AI Research Labs

Section 5 of 11 · 3 lessons

The Universal Approximation Theorem

What the theorem says, the intuition behind it, and what it does not promise about training, size or generalisation.

5.1 The Universal Approximation Theorem

5.2 Building Functions from Bumps

5.3 What the Theorem Does Not Promise



Free course with quizzes, progress tracking and Scoopy, an AI tutor:
airesearchlabs.si/learn/dl

Deep Learning Foundations · Edition November 2026 · © 2026 AI Research Labs. Free for personal study.

Where this section fits

Every lesson is a step in building a model. These are the steps this section teaches; the Big picture PDF and the online map show how they connect to the rest.

UNDERNEATH THE MAP · FOUNDATION

Universal approximation

Lessons here: 5.1, 5.2, 5.3 · feeds into Stack layers, Size the network

The Universal Approximation Theorem

One hidden layer with enough neurons can approximate any continuous function on a bounded domain as closely as you like. The precise statement, what each condition means, and the extension that covers ReLU.

The statement

Let σ be a continuous sigmoidal activation, $K \subset \mathbb{R}^d$ a compact (closed and bounded) set, and $h : K \to \mathbb{R}$ any continuous function. Then for every $\varepsilon > 0$ there is a network with **one hidden layer** of finitely many units,

$$\mathrm{NN}(x) = \sum_{j=1}^{n} v_j \, \sigma\big(w_j^\top x + b_j\big) \,,$$

such that

$$\sup_{x \in K} \big|h(x) - \mathrm{NN}(x)\big| < \varepsilon.$$

This is Cybenko's theorem (1989). Leshno and colleagues (1993) extended it to any continuous activation that is **not a polynomial**, which includes ReLU.

Reading each condition

CONDITION	WHAT IT MEANS
Compact K	Inputs come from a closed, bounded region, such as $[0,1]^d$. No promise about behaviour far outside it.
Continuous h	No jumps in the target. (Classification outputs are handled through probabilities; see lesson 5.3.)
Any $\varepsilon > 0$	You can make the error as small as you want, by using enough units.
$\sup$, the worst case	The error bound holds **everywhere** on K at once (uniform approximation), not just on average.
Non-polynomial activation	A polynomial activation only produces polynomials of bounded degree, which can't approximate everything.

PLAIN-LANGUAGE VERSION

"On a closed and bounded input domain, a neural network with a single hidden layer, enough neurons and a suitable non-linear activation can approximate any continuous function to any desired accuracy."

Common doubts

FIRST PASS **If one hidden layer is enough, why do we build deep networks?**

Because "enough neurons" can mean an impractically huge number, and the theorem says nothing about finding the weights or about generalising from limited data. Deep networks can often represent the same function with far fewer units, by composing simple pieces (Section 3). Lesson 5.3 lists what the theorem doesn't promise.

FIRST PASS **What do "continuous function", "approximate" and "compact set" mean in plain words?**

TERM	PLAIN MEANING	EXAMPLE
Continuous function	You can draw it without lifting your pen: no jumps	$\sin x$ is continuous; a step from 0 to 1 is not
Approximate	Get as close as you want to the target function, everywhere on the domain (within ε)	Error below 0.01 at every input in $[0, 1]$
Compact set	A closed, bounded range of inputs	$[0, 1]$ or a box $[0, 1]^d$; not "all real numbers"

"Approximate" here is about **representing** a function that's already known. It isn't about predicting new data: generalisation is a separate question the theorem doesn't answer.

FIRST PASS **Does the theorem say that in practice we need more than one hidden layer?**

No. It says one hidden layer is **enough to exist**. The reason we build deep networks in practice comes from elsewhere: the one-layer network may need an impractically large number of neurons, while a deeper network can represent the same function with far fewer (lesson 11.1).

Cheatsheet

Universal approximation

Continuous h on compact K, any $\varepsilon > 0$: a one-hidden-layer network with enough units has $\sup_{x \in K} |h(x) - \mathrm{NN}(x)| < \varepsilon$. Non-polynomial activations (incl. ReLU).

Online version with the interactive quiz: airesearchlabs.si/learn/dl/universal-approximation/the-universal-approximation-theorem

Building Functions from Bumps

The intuition behind universal approximation in one dimension: two steep sigmoids make a bump, many bumps make a staircase, and narrower steps track any continuous curve.

Two sigmoids make a bump

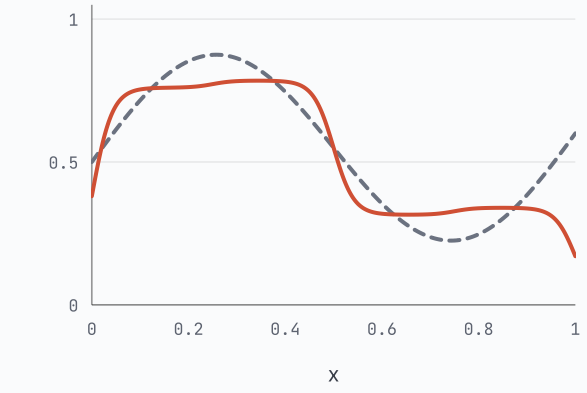
A very steep sigmoid $\sigma(k(x-a))$ with large k is almost a step that switches on at $x = a$. Subtract a second step that switches on at b:

$$\text{bump}_{[a,b]}(x) = \sigma\big(k(x-a)\big) - \sigma\big(k(x-b)\big)$$

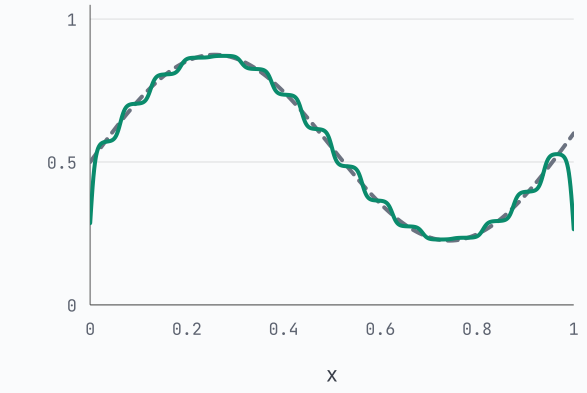
This is about 1 between a and b and about 0 elsewhere. It needs just two hidden units.

Many bumps make any curve

Split the domain into N small intervals. Give each one a bump scaled to the target's height there, and add them all up in the output unit. The result is a staircase that follows the target. Narrower intervals mean smaller steps and less error, at the cost of more units ($2N$ hidden units for N bumps).



Four bumps: a coarse staircase.



Sixteen bumps: already close everywhere.

This is not how a trained network actually works, but it shows why a single hidden layer has, in principle, enough expressive power.

Common doubts

GOING DEEPER **How does this extend to more than one input?**

In two or more dimensions you can build localised "towers" by combining several steep units in different directions, and then sum towers the same way. The number of units needed grows very quickly with the input dimension, which is one reason the theorem's networks can be impractically large.

Cheatsheet

Bump intuition

$\sigma(k(x-a)) - \sigma(k(x-b)) \approx 1$ on $[a, b]$. Sum scaled bumps → staircase approximation; more bumps, smaller error, more units.

Online version with the interactive quiz: airesearchlabs.si/learn/dl/universal-approximation/building-functions-from-bumps

What the Theorem Does Not Promise

The universal approximation theorem is an existence result. It says nothing about how many neurons, how to find the weights, or whether the network will generalise. Plus: how it applies to classifiers with discrete outputs.

Existence is not attainability

The theorem guarantees that a suitable network **exists**. It does not say:

- **How many neurons** you need: the number can be astronomically large.
- **How to find the weights**: it gives no training algorithm.
- **That gradient descent will find them**: training can get stuck even when a solution exists.
- **Anything about generalisation**: fitting a function on the training data is not the same as predicting new data well.
- **How much data** you need.

FROM THE LAB

A 2–2–1 sigmoid network *can* represent XOR, yet in a lab experiment it failed to train in about 20% of random starts: the two hidden units ended up computing the same function, or one saturated. With 3 or more hidden units, every run succeeded. Sufficient capacity is not the same as easy training.

Is one hidden layer enough for every task?

In theory yes, in practice usually not. Deep networks can represent many functions far more efficiently, with fewer units, by building **hierarchical features**, and they often train and generalise better for structured data such as images. Section 11 compares depth and width.

Classifiers have discrete outputs

The theorem is about continuous functions, but a class label is discrete. The way round it: a classifier doesn't output the label directly. It outputs **probabilities** (through a sigmoid or softmax), which are continuous functions of the input, and the network can approximate those. A decision rule, such as picking the most probable class, then turns them into a label.

EXAM-READY WORDING

"The universal approximation theorem is an existence result about approximation. It does not guarantee a practical network size, an efficient training procedure, good generalisation or modest data requirements."

Common doubts

FIRST PASS **Does the theorem mean overfitting is impossible?**

The opposite, if anything. A network powerful enough to approximate any function is also powerful enough to memorise noise in the training data. The theorem is about what can be represented, not about what will be learned from a finite sample.

Cheatsheet

UAT limits

Existence only: no bound on size, no training method, no guarantee GD finds it, nothing about generalisation or data needed.

UAT and classifiers

Approximate the continuous class probabilities (sigmoid/softmax), then apply a decision rule (argmax / threshold).

Online version with the interactive quiz: airesearchlabs.si/learn/dl/universal-approximation/what-the-theorem-does-not-promise