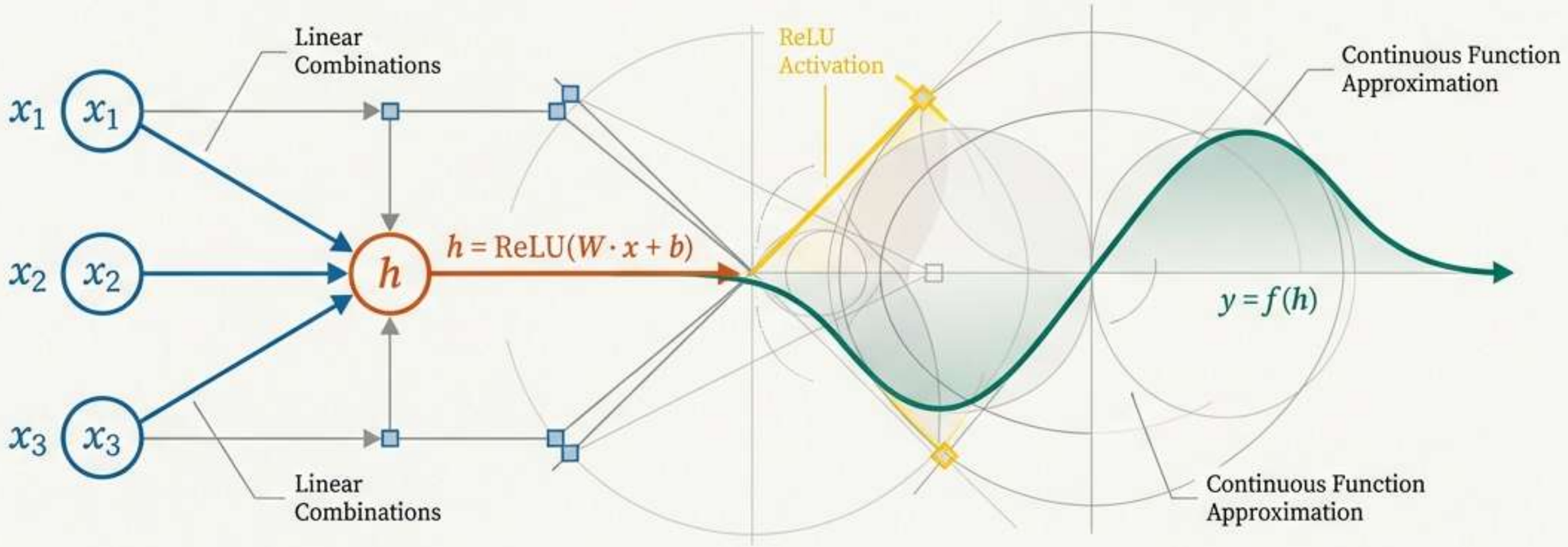
Building Intelligence: A First-Principles Guide to Shallow Neural Networks

From Simple Lines to Complex Functions



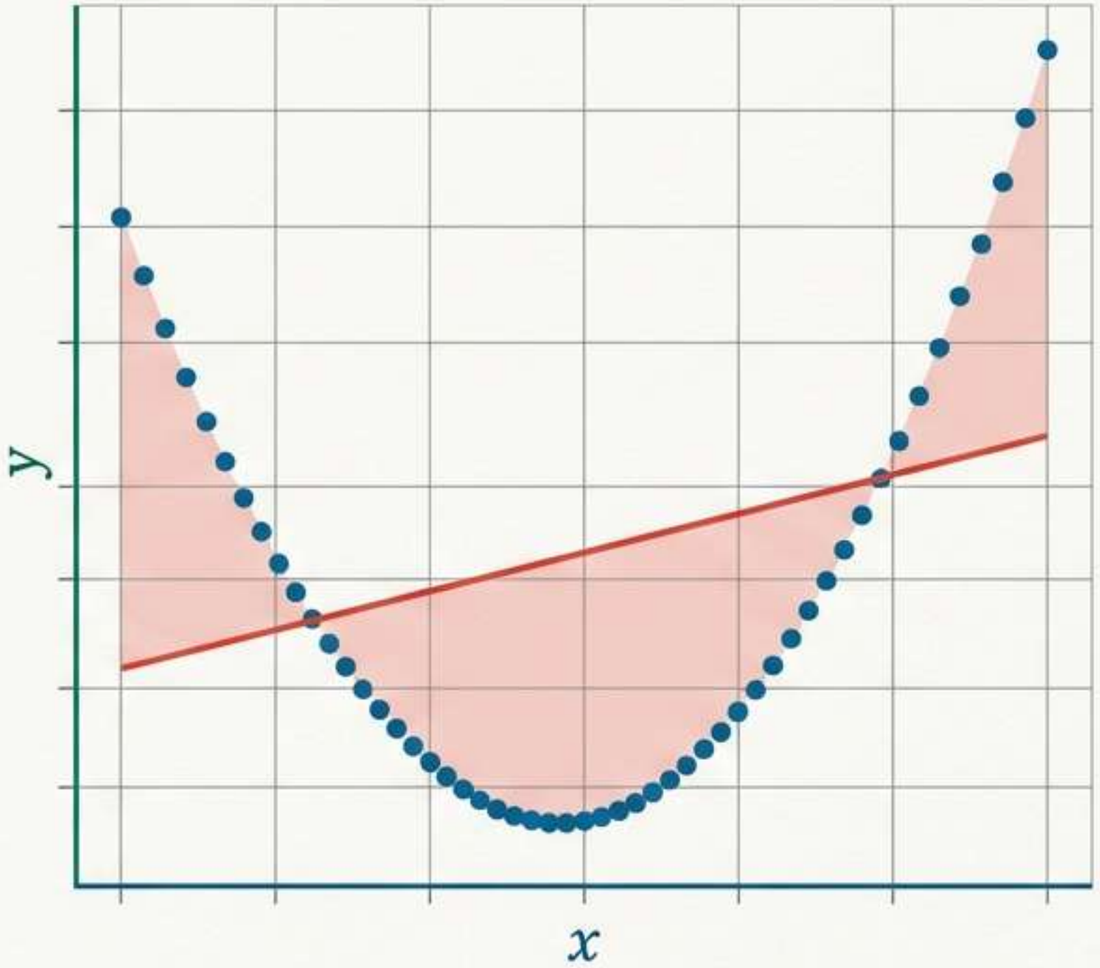
This presentation deconstructs how neural networks are built, not just what they are. We will assemble a powerful model from the simplest possible components.

We Begin with a Familiar Limit: A Single Line Isn't Enough

Linear regression, a foundational tool, models the relationship between an input x and an output y.

Its core strength is also its greatest weakness: it can only describe the world as a straight line.

$$y = \theta_0 + \theta_1 x$$



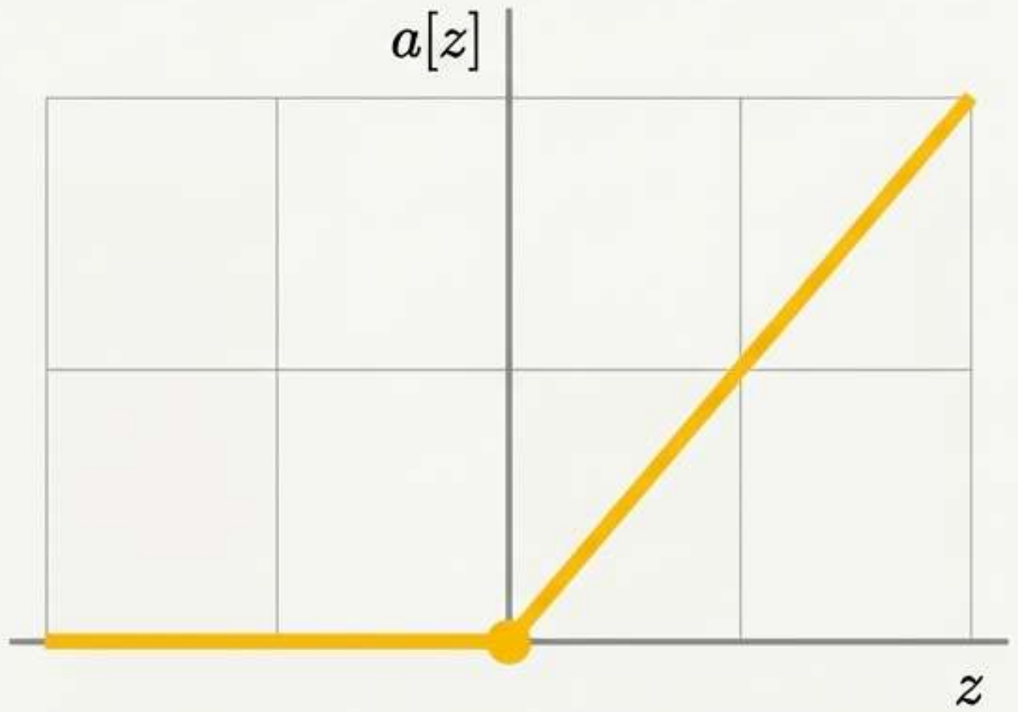
For many real-world problems, **a linear model is too simple.** We need a way to create more flexible, custom-fit functions.

Our First Building Block is a "Programmable Hinge"

The core of our new function is an *activation function*. The most common is the Rectified Linear Unit, or ReLU.

Its rule is simple: if the input is positive, let it pass through. If it's negative, set it to zero.

$$a[z] = \text{ReLU}[z] = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$



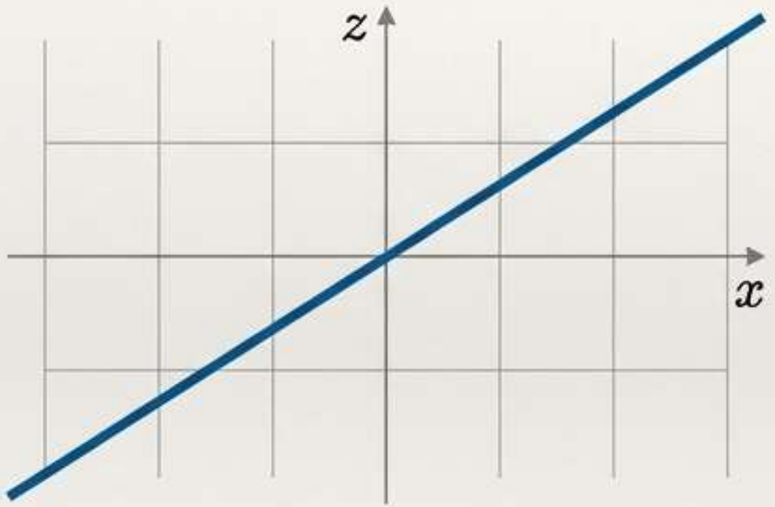
The ReLU acts like a one-way gate, clipping all negative values. This simple, non-linear operation is the key to creating complexity.

Functional Architecture

A Single Hidden Unit Creates One Controlled "Bend"

By passing a simple linear function through a ReLU, we transform a straight line into a “hinge”. This component is called a **hidden unit**.

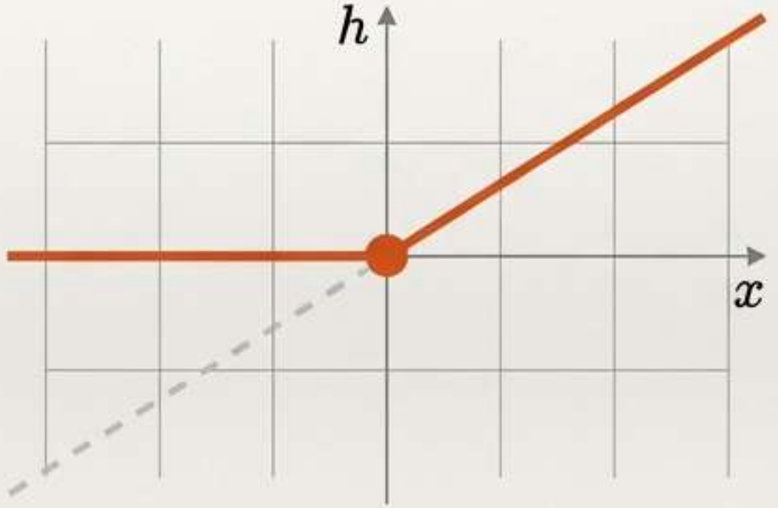
1. Linear Function



$$z = \theta_0 + \theta_1 x$$

ReLU

2. Apply ReLU



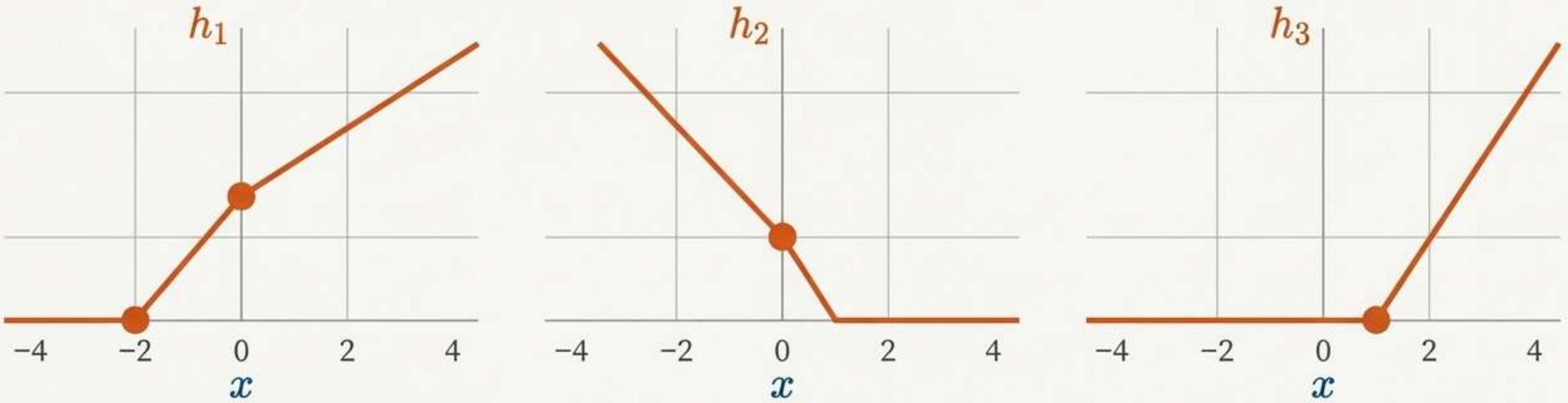
$$h = a[z]$$

The parameters θ_0 (intercept) and θ_1 (slope) now control the *location* and *angle* of this single bend. We can place this hinge wherever we want.

We Can Design and Combine Multiple Hinges to Form a Blueprint

A shallow neural network creates several of these hidden units in parallel, each with its own parameters. Think of them as a set of custom-designed building blocks.

$$h_1 = a[\theta_{10} + \theta_{11}x]$$
$$h_2 = a[\theta_{20} + \theta_{21}x]$$
$$h_3 = a[\theta_{30} + \theta_{31}x]$$



Each hidden unit responds to the input x differently, activating at a specific point and rising with a specific slope.

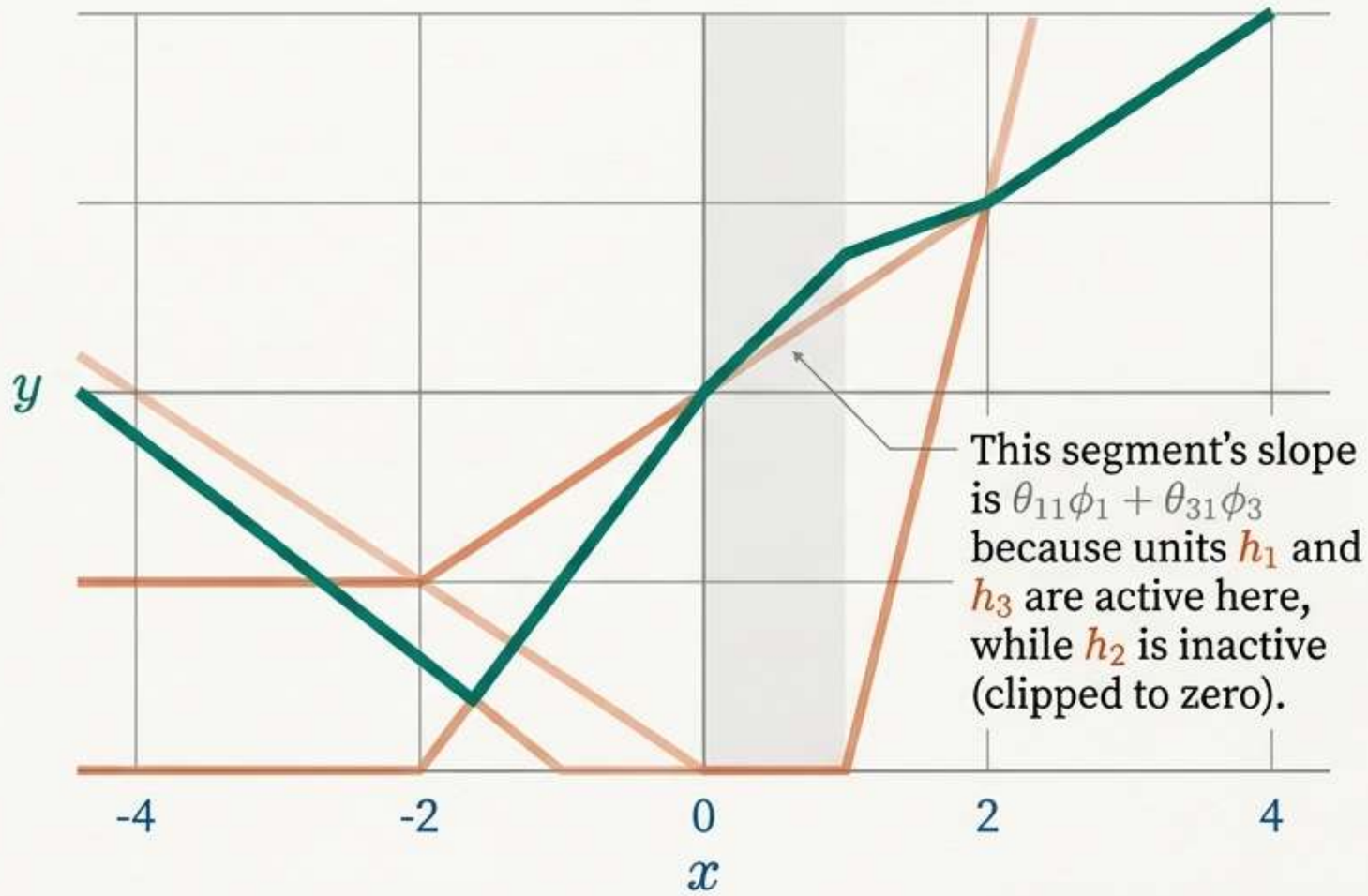
A Final Linear Combination Assembles the Blocks into a Custom Function

The Assembly Process

The final output y is simply a weighted sum of the hidden units, plus a final offset.

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

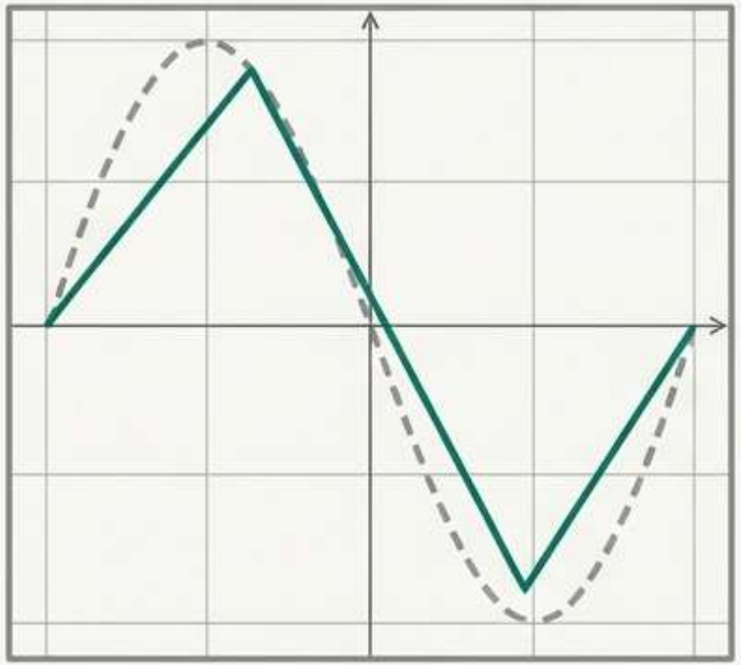
- The weights ϕ_1, ϕ_2, ϕ_3 control the *scale* or *importance* of each hinge.
- The offset ϕ_0 controls the overall *vertical height* of the final function.



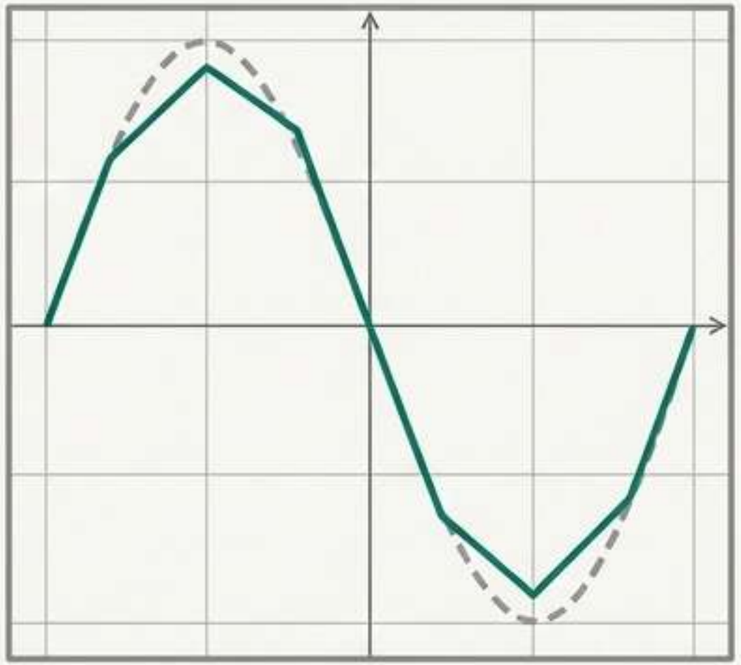
With Enough Hidden Units, We Can Approximate Any Continuous Function

This method is incredibly powerful. The **Universal Approximation Theorem** proves that a shallow network can approximate any continuous function to any desired precision, simply by adding more hidden units.

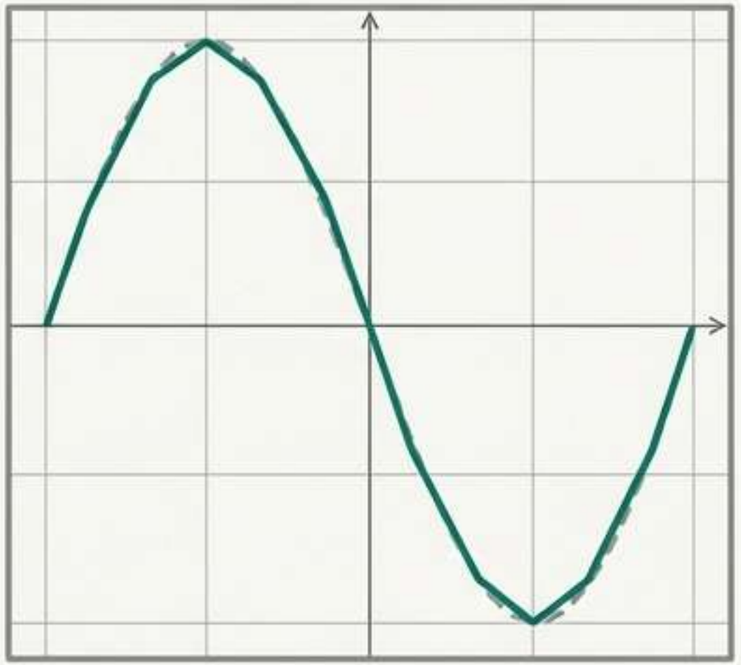
3 Hidden Units



6 Hidden Units

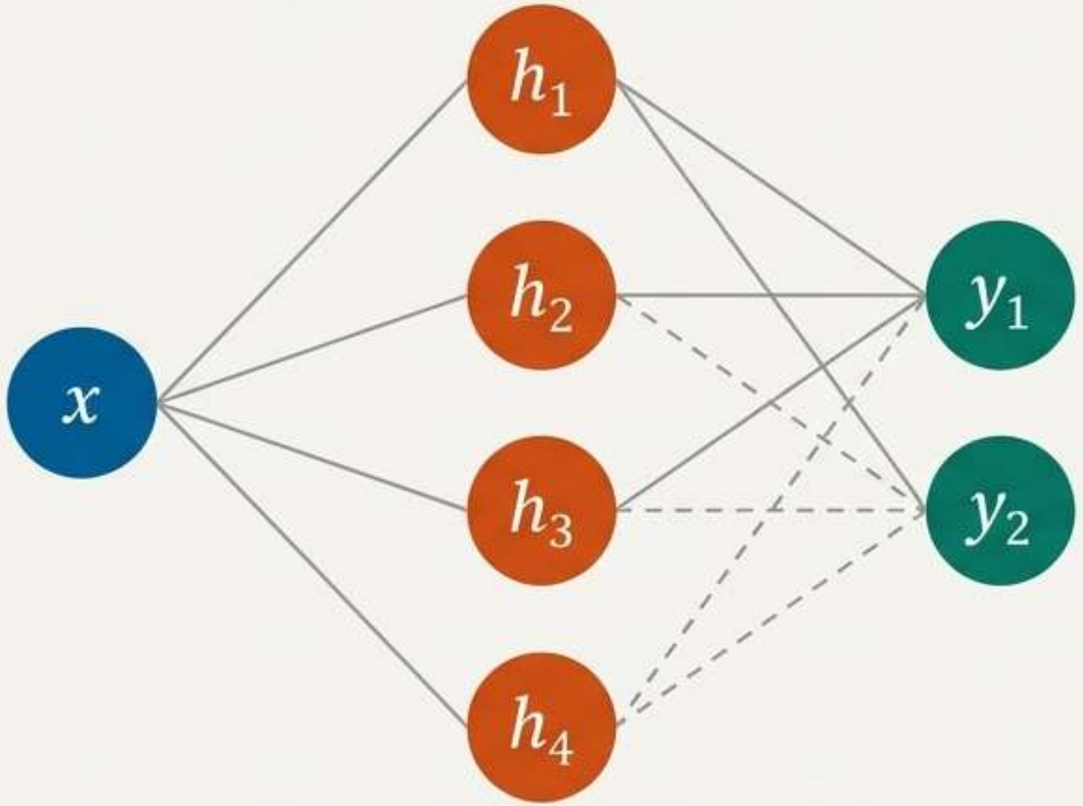


11 Hidden Units



Each hidden unit adds another potential 'joint' and linear region. As the number of units increases, the function becomes more flexible, allowing us to capture increasingly intricate patterns.

The Same Hidden Units Can Build Multiple, Related Outputs

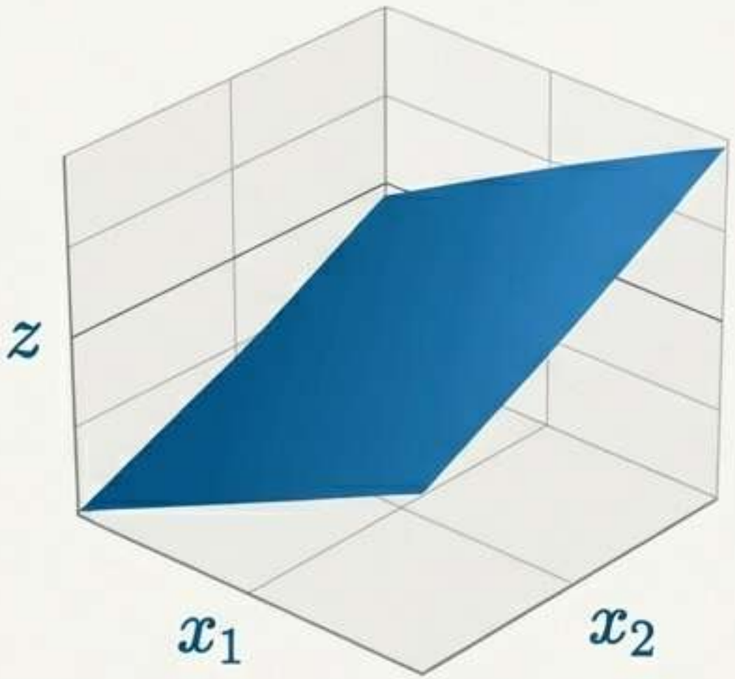


Since both outputs are built from the same hidden units, their 'joints' are synchronized. However, the final shape of each output function is independent, determined by its own set of weights ϕ.

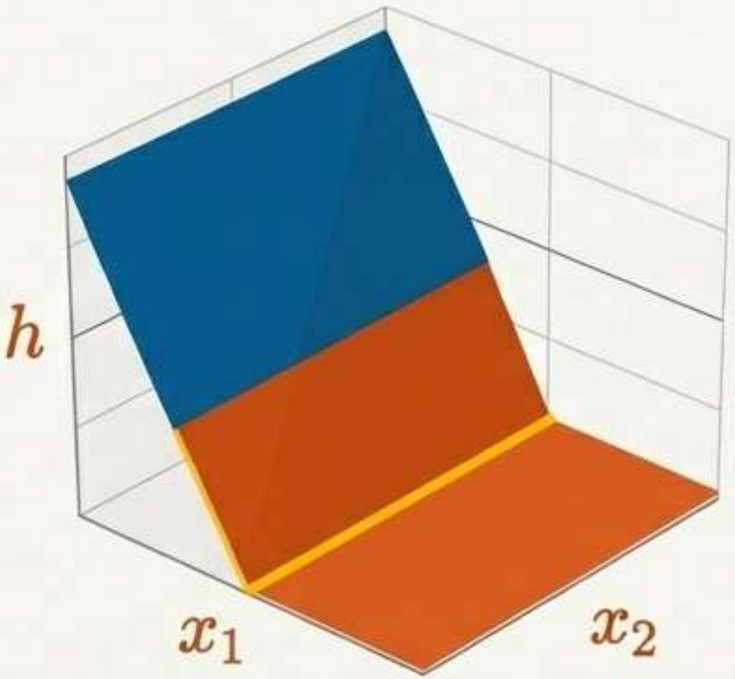
With Multiple Inputs, Our ‘Hinges’ Become Hyperplanes

When we have multiple inputs, like x_1 and x_2, the input to a hidden unit is no longer a line, but a plane: $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2$. The ReLU function then clips this entire plane at zero.

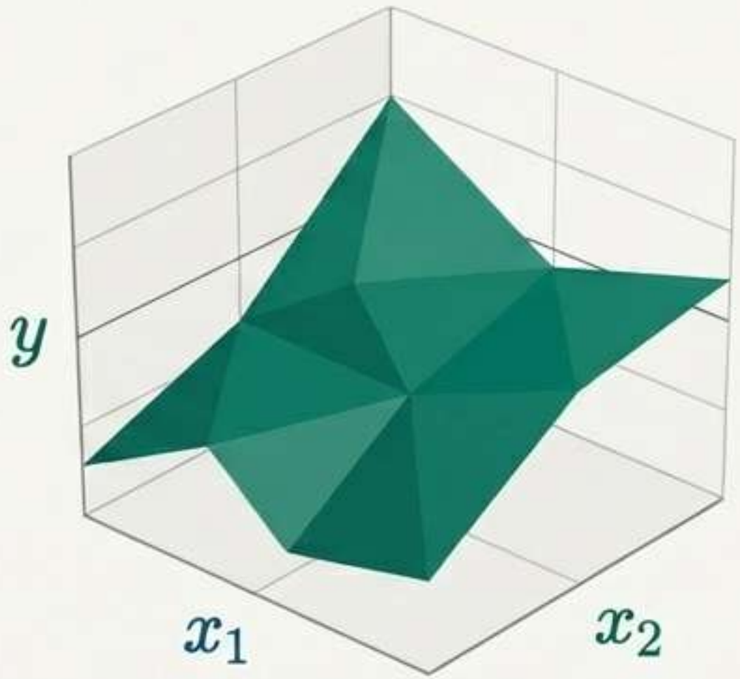
Linear Plane (z)



Clipped Plane (ReLU(z))

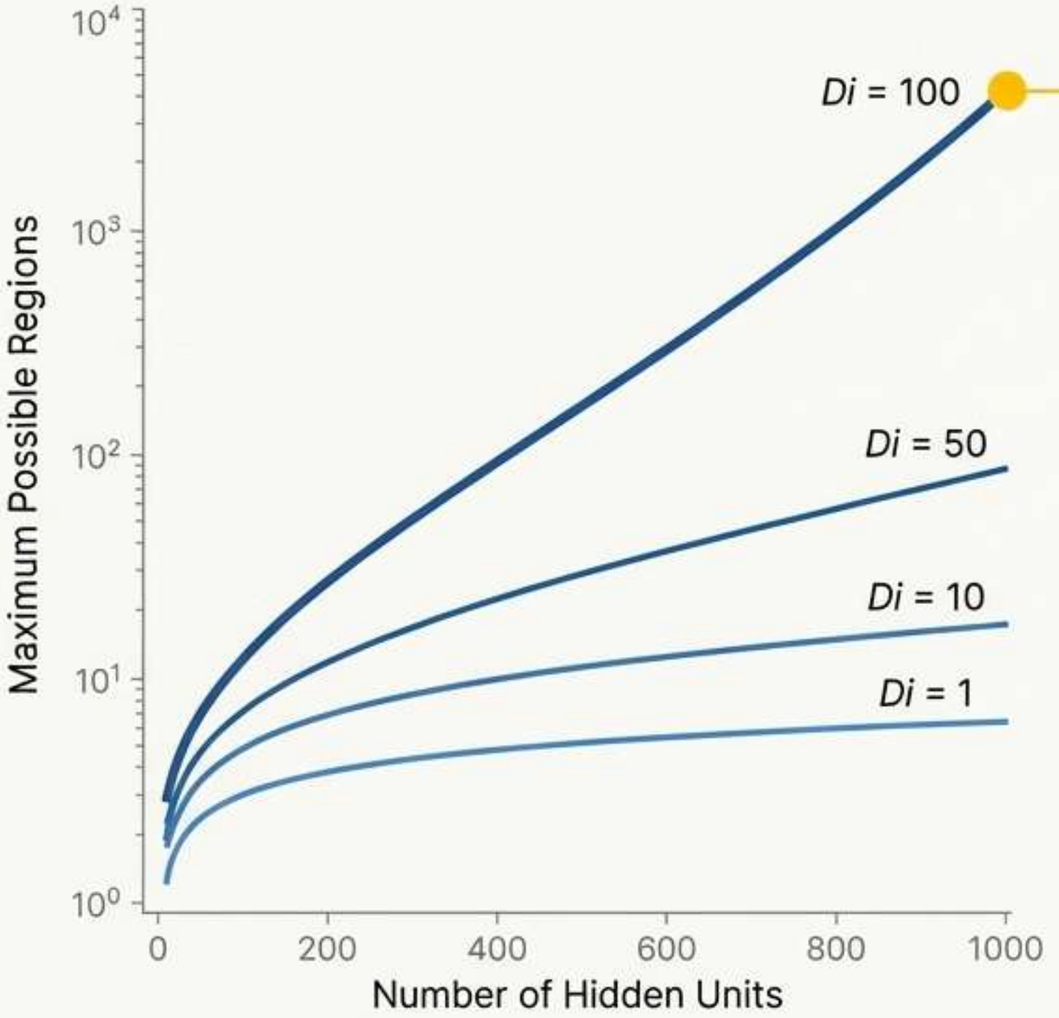


Sum of Clipped Planes (y)

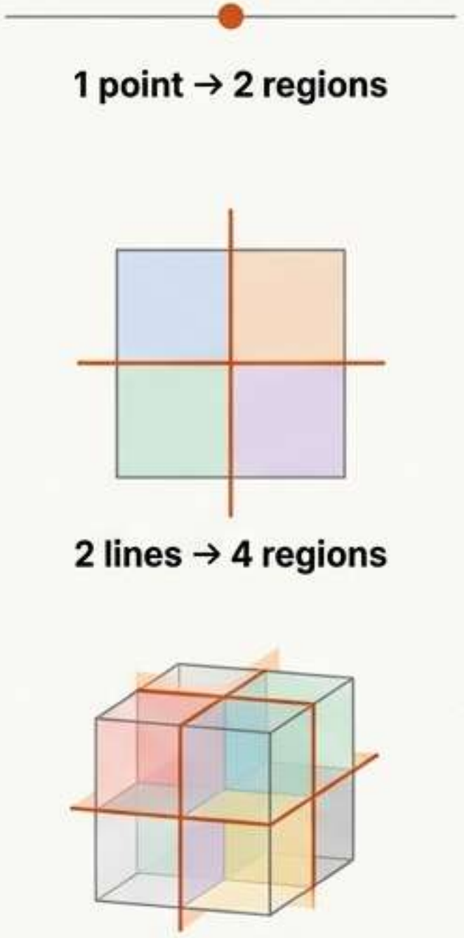


The network now carves the multi-dimensional input space into regions, applying a different simple linear function within each one.

The Number of Regions Grows Explosively with More Dimensions and Units



With 100 inputs and 500 hidden units, a small network by modern standards, we can create over 10^{107} **distinct linear regions.**



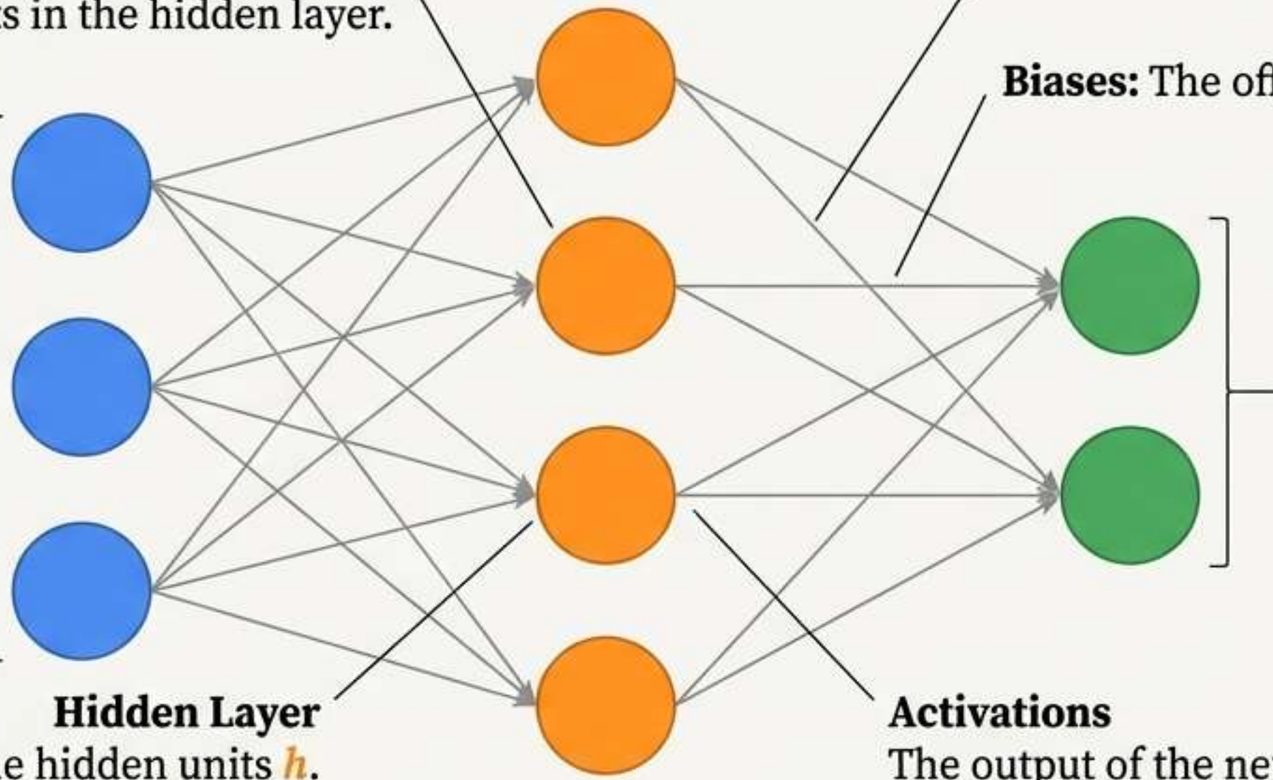
Each hidden unit effectively adds a hyperplane that can slice the input space. The number of resulting regions can grow exponentially with the number of input dimensions.

We Can Now Formally Name the Components of Our Architecture

Neurons/Hidden Units
The individual computational units in the hidden layer.

Weights: The slope parameters (θ, ϕ). Represented by connections/arrows.

Biases: The offset parameters (θ_0, ϕ_0).



Input Layer
Where the data x enters.

Output Layer
Produces the final prediction y.

Hidden Layer
Contains the hidden units h. The core of the computation.

Activations
The output of the neurons *after* the ReLU function is applied.

Feed-Forward & Fully Connected: Describes this architecture: information flows forward, and every unit in a layer connects to every unit in the next.

The Entire Architecture is Captured by Two Compact Equations

We can now write the general definition for a shallow network mapping a multi-dimensional input $\boldsymbol{x}$ to a multi-dimensional output $\boldsymbol{y}$ using D hidden units.

1. The Hidden Layer

$$\boldsymbol{h}_d = \boldsymbol{a}\left[\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di}\boldsymbol{x}_i\right]$$

For each hidden unit $\boldsymbol{d}$, compute a weighted sum of all inputs $\boldsymbol{x_i}$, add a bias θ_{d0}, and apply the ReLU activation function $\boldsymbol{a}[\bullet]$.

2. The Output Layer

$$\boldsymbol{y}_j = \phi_{j0} + \sum_{d=1}^{D} \phi_{jd}\boldsymbol{h}_d$$

For each output unit $\boldsymbol{j}$, compute a weighted sum of all the hidden unit activations $\boldsymbol{h}_d$ and add a final bias ϕ_{j0}.

Complex Functions Are Built from Simple, Piecewise Linear Blocks

1. **Compute Linear Functions**
   Start by computing several simple linear functions of the input(s).

2. **Apply Non-Linearity**
   Pass each result through a ReLU activation function to create 'hinges' or 'clipped planes'.

3. **Combine Linearly**
   Take a final weighted sum of these activated units to assemble the final, complex output.

Shallow neural networks are not a black box. They are an elegant and interpretable system for carving an input space into many simple regions and applying a unique linear model within each. By understanding these fundamental building blocks, we can grasp the power of even the most complex neural architectures.