SCALER School of Technology × Priyansh Saxena

Why Deep Learning?

Hand-Crafted Features • Universal Approximation • Representation Learning

Priyansh Saxena

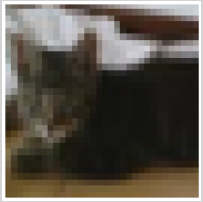
Scaler School of Technology | Deep Learning I: Neural Networks | Week 1 — Session 1/1

29 April 2026

The Challenge

Three images. Three problems.

- A handwritten digit
- A photograph of a cat
- A chest X-ray

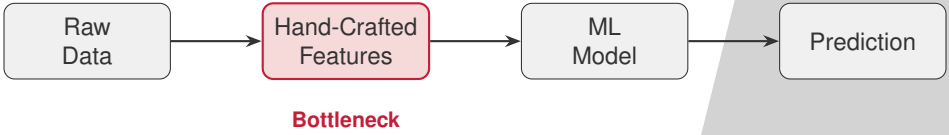


Question

How would you build features for each?

Each domain requires **completely different** expertise.
What if we didn't have to design features at all?

The Classical ML Pipeline



- The quality of the model is **bounded** by the quality of the features.
- Feature design requires deep **domain expertise** and months of iteration.
- Every new task means starting the feature pipeline from scratch.

Feature Engineering Examples

Images

- Count pixels in each region
- Measure color histograms
- Detect edges by hand

Text & Audio

- Count how often each word appears
- Check for specific keywords
- Measure pitch and loudness

Every domain needs **different features** — and a human has to design each one.

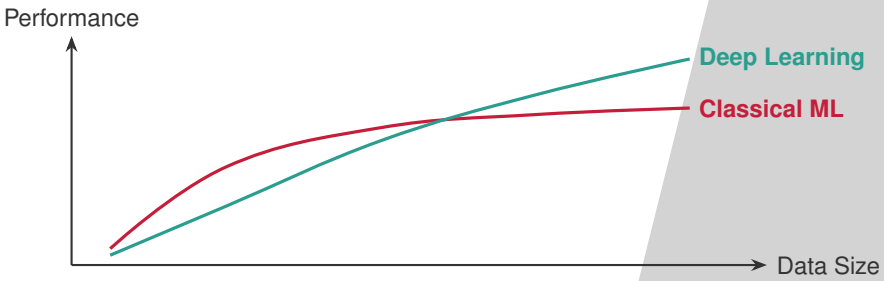
Where Hand-Crafted Features Break

Brittleness of Hand-Crafted Features

Features designed for one setting often fail when conditions change — even slightly.

- **Shift or rotate** an object in an image → many classical features break.
- Each new domain (medical, satellite, audio) needs **entirely new features**.
- Features designed for one task **don't transfer** to another.
- As data complexity grows, the feature engineer becomes the bottleneck, not the algorithm.

The Scaling Problem



Classical models plateau because their fixed features can't absorb more data.
Deep learning **scales** — more data means better learned representations.

Quiz 1

Q1

Why do raw pixel features fail for image classification?

Q2

True or False: “Adding more features always improves model performance.”

Quiz 1 — Answers

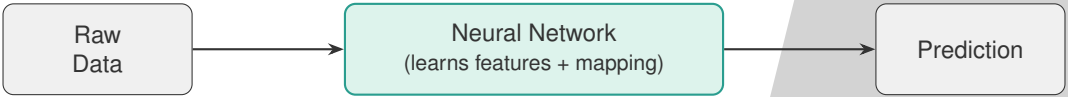
A1

Raw pixels have **no spatial structure**: a one-pixel shift creates a completely different feature vector. They also lack invariance to rotation, scale, and lighting.

A2 — False

More features can introduce **noise, redundancy, and the curse of dimensionality**. Without careful selection, extra features often *hurt* performance.

What If the Model Learned Its Own Features?



- No hand-crafted features — the network discovers what matters.
- The same architecture can work on images, text, and audio.
- Features **improve automatically** as you add more data.

But can a neural network really learn *any* function we need?

The Universal Approximation Theorem

Formal Statement (Cybenko, 1989)

A feedforward network with a single hidden layer containing a finite number of neurons can approximate any continuous function on a compact subset of $\mathbb{R}^n$ to arbitrary accuracy.

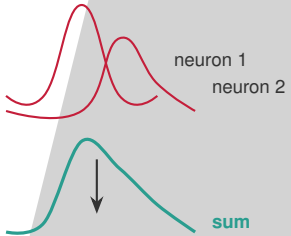
In plain English:

- Give me *any* smooth input-output relationship.
- I can build a one-hidden-layer network that gets as close as you want.
- The catch: I may need a *lot* of neurons.

Neural networks are **universal function approximators** — they can, in principle, represent any mapping you need.

The LEGO Analogy

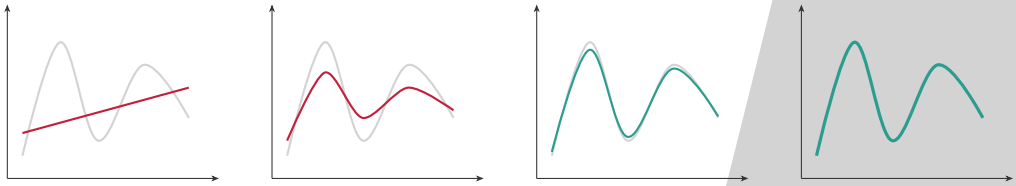
- Each **neuron** is a single LEGO brick — it can produce one simple shape (a bump or a step).
- Stack enough bricks → you can build **any shape**.
- More neurons = more bricks = finer detail.
- The network **learns** which bricks to place and where.



Caveat

The theorem guarantees *existence* — it says nothing about how to **find** the right configuration.

UAT in Action



Target function Approximation (few) Approximation (many)

More neurons = better fit.

The Catch

What the UAT does NOT tell you:

1. **How many neurons** you need (could be astronomically large).
2. **How to train** the network (no learning algorithm is guaranteed).
3. **How to generalize** — fitting training data $\neq$ working on new data.
4. **How to choose** the architecture (depth, width, activation).

The 23-Year Gap

The UAT was proved in **1989**. Deep learning only took off in **2012**.

What was missing? **Data** (ImageNet), **Compute** (GPUs), and **Training tricks** (ReLU, Dropout, BatchNorm).

Discussion

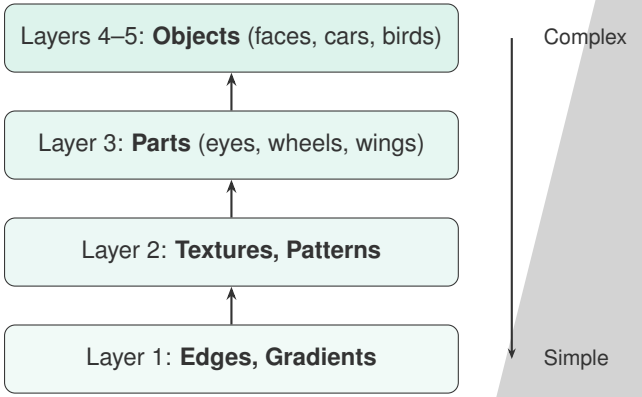
Discussion Prompt

Why did it take **23 years** from the Universal Approximation Theorem (1989) to AlexNet (2012)?

Coming Up Next

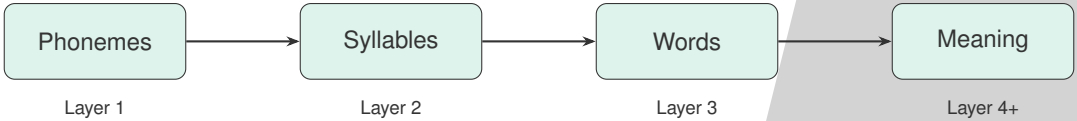
Part 2: Representation Learning — What Neural Networks Learn Inside

The Big Idea — Representation Learning



The network builds its own **feature pipeline** — from raw pixels to high-level concepts — automatically.

The Audio Analogy



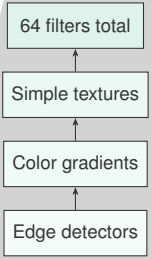
Just like vision networks learn edges $\rightarrow$ textures $\rightarrow$ objects,
speech networks learn phonemes $\rightarrow$ syllables $\rightarrow$ words $\rightarrow$ meaning.

Deep networks discover these hierarchical levels automatically.

What a CNN Learns — Early Layers

VGG16, First Convolutional Layer:

- 64 filters, each 3×3 pixels.
- They detect **edges**, **color gradients**, and **simple textures**.
- These filters were **discovered**, not programmed.
- They look remarkably similar to hand-crafted Gabor filters — but the network found them on its own.



What a CNN Learns — Deep Layers

Feature maps at increasing depth (VGG16):

- **Layer 0** (input): raw pixels — no abstraction.
- **Layer 7**: edges combine into corners and contours.
- **Layer 14**: textures and repeating patterns emerge.
- **Layer 28**: class-specific parts (eyes, wheels, feathers).

Key Insight

Early layers are **generic** — an airplane and a frog share the same edge detectors.
Deep layers are **class-specific** — they represent features unique to each category.

The Power of Learned Features

Features	Model	Accuracy
Raw pixels	Logistic Regression	31%
Raw pixels	Random Forest	42%
VGG features (pretrained)	Logistic Regression	**84%**

Same model. Same data. Only difference: better features.

This is the power of representation learning — a simple classifier on top of learned features outperforms complex classifiers on raw data.

Quiz 2

Q1

Which layers of a CNN detect edges — the **early** layers or the **deep** layers?

Q2

Can neural network features help a simple logistic regression beat a random forest on raw pixels?

Quiz 2 — Answers

A1

Early layers. The first convolutional layers learn edge and gradient detectors. Deeper layers combine these into increasingly complex patterns.

A2

Yes. As we saw, logistic regression on VGG features (84%) easily beats a random forest on raw pixels (42%). Good features matter more than model complexity.

The Simple Rule of Thumb

	Structured Data	Unstructured Data
Small Data	Classical ML wins	Either works
Large Data	Classical or DL	DL wins ★

Deep learning shines when you have **lots of unstructured data** (images, text, audio).

When Classical ML Still Wins

- **Small tabular data** — XGBoost and gradient-boosted trees are hard to beat on structured datasets with few thousand rows.
- **Interpretability is required** — regulations in finance and healthcare may demand explainable models.
- **Limited compute** — deep learning needs GPUs; classical ML runs on a laptop CPU in seconds.

Bottom Line

Deep learning is not always the answer. Choose the right tool for the problem.

The AlexNet Moment (2012)

ImageNet Large Scale Visual Recognition Challenge

Top-5 error dropped from **26.2%** to **16.4%** — a 10-point leap in a single year.

- Previous winners used hand-crafted features (SIFT + Fisher Vectors).
- AlexNet: an 8-layer CNN trained end-to-end on raw pixels.
- Key ingredients: ReLU activations, Dropout regularization, GPU training, data augmentation.
- The gap was so large that **every subsequent winner** used deep learning.

The result that changed everything.

What Made It Possible

Data

ImageNet: 1.2 million labeled images across 1,000 classes. The first dataset large enough to train deep networks without catastrophic overfitting.

Compute

GPUs: NVIDIA GTX 580 with 3 GB VRAM. Parallel matrix operations made training 10–50$\times$ faster than CPUs.

Ideas

ReLU: faster training.
Dropout: regularization.
Augmentation: more data from existing images.
All simple but critical.

We'll learn all of these in this course.

Your Course Map

Week	Topic
1	Why Deep Learning + Perceptrons & MLPs
2	Training Neural Networks (Backprop, Optimizers)
3	Regularization & Practical Training
4	Convolutional Neural Networks (CNNs)
5	CNN Architectures (AlexNet $\rightarrow$ ResNet)
6	Recurrent Neural Networks (RNNs, LSTMs)
7	Sequence-to-Sequence & Attention
8	Transformers & Self-Attention
9	Generative Models (VAEs, GANs, Diff

Architecture Quick Preview

- **CNNs** (Convolutional Neural Networks) — the go-to architecture for images. Learns spatial hierarchies through local filters.
- **RNNs** (Recurrent Neural Networks) — designed for sequences. Processes one time step at a time with memory of the past.
- **Transformers** — the architecture behind modern language models. Uses self-attention to process all tokens in parallel.
- **Generative Models** — networks that create new data: realistic images, music, text, and more.

Recap — 5 Key Takeaways

1. **Hand-crafted features are the bottleneck** in classical ML — they don't scale, don't transfer, and require deep domain expertise.
2. **The UAT guarantees** that neural networks can approximate any continuous function — but says nothing about how to train them.
3. **Representation learning** means the network builds its own feature hierarchy, from edges to objects, automatically.
4. **Deep learning wins** when you have large amounts of unstructured data; classical ML still wins on small tabular problems.
5. **AlexNet (2012)** was the tipping point — made possible by data (ImageNet), compute (GPUs), and key ideas (ReLU, Dropout).

Final Quiz

Q1

What is the main advantage of deep learning over classical ML?

Q2

What does the Universal Approximation Theorem **not** guarantee?

Q3

Give one scenario where you would pick classical ML over deep learning.

Final Quiz — Answers

A1

Deep learning **learns its own features** from raw data, eliminating the need for hand-crafted feature engineering and scaling better with more data.

A2

The UAT does not guarantee **how many neurons** are needed, **how to train** the network, or **how well it generalizes** to unseen data.

A3

Small tabular dataset (e.g., a few thousand rows of structured features) — XGBoost or gradient-boosted trees will likely outperform a neural network and train in seconds.

Thank You!

Questions?

Priyansh Saxena

All lecture materials available at:

www.priyanshsaxena.com/sst-deep-learning-nn