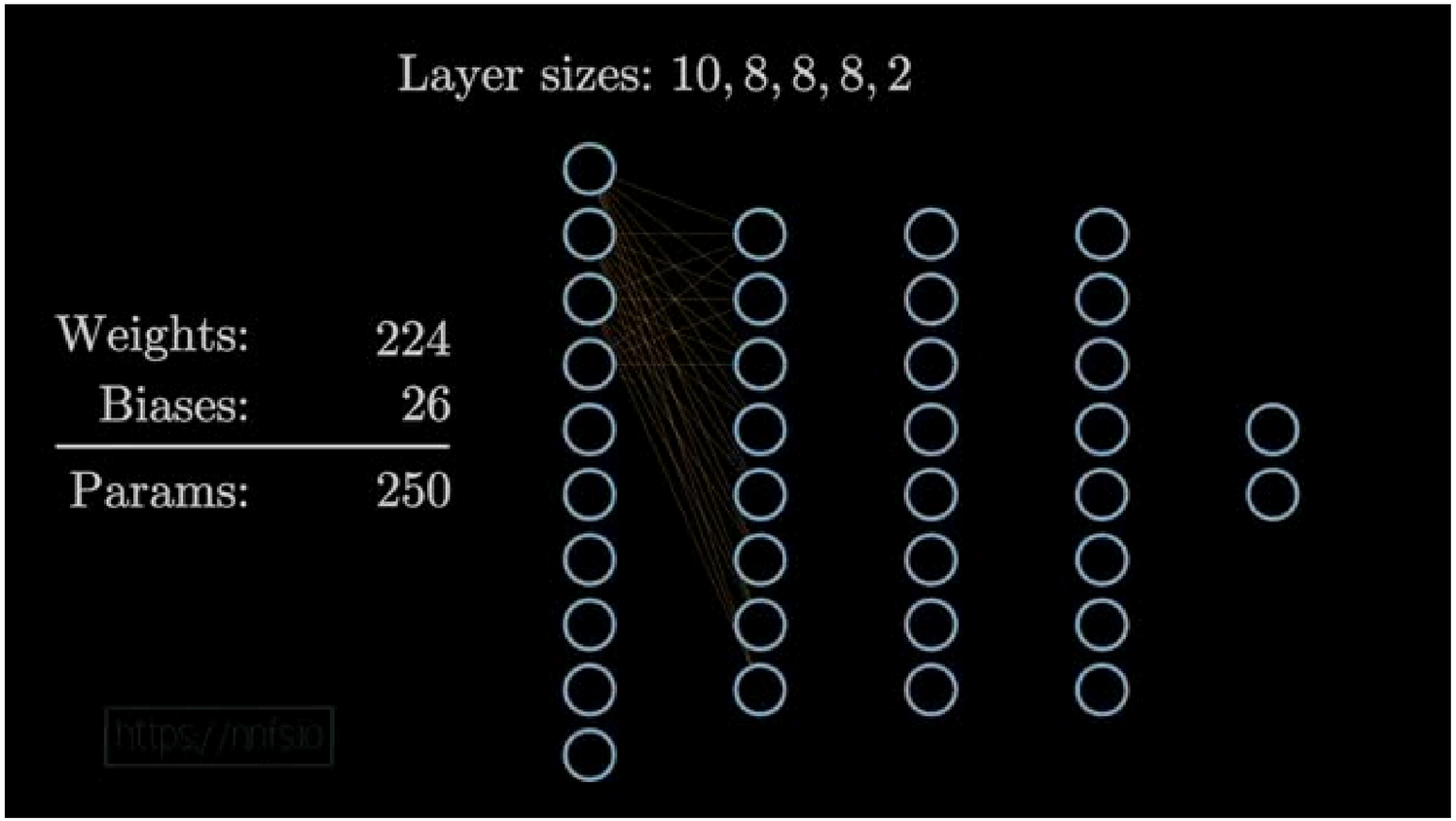
Biological Neuron



Artificial Neuron

Multiple Neurons

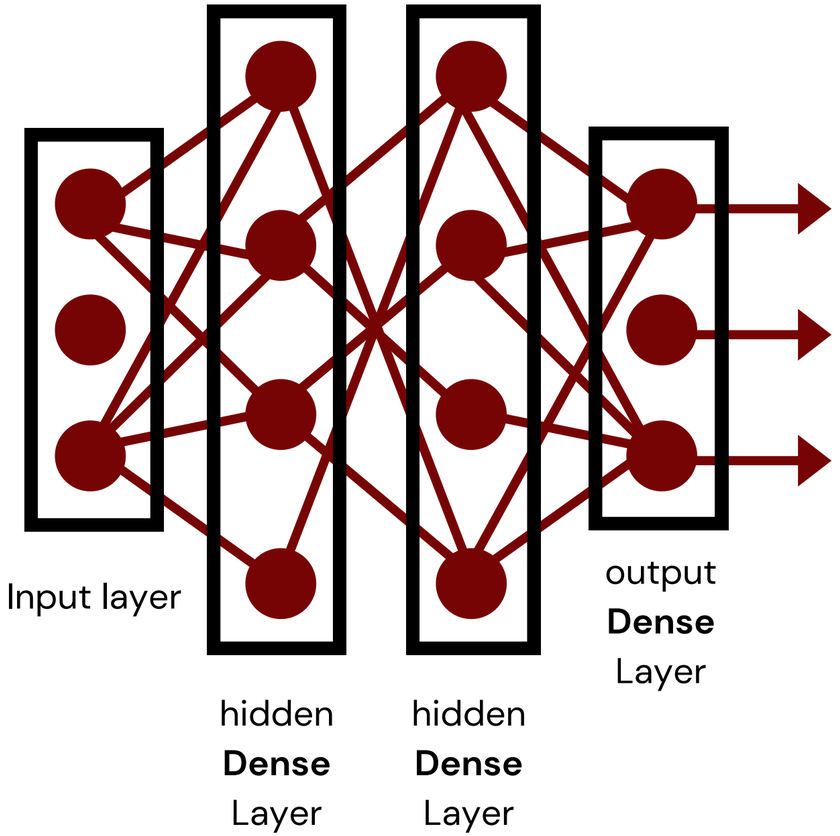
- A single neuron by itself is relatively useless.
- When interconnected, numerous neurons can produce results that often exceed other machine learning methods.
- Due to their complexity, neural networks are often seen as black boxes, as their decision-making processes are unclear.



First Technical terminology, "Dense Layers"

- In a dense layer, every neuron connects to all neurons in the following layer.
- Each connection between neurons has a weight associated with it, which is a **trainable parameter**.
- A bias is added to help fit the real-world types of dynamic data.

What allows a Neural Network to fit non-linear data is not just the network itself, but the **activation functions** used within it!

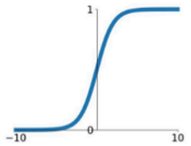


What is an **Activiation Function** ?

- Activation functions help a neural network learn and model complex patterns by introducing **non-linearity**.
- They enable the network to capture non-linear relationships in data, like **curves** and **complex shapes**.
- Without them, the network would behave like simple linear regression.

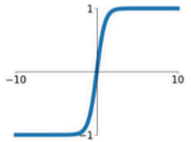
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



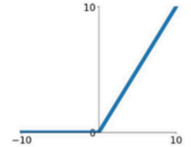
tanh

$$\tanh(x)$$



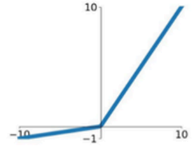
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

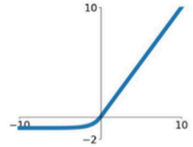


Maxout

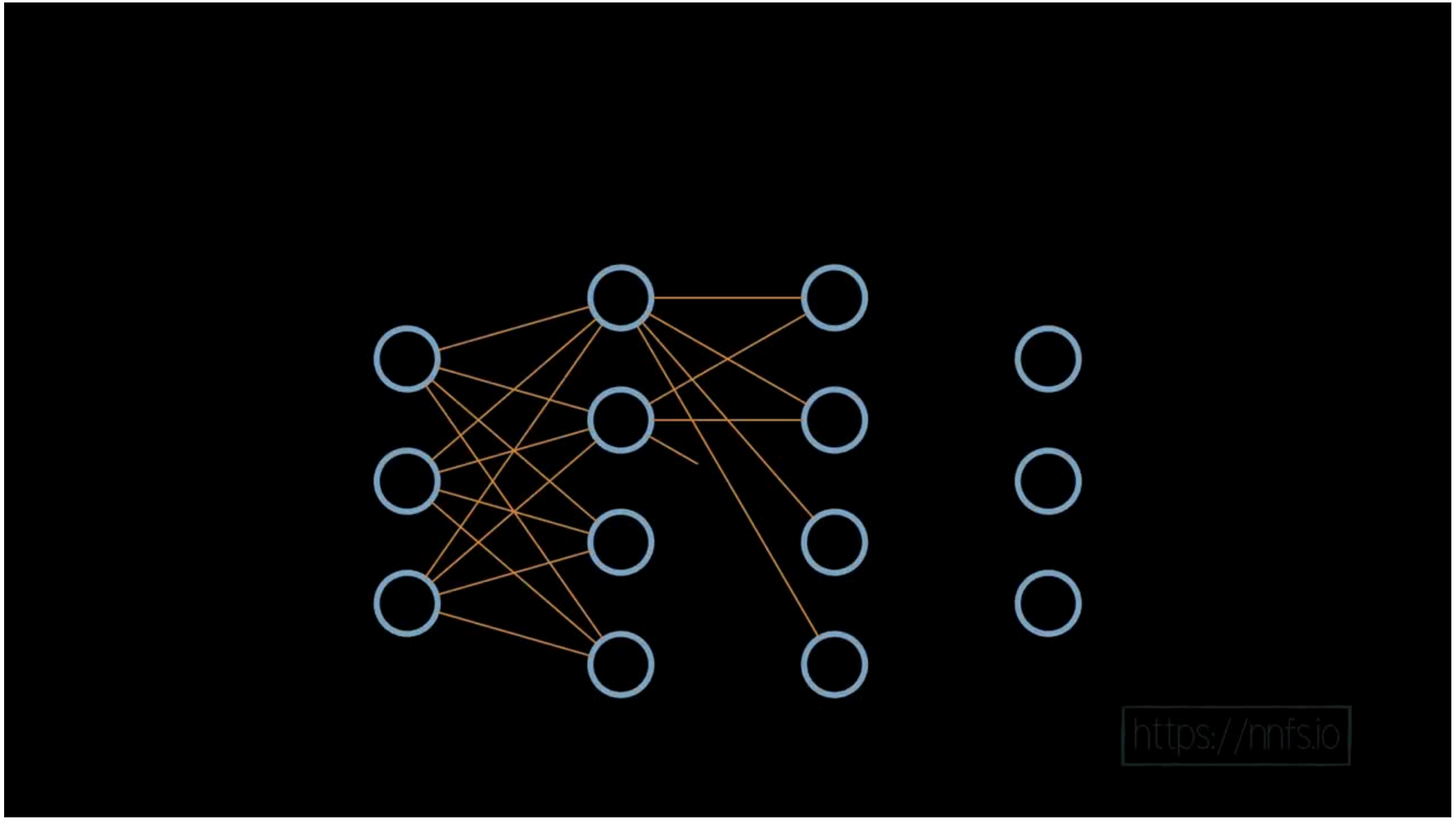
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

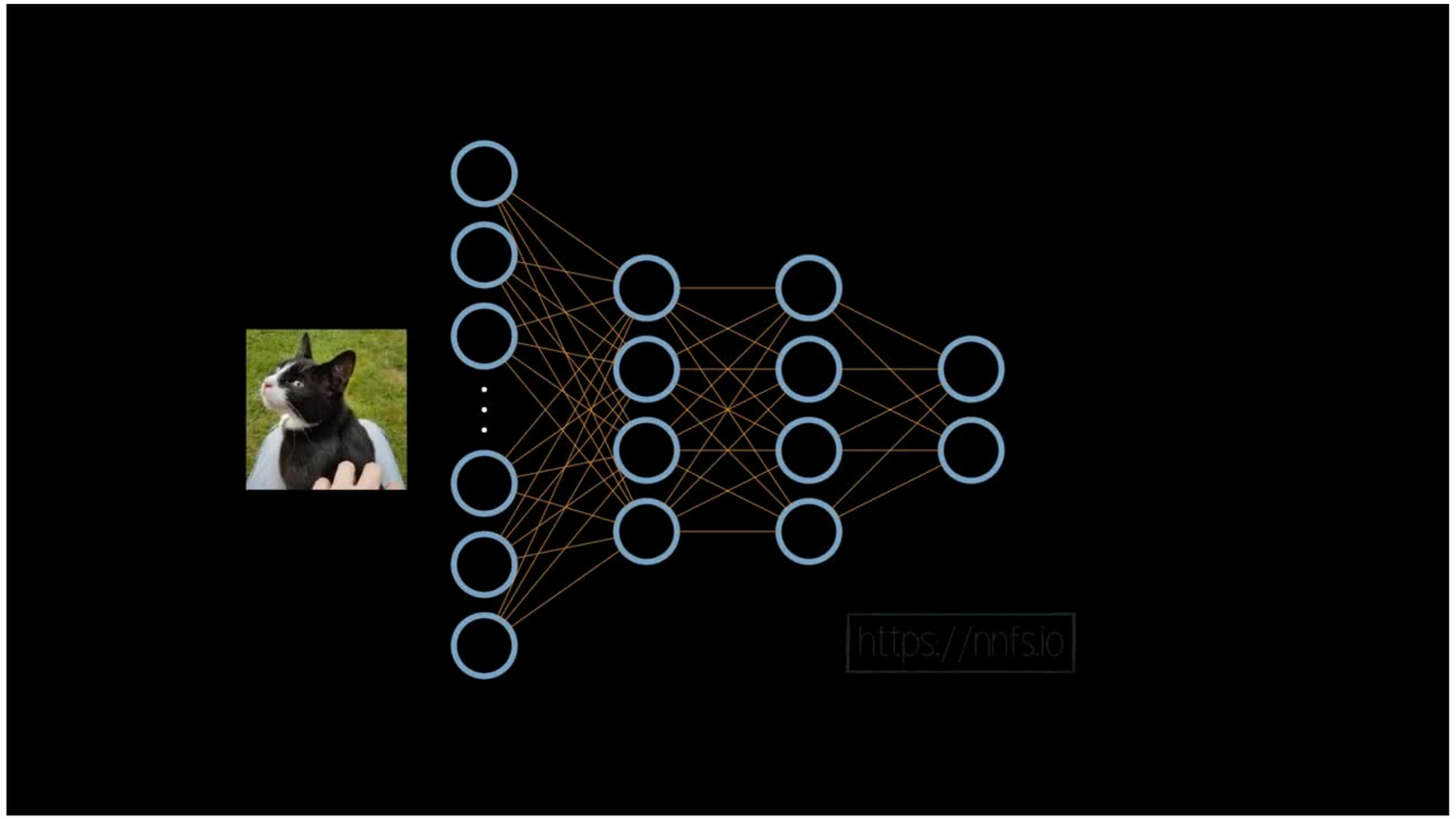


Weights, biases on a single neuron.



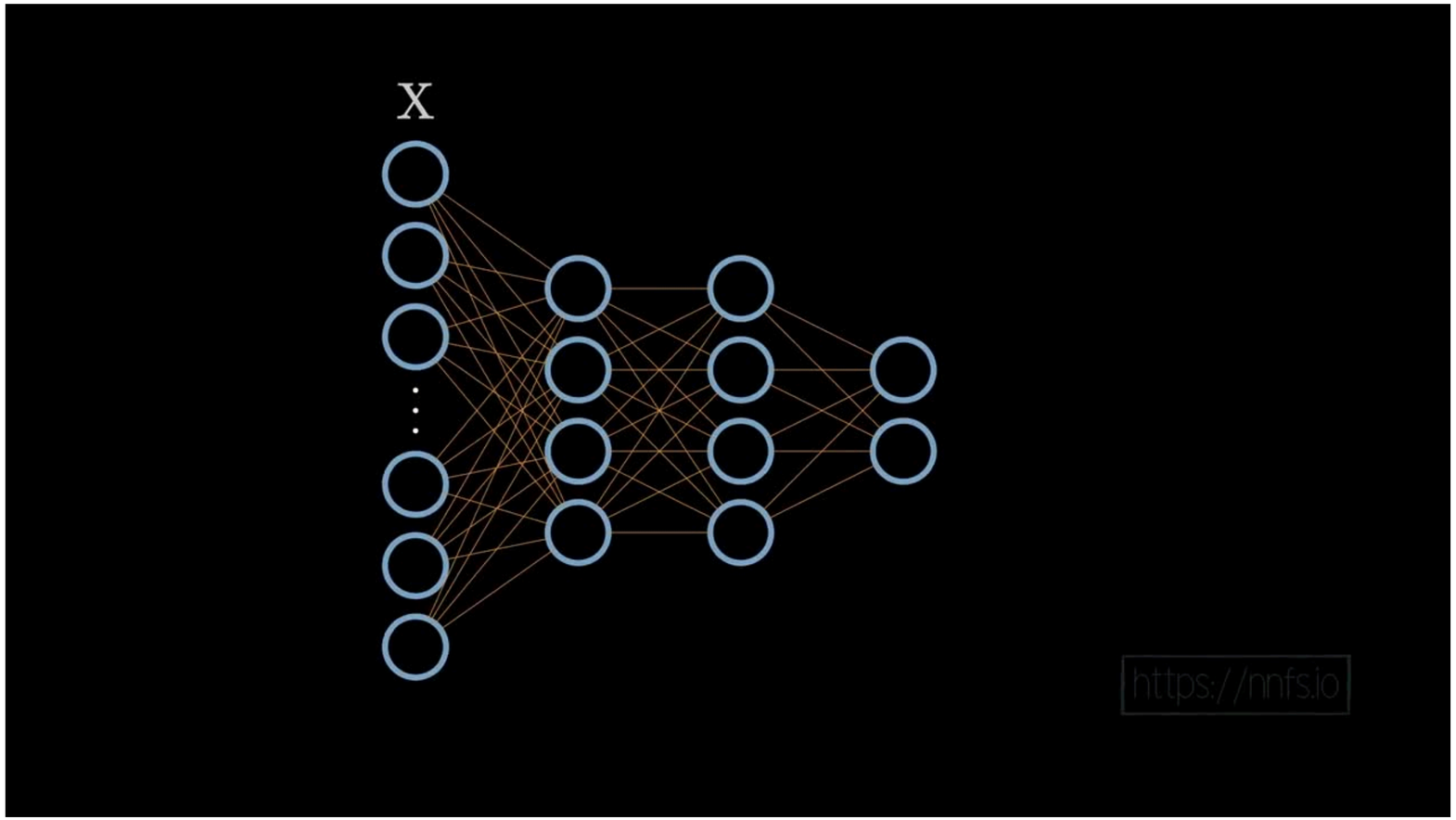
How do Neural Networks perform classification?

- Input Layer: Holds the input data (image pixels)
- Data is often preprocessed (scaled/normalized) to fit a range like 0–1 or -1–1.
- Each neuron will procude on output based on input.



Lots of mathematics is behind these networks.

- Neural networks are computationally intensive models!
- The process of transforming the input into a label or a number is called **forward propagation** or **model inference.**



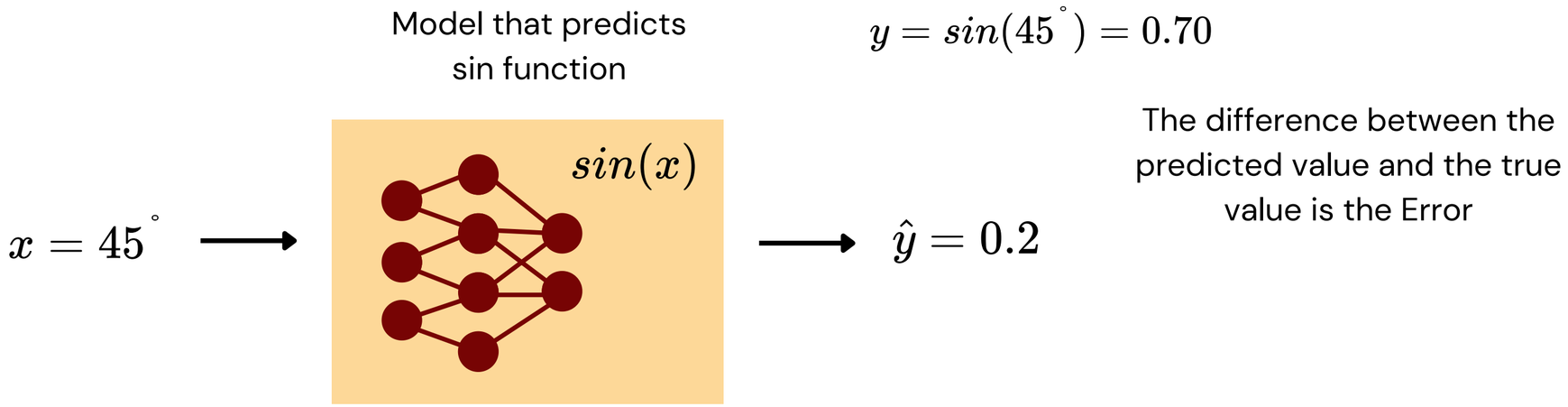
Let's implement a Neural Network

Scan the Drive

How Do Neural Networks Learn?

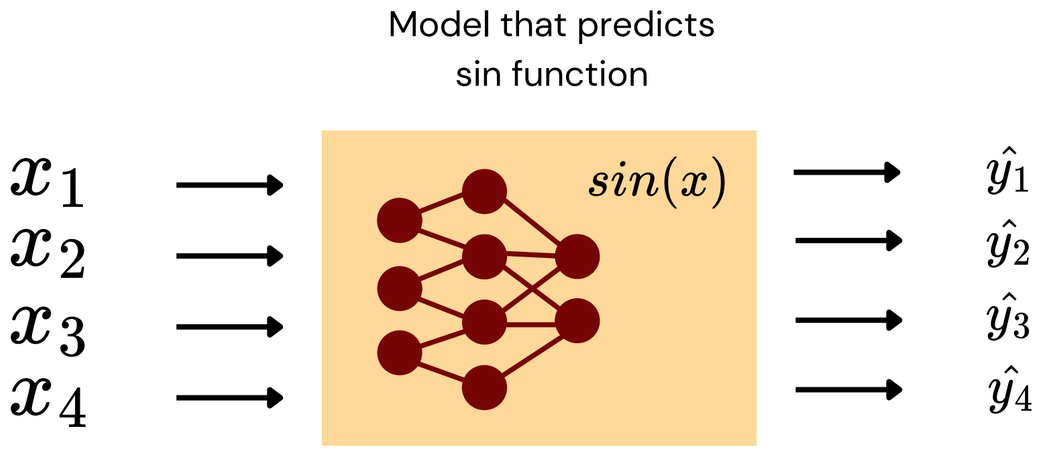
Calculating Network Error with Loss

- The loss measures how far the neural network's prediction is from the actual correct answer for one example.
  - It's like a "score" for how bad the prediction was for that single input.
- The error is assessed using various methods and formulas that depend on the nature of the problem.



The Cost Function

- The cost function is the average of all the losses across all examples in the dataset.
- It gives an overall "score" for how well the network is doing on the entire dataset.
- The goal of training is to minimize this cost



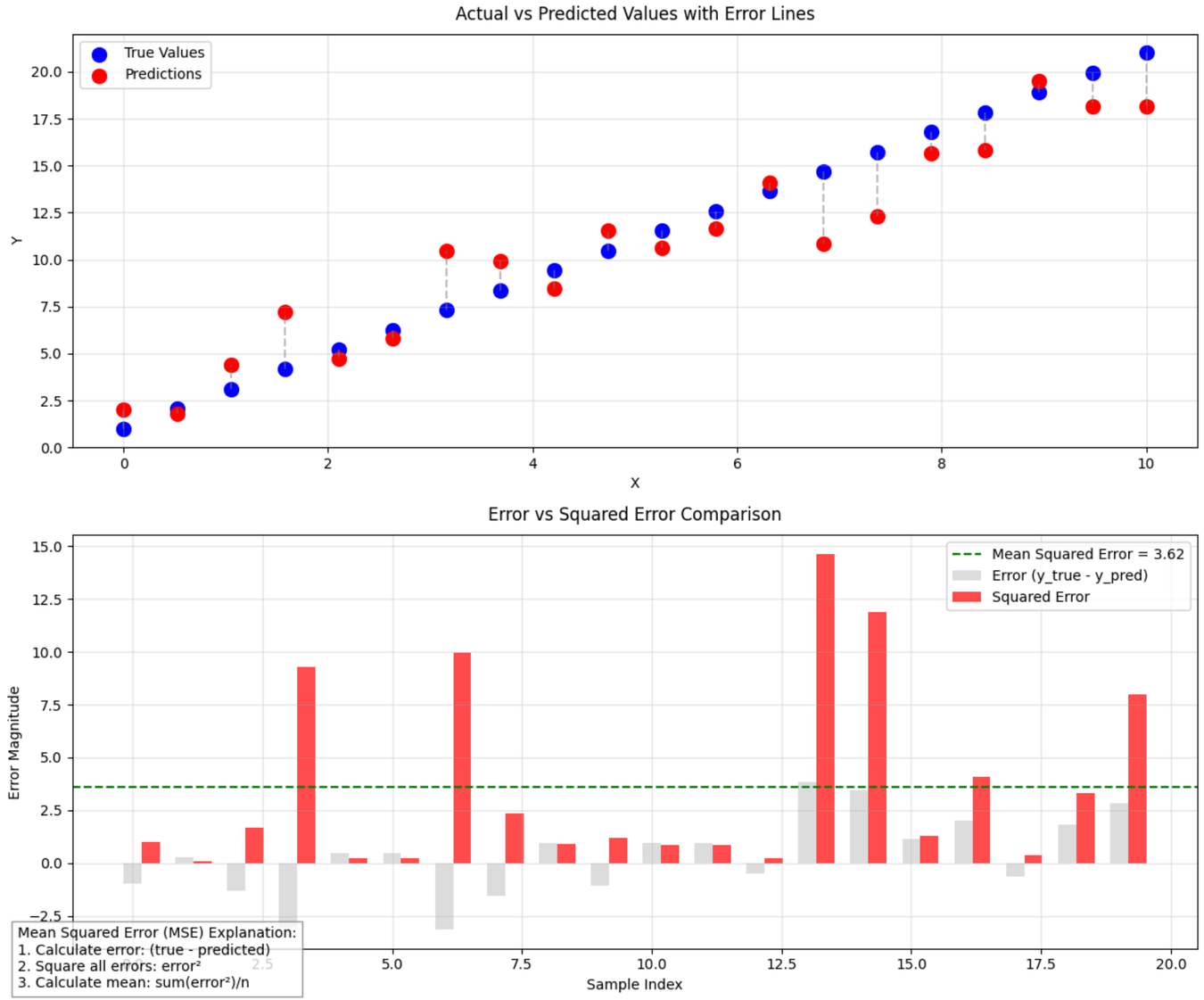
The cost is the average of losses

$$Cost = \frac{1}{n}\sum_{i=1}^{n} Loss_i$$

Mean Squared Error (MSE)

- Measures the average squared difference between predicted and actual values.
- Use Case: Predicting numbers, e.g., house prices.
- Here is its mathematical formula:

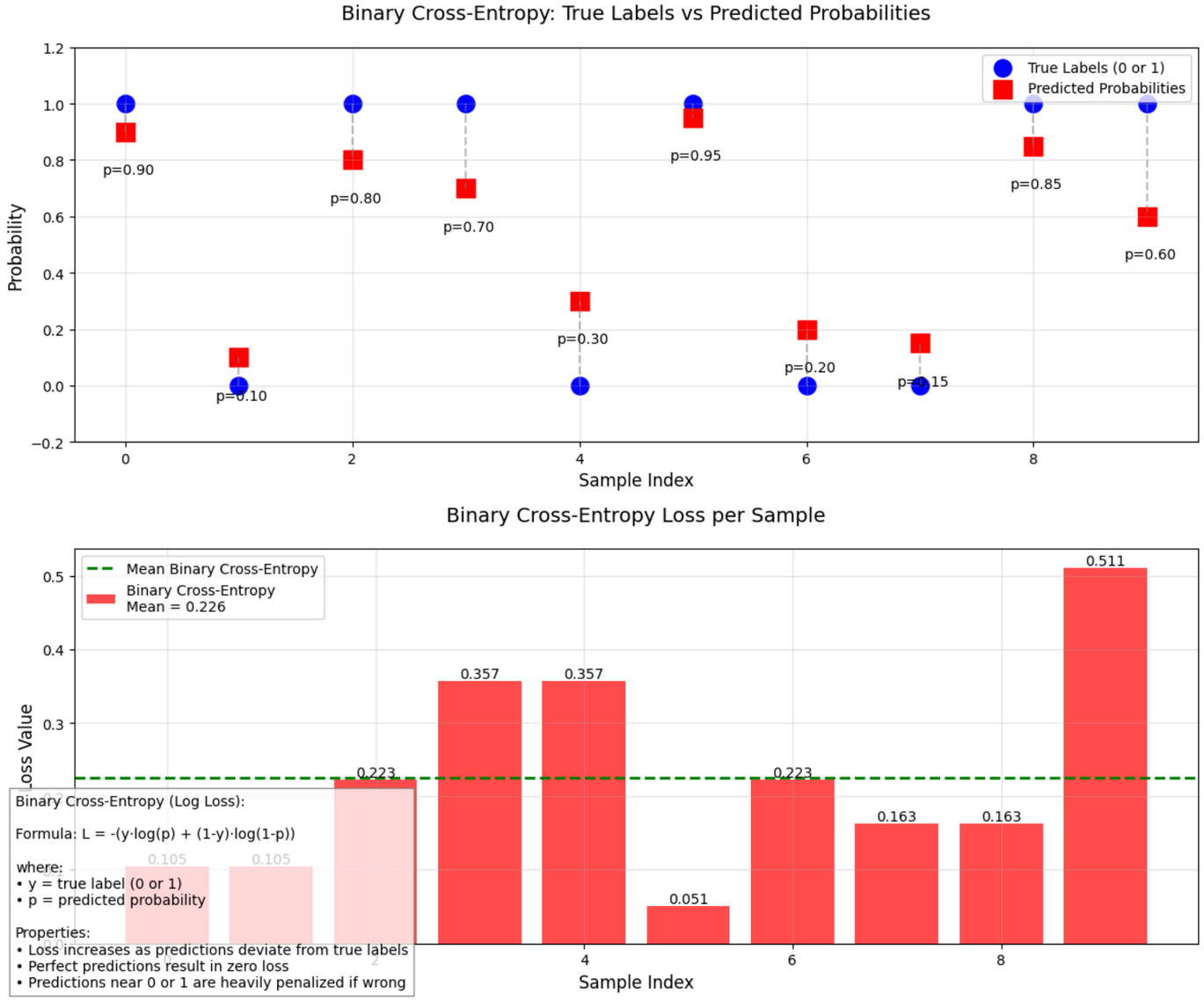
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$$



Binary Cross-Entropy (Log Loss)

- Used for binary classification tasks, where the output is either 0 or 1 (e.g., "dog" vs. "not dog").
- The predicted output is treated as a probability (between 0 and 1) of belonging to class 1
- The loss penalizes incorrect predictions more heavily when the model is very confident but wrong.

$$BCE = \frac{1}{n}\sum_{i=1}^{n}[y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)]$$

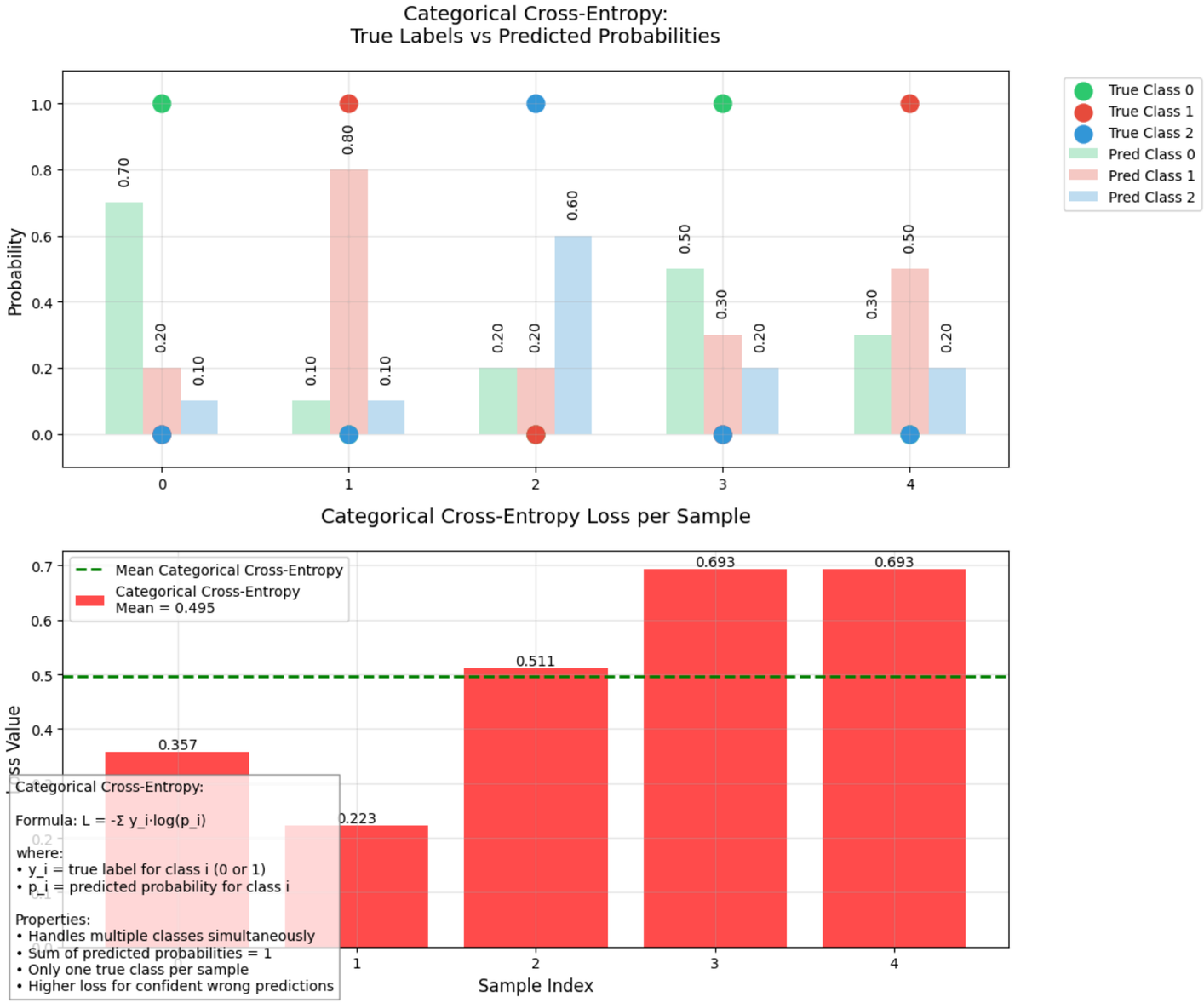


Categorical Cross-Entropy

- Used for multi-class classification tasks, where the output can belong to one of **k** classes (e.g., dog, cat, bird).
- Each output neuron corresponds to a class.
- For each example, only the predicted probability of the correct class contributes to the loss

$$CCE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij} log(\hat{y_{ij}})$$

$$\hat{y_{ij}} = \frac{e^{z_j}}{\sum_{l=1}^{k} e^{z_l}}$$



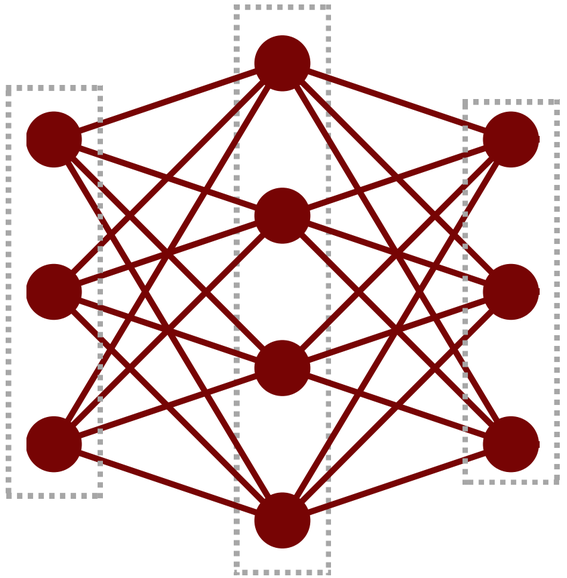
Let's implement a Cost function

scan the notebook

Where the magic unfolds: Backpropagation

- Backpropagation is a process that helps a neural network learn by adjusting the weights and biases in the right direction to reduce the error (loss).
- The goal of backpropagation is to minimize this error by tweaking the weights and biases.

Step 1: Forward Propagation



$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$$

Step 2: Calculate the Loss

Which means determine How wrong the model from the real value

Step 3: Calculate the Derivative of the Error (Gradient)

For each Layer Calculate the Derivative of each output, this will produce the gradient

Step 4: Update the Weights and Biases Using Gradient Descent

$$W_{new} = W_{old} - \alpha \nabla J(W)$$

Repeat Until Convergence

Backpropagation

Forward Pass

- The input data passes through the network, layer by layer.

Calculate the Error

- Using MSE, compute how far the network's prediction is from the actual value

Backward Pass

- Starting with the output layer, we calculate **how the error changes with respect to the predicted value**

$$\frac{\partial Error}{\partial \hat{y}} = \hat{y} - y$$

- Work backward through the layers; for each weight in the network, calculate how much it contributed to the error using the **chain rule**

$$\frac{\partial Error}{\partial w} = \frac{\partial Error}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w}$$

from the previous step

Derivative of the activation function

How the weight affects the weighted sum

- Update weights by using the gradient.

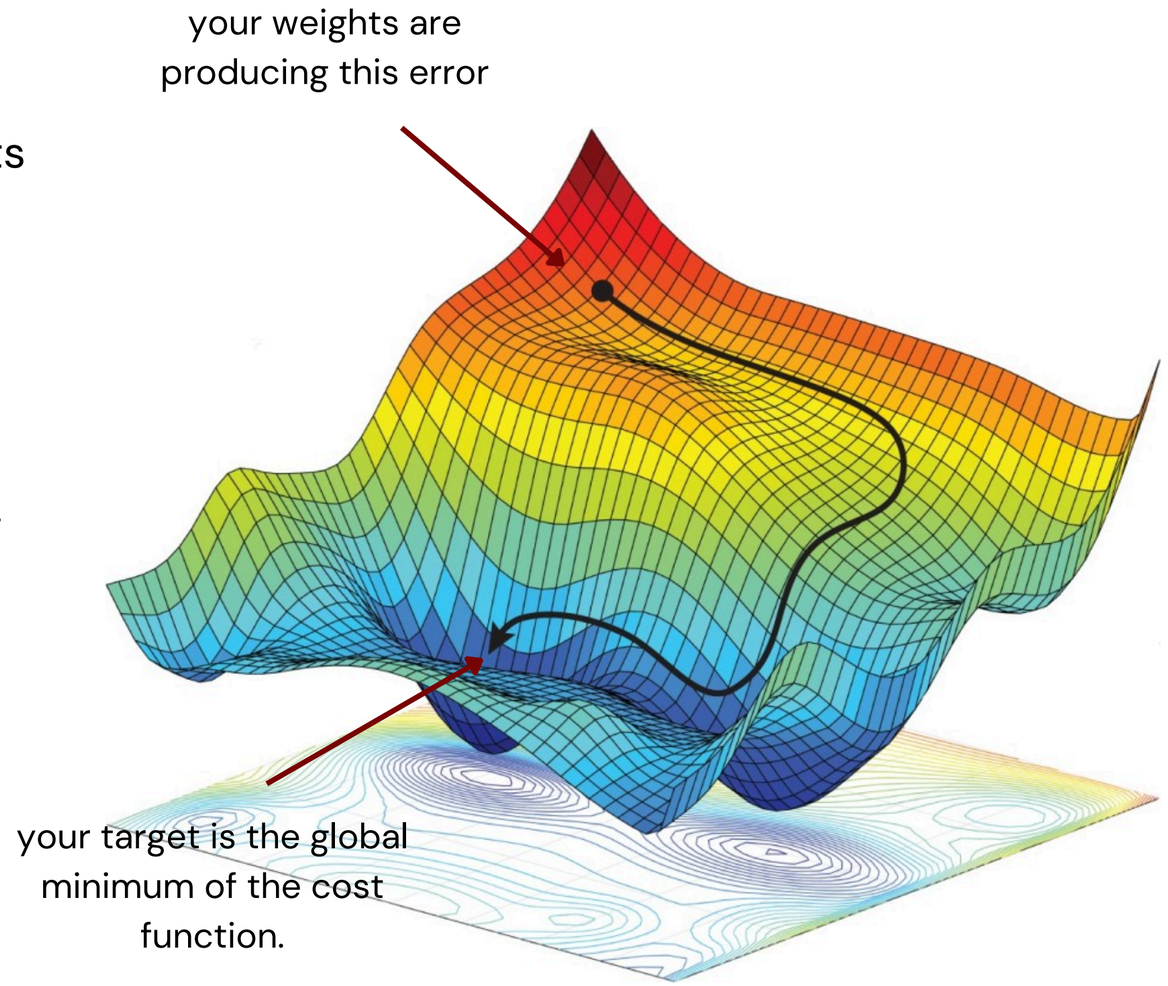
$$w_{new} = w_{old} - \alpha \cdot \frac{\partial Error}{\partial w}$$

The alpha is a hyperparameter called learning rate.

Let's implement a Back Propagation

Introducing Optimization

- Optimization is the process of adjusting the weights and biases of a neural network so that it makes better predictions.
- After calculating the error (loss) during training, optimization algorithms decide how much to change the weights and biases to reduce the error.
- This process uses the gradients calculated during backpropagation.
  - Optimizers are algorithms or methods used to update the weights and biases in a neural network.



Optimizers

Gradient Descent

- Updates weights by moving in the direction of the negative gradient of the loss function.

Stochastic Gradient Descent (SGD)

- Instead of using all the training data to compute the gradient, it uses one sample at a time.
- Faster but noisier updates.

Mini-Batch Gradient Descent

- Combines both: it computes gradients using small batches of data instead of the entire dataset or just one sample.
- Balances speed and stability.

Adam (Adaptive Moment Estimation)

- Most popular optimizer.
- Combines the ideas of Momentum and another method called RMSProp.
- Automatically adjusts the learning rate for each parameter based on the gradient's history.

Equation 11-9. Adam algorithm

1. $\mathbf{m} \leftarrow \beta_1 \mathbf{m} - (1 - \beta_1) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
2. $\mathbf{s} \leftarrow \beta_2 \mathbf{s} + (1 - \beta_2) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \otimes \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
3. $\widehat{\mathbf{m}} \leftarrow \dfrac{\mathbf{m}}{1 - \beta_1^t}$
4. $\widehat{\mathbf{s}} \leftarrow \dfrac{\mathbf{s}}{1 - \beta_2^t}$
5. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \widehat{\mathbf{m}} \oslash \sqrt{\widehat{\mathbf{s}} + \varepsilon}$

Let's implement an Optimizer

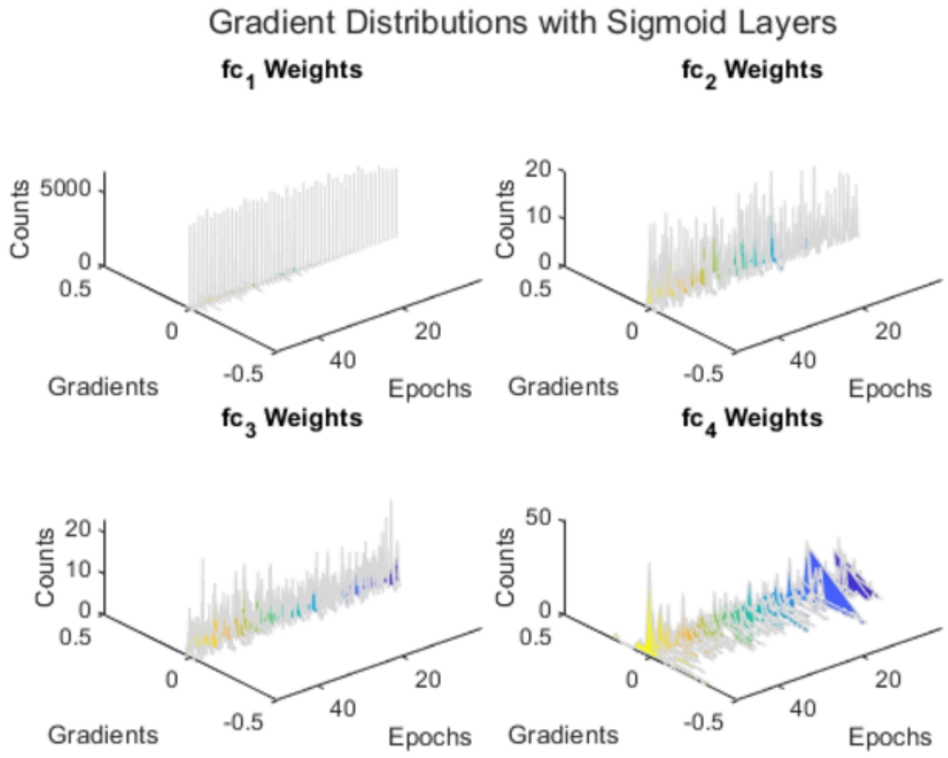
Training Deep Neural Networks

Training deeper neural networks (with many layers and connections) introduces several problems

- Gradients may shrink (vanish) or grow (explode) during backpropagation, making training difficult.
- Not enough training data or costly labeling can be an issue.
- Large models risk overfitting, especially with noisy or limited data.

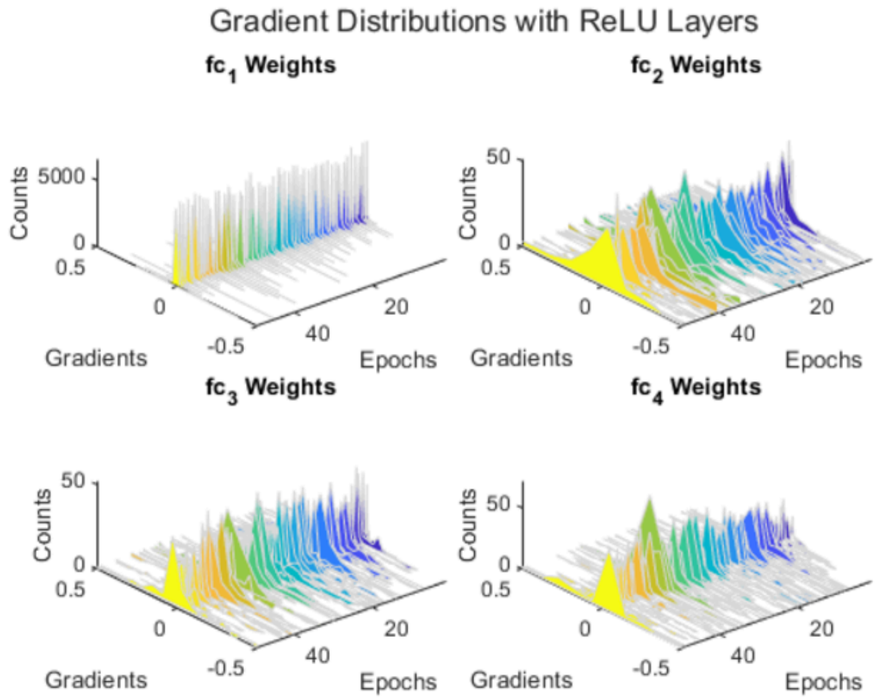
The Vanishing Gradients Problem

The vanishing gradients problem arises when gradients become very small as they move backward through layers during training, hindering effective learning in the network.



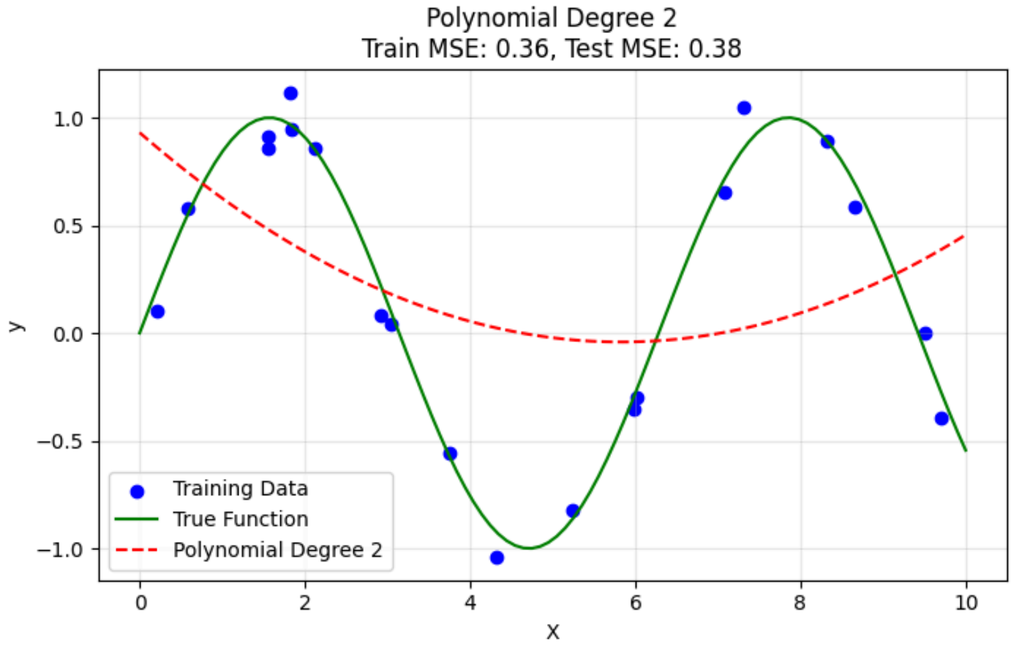
The Vanishing Gradients Problem

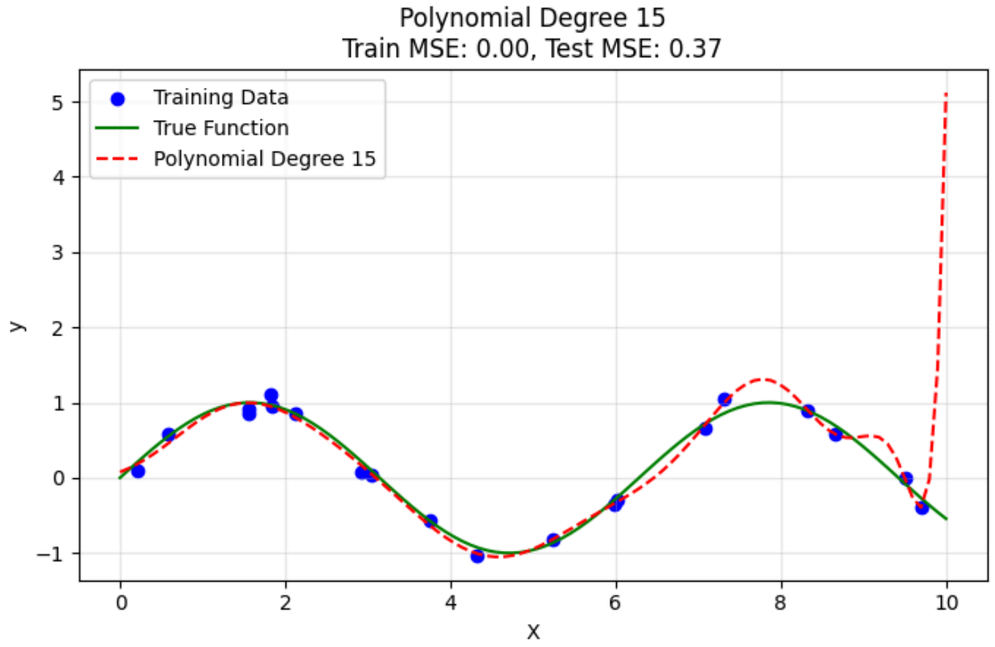
- Use activation functions like ReLU (Rectified Linear Unit), which doesn't squash values, allowing gradients to flow better.
- Use techniques like batch normalization or proper weight initialization to keep the gradients healthy during training.



Overfitting Problem

- Overfitting occurs when a model learns training data too well, failing to generalize to new, unseen data.
- Performs very well on the training data but poorly on the test or validation data.





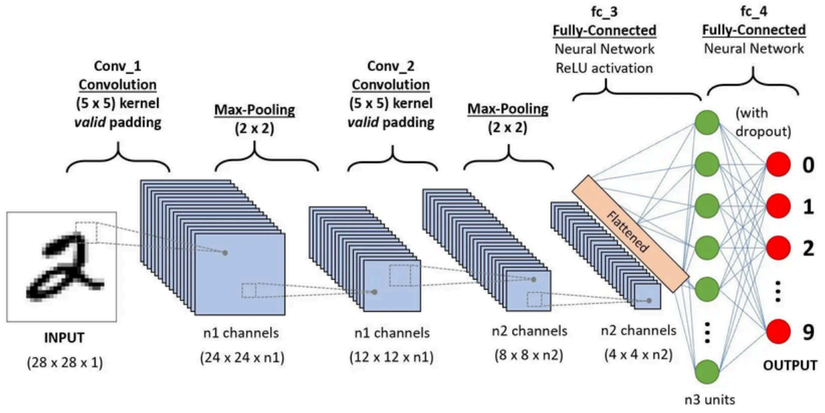
Overfitting Problem

Solutions

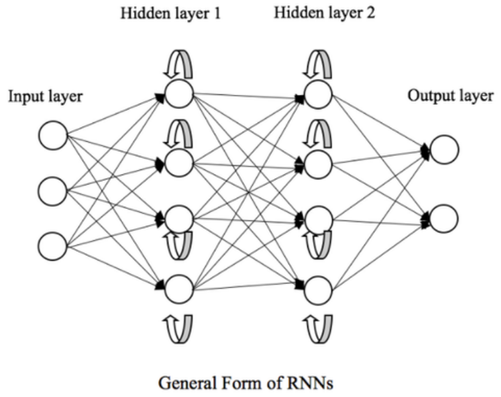
- More data can help the model learn general patterns and reduce the chance of memorizing specific details or noise.
- Use a smaller model (fewer layers or neurons) to prevent the network from learning irrelevant details.
- Stop training early before the model starts to overfit.
- During training, randomly "drop" (set to zero) some of the neurons in each layer.
- Use cross-validation to evaluate the model on various data sets, helping to identify overfitting early.

Starting A Real Deep Learning Project

The world of deep learning is vast!



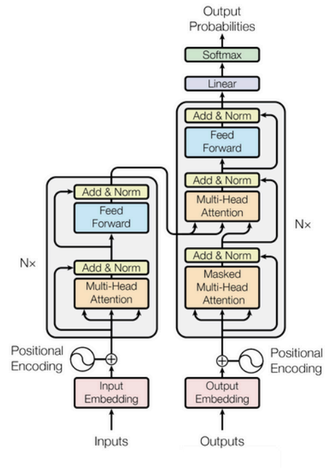
Convolutional Neural Networks



Recurrent Neural Networks

BERT

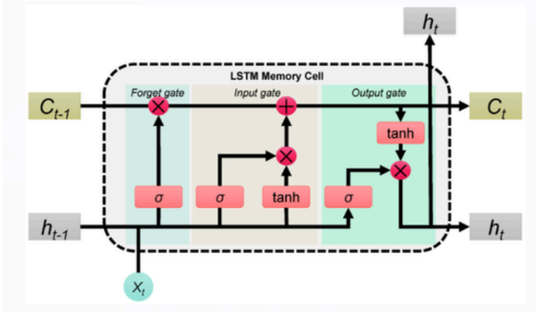
Encoder



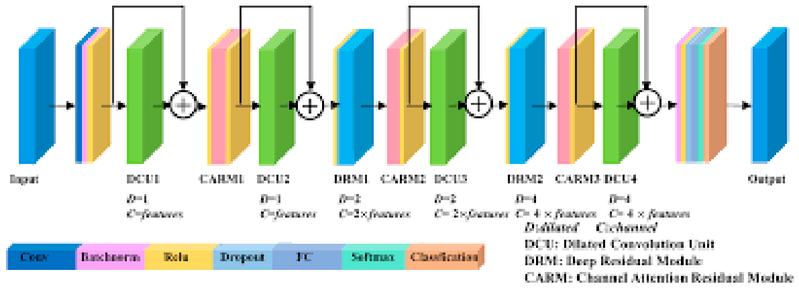
GPT

Decoder

Transformer Architecture



LSTM's



Residual Neural Networks

And much more....

Dive deeper

Stanford CS229 Machine Learning

+

Andre NJ Coursera Machine Learning Specialization

→

Stanford CS230 Deep Learning

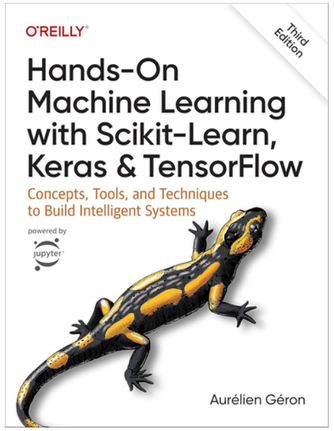
+

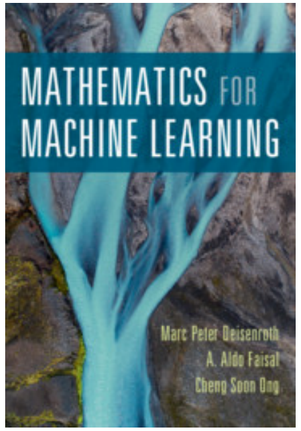
Andre NJ Coursera Deep Learning Specialization

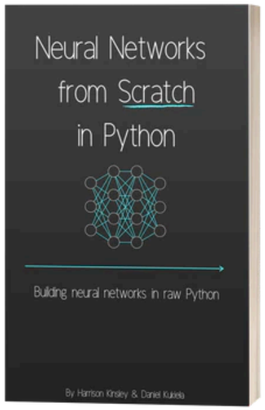
→

Stanford CS231n Computer Vision

Stanford CS224n Natural Language Processing with Deep Learning







The internet

Thank you