Deep Learning in Finance

Fall 2025

Oualid Missaoui

What this course is ?

- What this course is?
  - In-depth exploration of advanced deep learning techniques for financial time-series & tabular data
  - Seminar format:
    - Theory -> demos -> discussion
  - Pre-requisite:
    - Familiarity with deep neural nets & backpropagation derivations
    - Python/Pytorch
- Logistics & Expectations
  - Seven sessions, each 2.5 hours
  - Come prepared: skim primary text readings; try the starter code each week

Topics at glance

- Theoretical Justifications (approximation, optimization, generalization; double descent; spectral theory)
- Transformers for Financial Time Series Forecasting
- Neural ODEs for Time-Series
- GANs for financial synthetic data
- Diffusion models for financial synthetic data
- Graph Neural Networks for financial data
- Virtue of complexity in finance

Evaluation & Materials

- Evaluation:
  - HomeWorks
  - Final Project
- Materials:
  - Lecture slides
  - Companion GitHub repository (Python/Pytorch)

Theoretical Justification of Deep Learning

Approximation & Scaling Laws

Oualid Missaoui

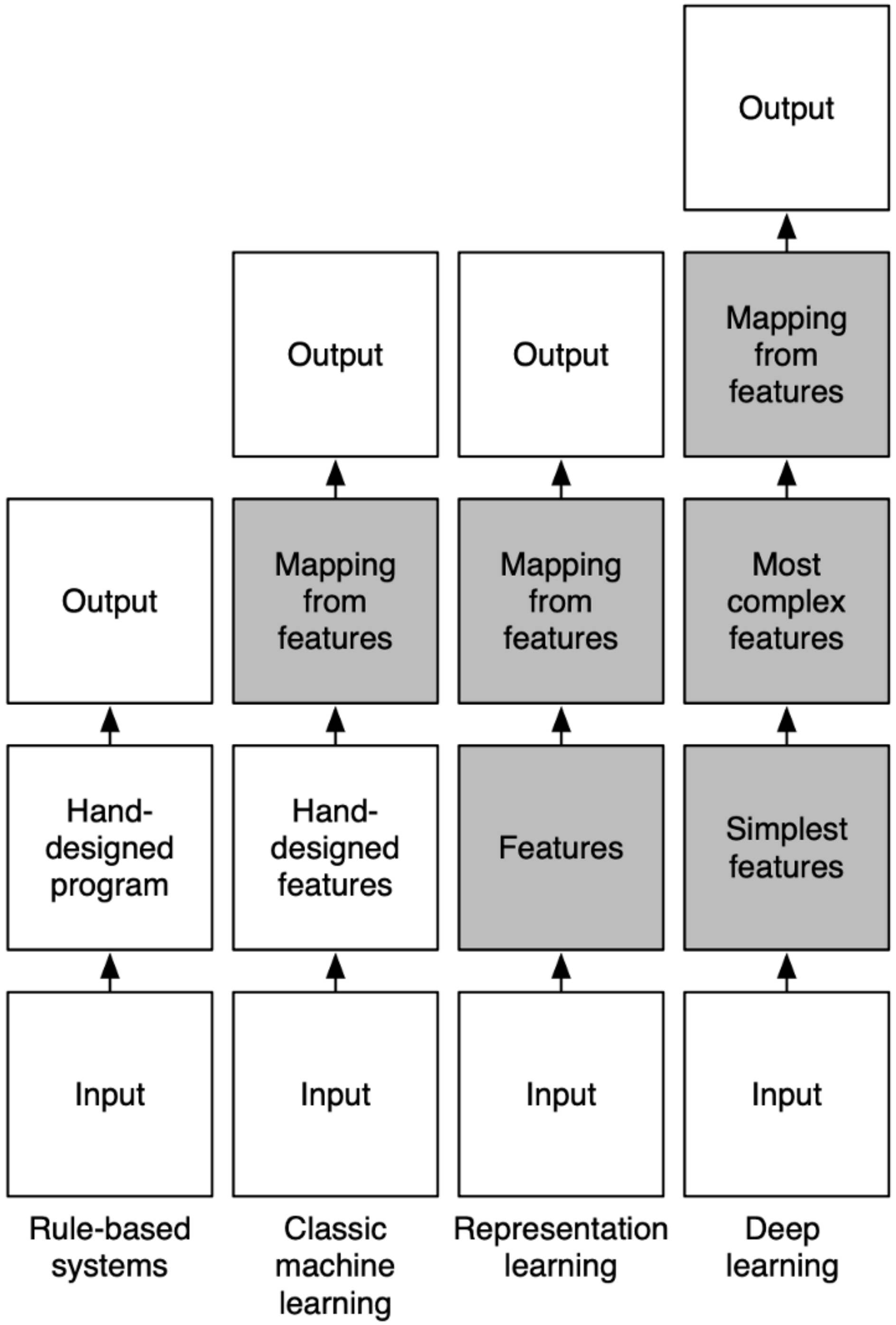
Agenda

- Success of Deep Learning
- Classic statistical learning theory recap
- Approximation
    - Universal Approximation Theorems
    - Beyond Universality: Expressivity and depth
- Neural Scaling Laws
- References

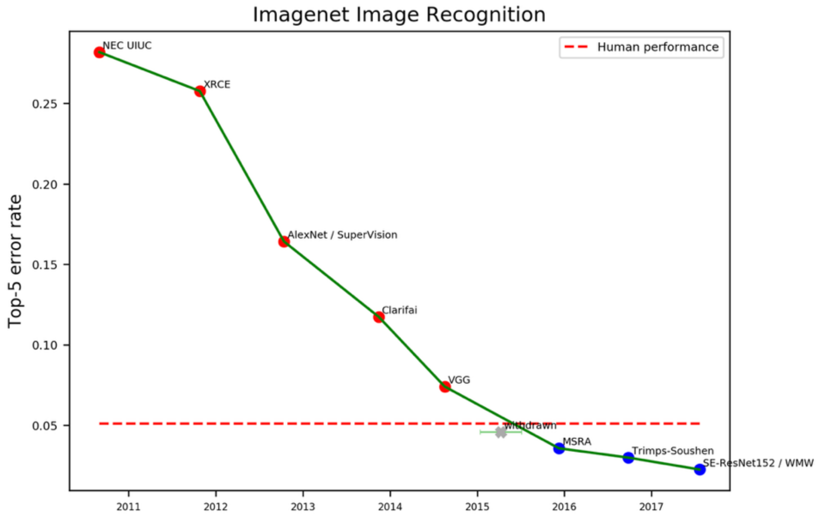
What s deep learning ?

- Big idea:
  Deep learning learns a *hierarchy of representations* directly from raw data and optimizes the whole pipeline end-to-end.
- Why it matters:
  - Replaces manual feature engineering with learned features at multiple levels of abstraction.
  - Scales with data + compute; performance keeps improving as both grow.
  - Encodes powerful inductive biases (e.g., parameter sharing in convs, attention in Transformers, compositionality).
  - Enables transfer: pretrain on broad data → fine-tune for a task.

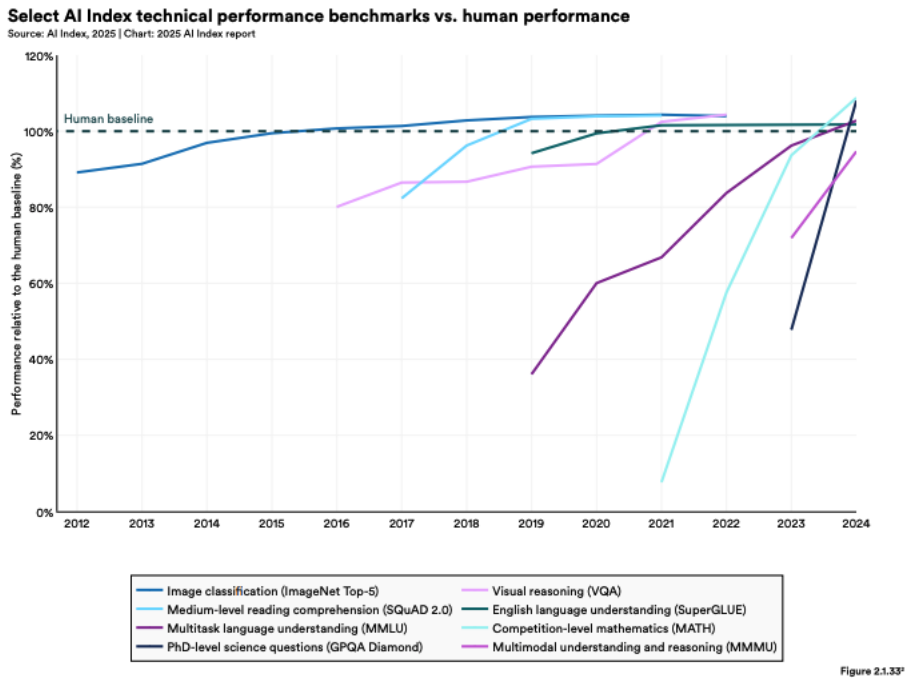
https://www.iro.umontreal.ca/~bengioy/talks/mlss-austin.pdf

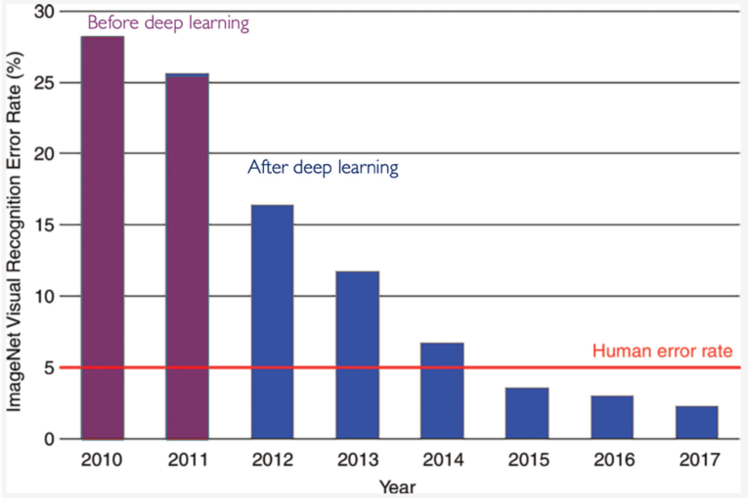


Success of deep learning

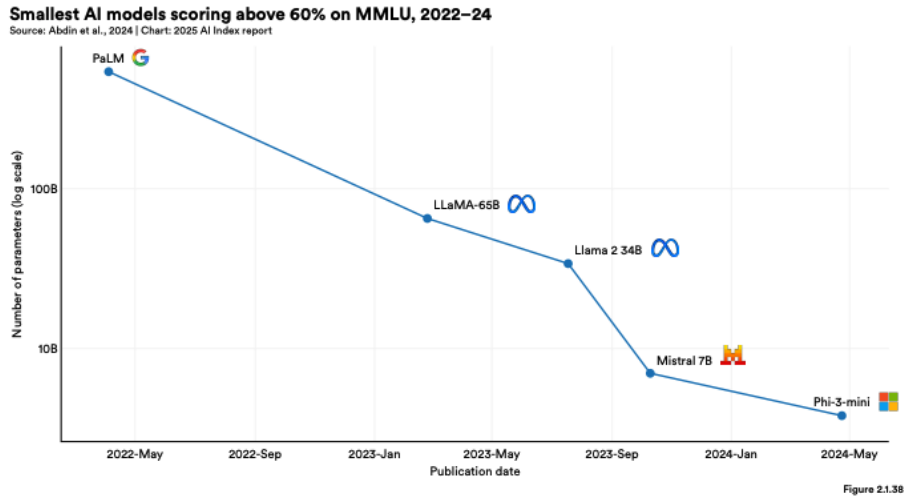


source





source

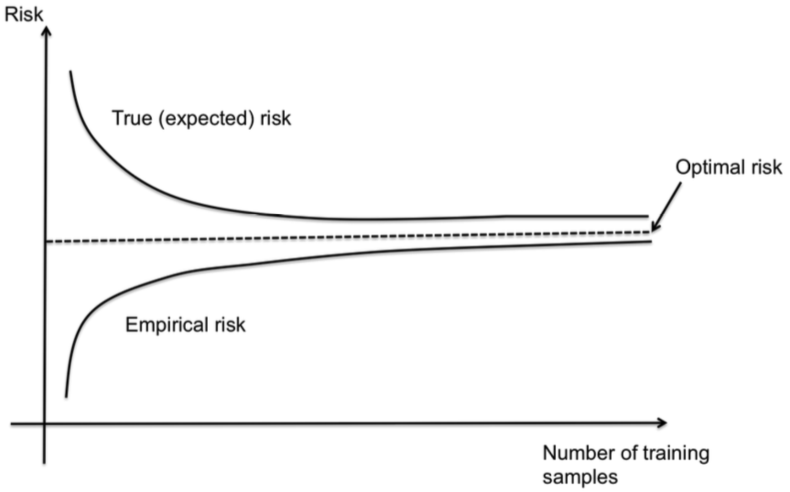


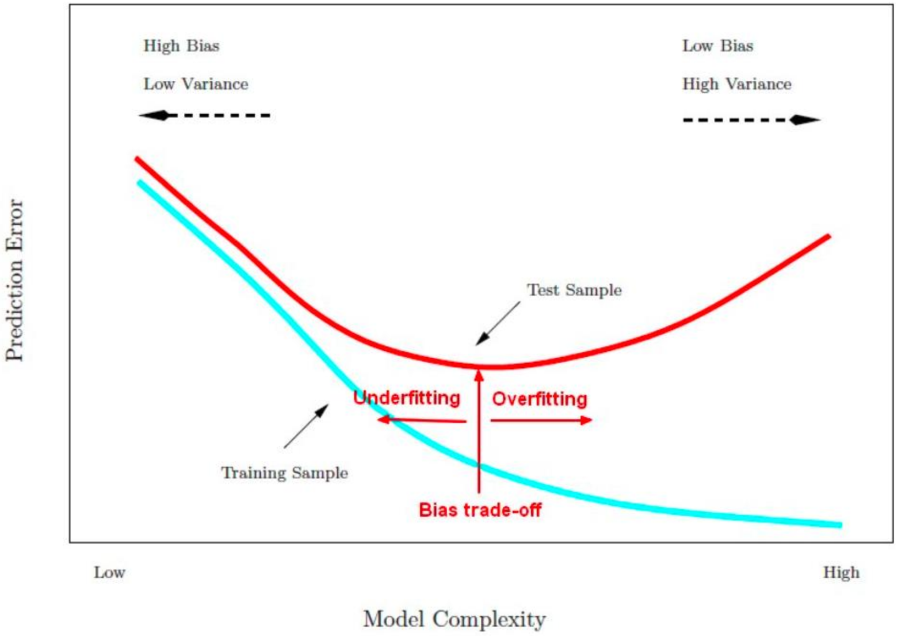
source

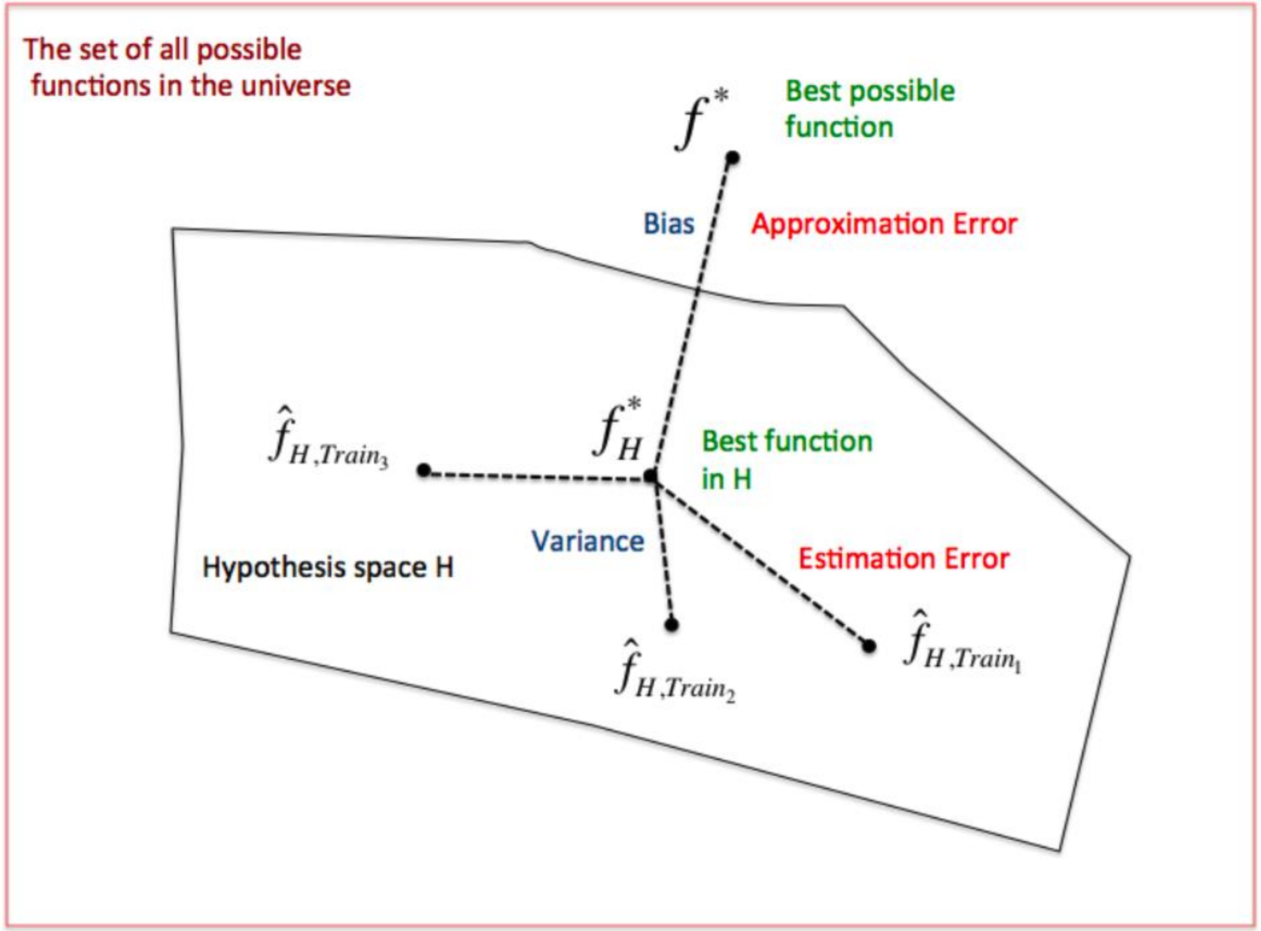
From Breakthrough to Deployment (2024–25)

- **Conclusions**
  - Error collapsed: perception tasks surpassed human level; gains have plateaued on classic benchmarks.
  - Capability broadened: strong progress across vision, language, multimodal—uneven on math/logic.
  - Efficiency jumped: “good-enough” performance now with small models (<10B), slashing cost/latency.
- **Implications**
  - New bottleneck = reasoning, reliability, and transfer—less about raw recognition.
  - Best ROI: domain-tuned small models + tool use + retrieval/knowledge graphs.
  - Eval must shift to cost-quality-safety: robustness, data provenance, and verifiability.
- **So what for us?**
  - Prioritize data/decision memory and task-specific fine-tuning over brute scale.
  - Measure dollars per correct action, not leaderboard points.
  - Build for constrained deployment (edge/on-prem) by default.

Classic statistical learning theory recap







Bias–Variance: what to remember

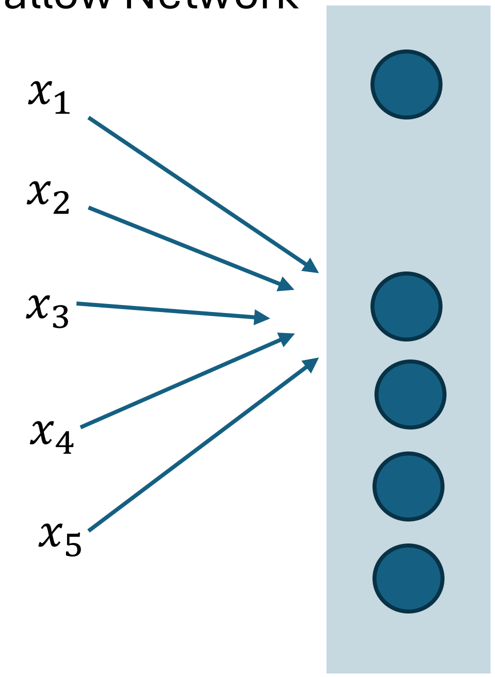
- Test risk combines irreducible noise, approximation error, and estimation error.
- Increasing model capacity usually lowers approximation error but raises estimation error unless you add data or regularization.
- Underfitting appears when both training and validation errors remain high.
- Overfitting appears when training error is low but validation error is high.
- When both errors are high because of noisy labels or poor targets, focus on improving the data.
- You control three dials: model capacity, regularization, and the quality and quantity of data.
- Practical regularization includes weight decay, dropout, data augmentation, and early stopping.
- In deep learning, optimization provides implicit regularization and can produce double descent, yet the bias–variance trade-off still governs generalization.
- Plot learning and capacity curves first so you know whether to buy data, add regularization, or scale the model.
- Aim for the best outcome per dollar by favoring compact, well-regularized models supported by high-quality data.

The Universal Approximation Theorem

Theorem. *Let $g(x)$ be a continuous function defined in a compact subset $S \subset \mathbb{R}^l$ and any $\epsilon > 0$. Then there is a two layer neural network $\hat{g}(x)$ with $K(\epsilon)$ hidden nodes so that*

$$|g(x) - \hat{g}(x)| < \epsilon \quad \forall x \in S$$

Shallow Network



- A single hidden-layer network with a non-linear activation can approximate any continuous function on a bounded input domain.
- *Compact subset* just means we restrict inputs to a normalized, bounded region; exactly what we do in practice.
- The result is about representational capacity only; it does not guarantee that training will find the best weights or need only a little data.

George Cybenko. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4):303-314, 1989.

Proof of the Universal Approximation Theorem (1)

Definition. *A function σ is discriminatory if given a measure $\mu \in M(I_n)$ such that*

$$\int_{I_n} \sigma(w^T x + b) d\mu(x) = 0, \forall w \in \mathbb{R}^n, b \in \mathbb{R}$$

implies that $\mu = 0$

Definition. *A function $\sigma : \mathbb{R} \to \mathbb{R}$ is called sigmoidal if it is non-decreasing and satisfies the asymptotic conditions*

$$\lim_{z \to -\infty} \sigma(z) = 0 \quad \textit{and} \quad \lim_{z \to \infty} \sigma(z) = 1.$$

An example of such a function is the logistic function $\sigma(z) = \frac{1}{1+e^{-z}}$.

Lemma. *Any bounded, measurable, sigmoidal function is discriminatory. In particular, any continuous sigmoidal function is discriminatory.*

Proof of the Universal Approximation Theorem (2)

Proof. Let $C(S)$ denote the space of continuous functions on the compact set $S \subset \mathbb{R}^n$, equipped with the uniform norm. Define the set of two-layer neural networks:

$$\mathcal{N} = \left\{ \hat{g}(x) = \sum_{k=1}^{K} \alpha_k \sigma(\beta_k^T x + \gamma_k) \,\middle|\, \alpha_k, \gamma_k \in \mathbb{R},\ \beta_k \in \mathbb{R}^n,\ K \in \mathbb{N} \right\},$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a continuous, sigmoidal activation function. We will show that $\mathcal{N}$ is dense in $C(S)$.

Suppose, for contradiction, that $\mathcal{N}$ is **not dense** in $C(S)$. By the Hahn-Banach theorem, there exists a non-zero bounded linear functional $L : C(S) \to \mathbb{R}$ that vanishes on $\mathcal{N}$, i.e., $L(f) = 0$ for all $f \in \mathcal{N}$.

By the Riesz representation theorem, any bounded linear functional on $C(S)$ can be represented by a finite signed Borel measure μ on S:

$$L(f) = \int_S f(x)\, d\mu(x) \quad \forall f \in C(S).$$

Since L vanishes on $\mathcal{N}$, we have:

$$\int_S \sigma(\beta^T x + \gamma)\, d\mu(x) = 0 \quad \forall \beta \in \mathbb{R}^n, \gamma \in \mathbb{R}.$$

By the lemma established above, any continuous sigmoidal function σ is discriminatory. Therefore, this integral condition implies $\mu = 0$, which contradicts the assumption that L is a non-zero functional.

Hence, $\mathcal{N}$ must be dense in $C(S)$. Density implies that for any $g \in C(S)$ and $\epsilon > 0$, there exists $\hat{g} \in \mathcal{N}$ such that:

$$\sup_{x \in S} |g(x) - \hat{g}(x)| < \epsilon.$$

In particular, $\hat{g}$ can be written as $\hat{g}(x) = \sum_{k=1}^{K(\epsilon)} \alpha_k \sigma(\beta_k^T x + \gamma_k)$, where $K(\epsilon)$ is finite. This completes the proof.

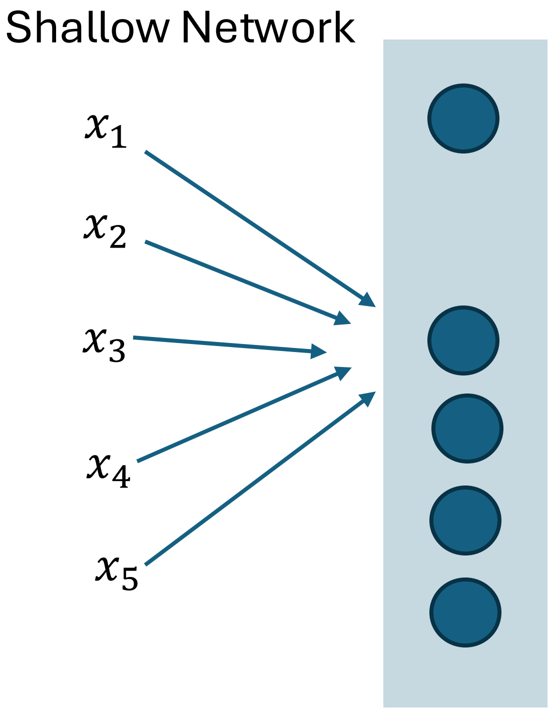
$\square$

Complexity of approximation

Theorem. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable, and not a polynomial. For $f \in W_m^n$ the complexity of shallow networks that provide accuracy at least ϵ is*

$$N = \mathcal{O}(\epsilon^{-\frac{n}{m}})$$

and is the best possible



Key implications:

- Curse of dimensionality: Shallow networks struggle significantly in high-dimensional spaces, as complexity scales exponentially with dimension.
- Trade-off: There's a clear trade-off between dimensionality n accuracy ϵ and smoothness q. Improving approximation accuracy quickly becomes computationally prohibitive as dimensions grow unless smoothness is very high.
- **Optimality**: This theorem also provides a lower bound—no shallow neural network architecture (with these assumptions) can surpass this complexity rate. Thus, the provided scaling is the best you can hope for in shallow approximations.

Clearly, shallow networks, though universal approximators, quickly become inefficient as dimensionality increases.

Compositional functions

Many of the computations performed on images should reflect the symmetries in the physical world that manifest themselves through the image statistics. Assume for instance that a computational hierarchy such as

$$h_l(\cdots h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8))\cdots))) \qquad (2.1)$$

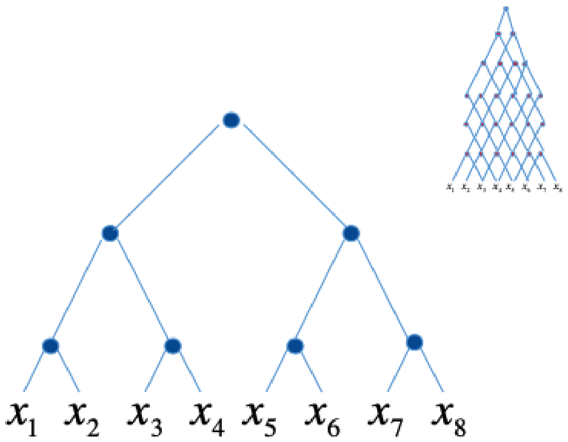


Figure 3: *A binary tree hierarchical network in 8 variables, which approximates well functions of the form (2.2). Each of the nodes consists of n units and computes the ridge function $\sum_{i=1}^{n} a_i \sigma(\langle \mathbf{v}_i, \mathbf{x}\rangle + t_i)$, with $\mathbf{v}_i, \mathbf{x} \in \mathbb{R}^2$, $a_i, t_i \in \mathbb{R}$. Similar to the shallow network such a hierarchical network can approximate any continuous function; the text proves how it approximates compositional functions better than a shallow network. Shift invariance may additionally hold implying that the weights in each layer are the same. The inset at the top right shows a network similar to ResNets: our results on binary trees apply to this case as well with obvious changes in the constants*

For explaining our ideas for the deep network, we consider compositional functions conforming to a binary tree. For example, we consider functions of the form (cf. Figure 3)

$$f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8))). \qquad (2.2)$$

For the hierarchical binary tree network, the spaces analogous to $W^{\mathrm{NN}}_{r,q}$ are $W^{\mathrm{NN}}_{H,r,2}$, defined to be the class of all functions f which have the same structure (e.g., (2.2)), where each of the constituent functions h is in $W^{\mathrm{NN}}_{r,2}$ (applied with only 2 variables). We define the corresponding class of deep networks $\mathcal{D}_n$ to be set of all functions with the same structure, where each of the constituent functions is in $\mathcal{S}_n$. We note that in the case when q is an

integer power of 2, the number of parameters involved in an element of $\mathcal{D}_n$ – that is, weights and biases, in a node of the binary tree is $(q-1)(q+2)n$.

The following theorem (cf. [13]) estimates the degree of approximation for shallow and deep networks. We remark that the assumptions on σ in the theorem below are not satisfied by the ReLU function $x \mapsto |x|$, but they are satisfied by smoothing the function in an arbitrarily small interval around the origin.

Hrushikesh N. Mhaskar and Tomaso Poggio (2016) , Deep vs. Shallow Networks: an Approximation Theory Perspective

Depth Efficiency

Theorem. *For $f \in W_m^{n,2}$ consider a deep network with the same compositional architecture and with an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ which is infinitely differentiable, and not a polynomial. The complexity of the network to provide approximation with accuracy at least ϵ is:*

$$N = \mathcal{O}((n-1)\epsilon^{-\frac{2}{m}})$$

Clearly, deep networks, when aligned with the compositional structure of the target function, can approximate much more efficiently, with complexity scaling only linearly in dimension and polynomially in accuracy.

Feature	**Shallow Networks**	**Deep Networks**
Function class	$f \in W_m^n$ (general Sobolev)	$f \in W_m^{n,2}$ (compositional structure)
Approximation complexity	$\mathcal{O}(\epsilon^{-n/m})$	$\mathcal{O}((n-1)\epsilon^{-2/m})$
Curse of dimensionality	Yes (exponential in n)	Avoided (linear in n)
Optimality	Yes (best possible)	Not necessarily, but much better
Assumes structure in f	No	Yes (hierarchical/compositional)

Telgarsky, M. (2016). *Benefits of depth in neural networks*.
Hrushikesh N. Mhaskar and Tomaso Poggio. (2016) Deep vs. Shallow Networks: an Approximation Theory Perspective
Tomaso Poggioa, Andrzej Banburskia , and Qianli Liao (2019) Theoretical Issues in Deep Networks

Beyond approximation: Empirical Laws of Neural Scaling

- We’ve just seen how deep networks can overcome the curse of dimensionality for *structured functions*
    - they offer provably better approximation rates than shallow networks when the function has a compositional architecture
- These theorems give us a *theoretical foundation* for why deep networks are more powerful. But in practice, the success of deep learning has gone far beyond what these theorems predict—especially in large-scale models.
- This raises a natural question: *As we make models bigger*, train on *more data*, and use *more compute*, how does performance improve? This is not just a theoretical curiosity—it's become a central principle in modern machine learning.
- This brings us to **neural scaling laws**—empirical laws that describe how error decreases as a power law in model size, dataset size, or compute. While the approximation results give us asymptotic rates for a fixed function class, scaling laws describe behavior across model classes as we scale the system.
- In a sense, scaling laws are the *empirical counterparts* to the approximation bounds we just saw. Instead of asking ‘how complex a network do I need to approximate a fixed function?’, we ask ‘how does performance scale as I increase capacity?

Scaling Laws

Scaling laws are empirical regularities—typically **power laws**—that describe how a model's error-like metric (e.g., pre-training loss) changes **predictably** as we scale core resources under a fixed training *recipe* (same objective, architecture family, optimizer/schedule, and data distribution). In practice, single-factor "slices" (varying one resource while the others are not bottlenecking) look like straight lines on log–log plots.

A commonly used aggregate form for the *reducible* part of the loss is:

$$\boxed{L(N, D, C) \;\approx\; L_\infty \;+\; A\,N^{-\alpha} \;+\; B\,D^{-\beta} \;+\; E\,C^{-\gamma}}$$

where:

- L_∞ is the irreducible loss (noise floor),
- N = non-embedding parameters, D = training tokens, C = training compute (FLOPs),
- $A, B, E > 0$ are fit constants and $\alpha, \beta, \gamma > 0$ are **scaling exponents**. Single-factor slices then have slopes $-\alpha$, $-\beta$, and $-\gamma$ on log–log axes.

Why this matters. These laws let you **forecast** performance when scaling up and **plan budgets** (e.g., under $C \approx kND$ you can derive compute-optimal $N^*(C)$ and $D^*(C)$). They're descriptive (recipe-dependent), and can bend under data duplication/quality changes, optimization-regime shifts, or distribution shift.

- L: training/pretraining loss (e.g., cross-entropy)
- L_∞: irreducible loss (asymptote / noise floor)
- N: non-embedding parameters (model capacity)
- D: training tokens (effective after curation/dedup)
- C: training compute in FLOPs actually performed
- k: proportionality constant in $C \approx kND$ (depth, sequence length, efficiency)
- $A, B, E > 0$: fit constants ("amplitudes") for the N, D, and C terms
- α: parameter-scaling exponent (slope of $\log(L - L_\infty)$ vs $\log N$)
- β: data-scaling exponent (slope vs $\log D$)
- γ: compute-scaling exponent (slope vs $\log C$, single-factor slice)
- $\eta = \frac{\alpha\beta}{\alpha+\beta}$: envelope exponent for compute-optimal frontier $L^*(C)$

Data Scaling Laws

Equation of the Law

$$L(D) \approx L_\infty + B\,D^{-\beta}, \quad \text{so on log–log axes: slope} = -\beta.$$

As training tokens D grow; when model capacity and steps aren't limiting; the **reducible loss** $L - L_\infty$ follows a power law.

How to fit well

- Use reducible loss $L - L_\infty$ (estimate L_∞ jointly or from a floor).
- Vary D over ≥ 1 order of magnitude; keep N (params) and steps large enough.
- Deduplicate/curate; report CIs for β.

Interpretation (elasticity)

- More tokens reduce loss with predictable elasticity

$$\varepsilon_D = \frac{\partial \log(L - L_\infty)}{\partial \log D} = -\beta.$$

Pitfalls

- Heavy repetition / low quality data flatten β.
- Domain shift moves slope and the asymptote L_∞.

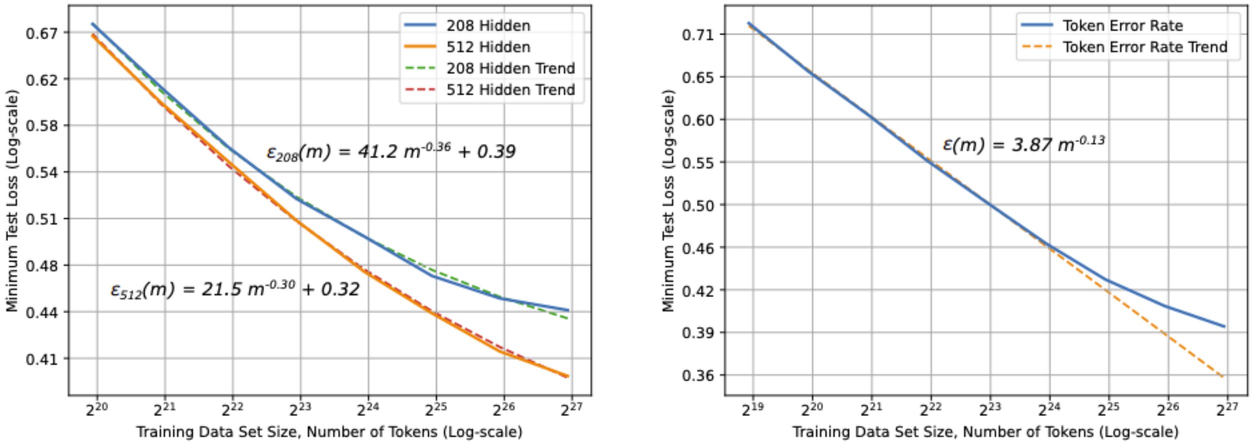


Figure 1: Neural machine translation learning curves. Left: the learning curves for separate models follow $\varepsilon(m) = \alpha m^{\beta_g} + \gamma$. Right: composite learning curve of best-fit model at each data set size.

Joel Hestness et al (2017), Deep Learning Scaling is Predictable, Empirically

Compute Scaling laws

Equation (single-factor slice)

$$L(C) \approx L_\infty + E C^{-\gamma}.$$

Compute constraint (decoder LM, fixed recipe)

$$C \approx k N D$$

where N = parameters, D = token exposures, and $k > 0$ absorbs depth/seq length/efficiency.

Compute-optimal allocation (derivation sketch)

Minimize

$$L(N, D) = AN^{-\alpha} + BD^{-\beta}$$

s.t. $ND = C/k$. Then

$$N^*(C) = \kappa_N C^{\frac{\beta}{\alpha+\beta}}, \qquad D^*(C) = \kappa_D C^{\frac{\alpha}{\alpha+\beta}},$$

and the **optimal envelope** becomes

$$L^*(C) = L_\infty + K C^{-\eta}, \qquad \eta = \frac{\alpha\beta}{\alpha+\beta}.$$

Inference-aware note

- If serving dominates TCO, training **smaller** N longer (larger D) can beat a huge model at fixed budget.
- Test-time compute (best-of-n, self-consistency) also "scales" quality at fixed N.

Ops checklist

- Keep optimizer/schedule fixed while fitting γ; regime changes bend the curve.
- Track achieved TFLOPs/MFU to avoid miscounting C.

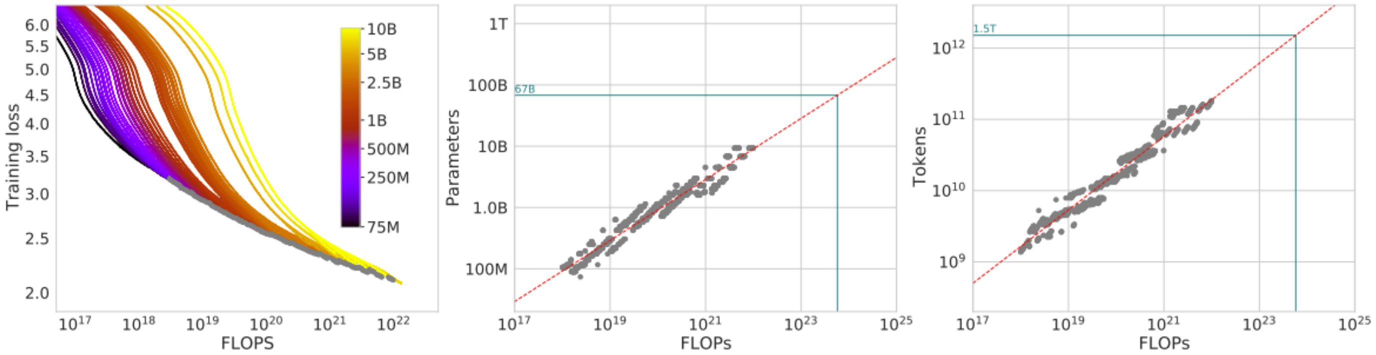


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

Jordan Hoffmann et al (2022) , [Training Compute-Optimal Large Language Models](#)

Parameters scaling laws

Equation (single-factor slice)

$$L(N) \;\approx\; L_\infty \;+\; A\,N^{-\alpha}, \quad \text{so on log–log axes: slope} = -\alpha.$$

Meaning

In the **capacity-limited** regime (enough tokens & steps), increasing N steadily reduces loss.

How to fit well

- Ensure enough tokens/steps (avoid undertraining); vary N over ≥ 1–2 orders.
- Fit $L - L_\infty$ vs $\log N$; bootstrap CIs for α.

Interpretation (elasticity)

$$\varepsilon_N \;=\; \frac{\partial \log(L - L_\infty)}{\partial \log N} \;=\; -\alpha.$$

Pitfalls

- Too few tokens/steps flatten the apparent $-\alpha$ slope ("big models don't help" = bottleneck elsewhere).
- Architecture/parametrization changes (e.g., depth/width, μP) shift constants and can nudge α.

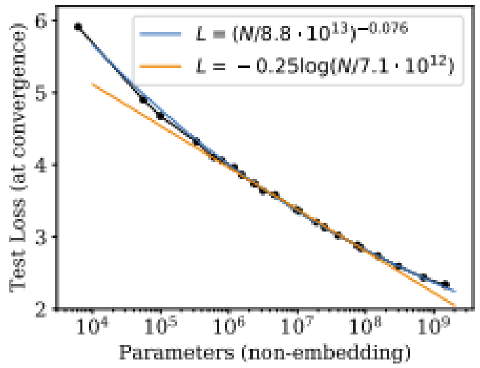


Figure 23 The trend for performance as a function of parameter count, $L(N)$, is fit better by a power law than by other functions such as a logarithm at a qualitative level.

Jared Kaplan et al (2020), Scaling Laws for Neural Language Models

What Universal Approximation Theorems Really Tell Us ?

Advantages:

They guarantee representational capacity.

Limitations:

- They do **not** explain why or how deep nets generalize.
- They do **not** address training dynamics or sample efficiency.

- So if representational power isn't enough, what explains generalization in overparameterized deep networks ?
- **That is what we will explore in** Optimization & Generalization in the second part of the lecture.

What Scaling Laws Really Tell Us ?

- **Advantages:**
  - They provide **empirical regularities**: loss typically falls as a **power law** with model size and data.
  - They guide **budget allocation**: show the **compute-optimal tradeoff** between parameters, data, and training time.
  - They enable **predictability**: small pilot runs can forecast returns at larger scale.
  - They highlight **diminishing returns** and when to spend on **data quality** instead of more parameters.

- **Limitations:**
  - They are **descriptive, not causal** explanations of why models work.
  - Exponents **depend on task, data, and architecture**; they can shift with changes or distribution shift.
  - They say **nothing about optimization stability** (learning rates, curvature, mode collapse).
  - They do **not guarantee generalization** or sample efficiency; they average over many confounders.

- **So if scaling alone isn't enough, what explains generalization and efficient training? That is what we will explore** next: optimization & generalization—implicit bias of SGD, flat vs sharp minima, margins, regularization & augmentation, data curriculum, and how architecture + data shape out-of-sample performance.

References

- George Cybenko. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4):303-314, 1989.
- Jared Kaplan et al (2020), Scaling Laws for Neural Language Models
- Joel Hestness et al (2017), Deep Learning Scaling is Predictable, Empirically
- Telgarsky, M. (2016). *Benefits of depth in neural networks*.
- Hrushikesh N. Mhaskar and Tomaso Poggio. (2016) Deep vs. Shallow Networks: an Approximation Theory Perspective
- Tomaso Poggioa, Andrzej Banburskia , and Qianli Liao (2019) Theoretical Issues in Deep Networks
- Hrushikesh N. Mhaskar and Tomaso Poggio (2016) , Deep vs. Shallow Networks: an Approximation Theory Perspective