Machine Learning Notes 12.16

December 27, 2025

1 AI4Protein: Structure, Ensemble, and Dynamics

Proteins are the central "workhorses" of the cell: sequences of amino acids fold into 3D structures that enable binding, catalysis, mechanical action, and ultimately biological function. These notes introduce the protein structure prediction problem and explain the core ideas behind AlphaFold2, focusing on (i) representations (sequence, multiple sequence alignment, pairwise geometry, rigid frames), (ii) the Evoformer feature-processing stack, and (iii) the SE(3)-aware structure module with Invariant Point Attention (IPA). We end with a brief discussion of why proteins are not rigid objects and why ensembles/dynamics are the next frontier.

1.1 From sequence to function: why structure matters

Biology is often summarized by a flow of information known as the *central dogma of molecular biology*: genetic material encodes amino-acid sequences, which fold into protein structures that carry out molecular functions (e.g., recognition or catalysis) and support organism-level functions (e.g., metabolism or host defense). In practice, structure is a key intermediate: knowing a sequence is not enough to deduce how a protein binds a partner, positions catalytic residues, or forms a transport pathway.

Historically, structures were determined by experimental pipelines such as X-ray crystallography, NMR, or cryo-EM. While these methods are powerful, they are time-consuming and resource-intensive for many targets. Modern AI approaches changed the landscape by predicting accurate structures from sequence in minutes to hours, which dramatically lowers the barrier to structure-enabled hypothesis generation and design.

1.2 Why protein folding is difficult

There are two intertwined sources of difficulty. One is the combinatorial sequence space and long-range dependence. A protein of length L over the standard 20 amino acids has a discrete sequence space of size

$$|\mathcal{S}| = 20^L. \tag{1}$$

Even for $L \approx 100$, this number is astronomically large. At the same time, folding and stability are governed by nonlocal interactions: residues distant along the chain can be adjacent in 3D and strongly coupled.

Another obstacle is the limited supervised structural data. Supervised learning benefits from abundant labeled data, but experimentally solved protein structures are far fewer than known sequences. This "data bottleneck" was a major obstacle for decades.

1.3 Representing protein structure

When predicting structure, it helps to choose a representation that respects physics and symmetry.

1.3.1 Backbone vs. side chains

A common abstraction separates:

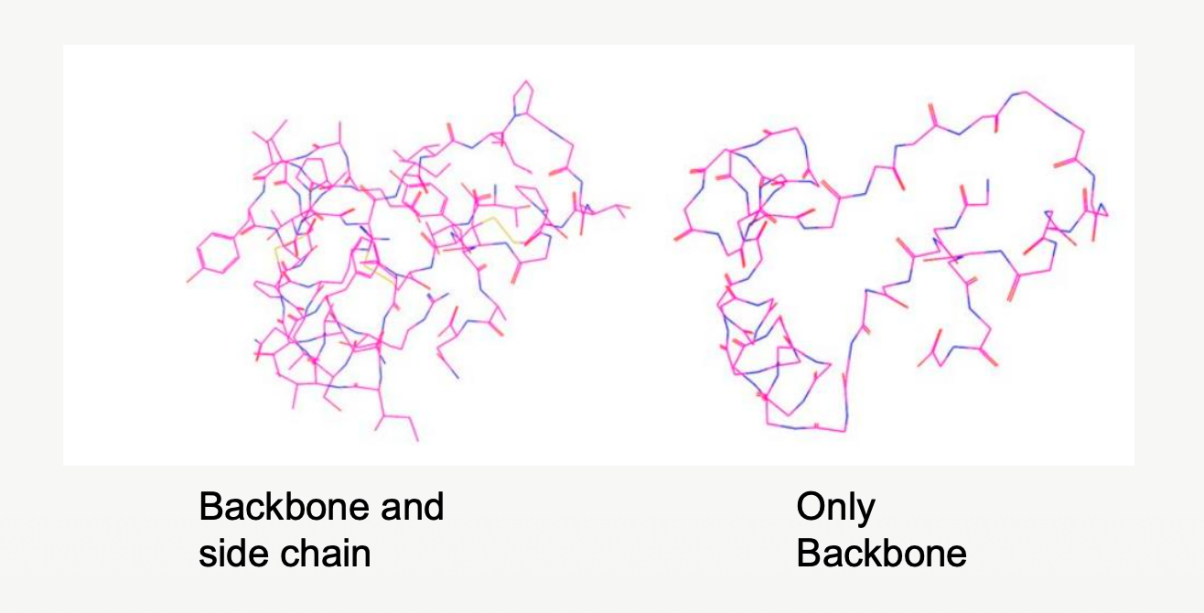


Figure 1: Protein structure representation (lecture PPT p.10).

- **Backbone**: the repeating main-chain atoms that determine the overall fold;
- **Side chains**: residue-specific atoms whose conformations (rotamers) refine packing and chemistry.

Many pipelines first predict an accurate backbone and then refine side chains.

1.3.2 Rigid frames and SE(3) transforms

Rather than predicting all-atom coordinates directly, AlphaFold2 uses a per-residue local coordinate frame (a rigid-body transform) to describe backbone placement. A rigid transform is

$$T = (R, t), \quad R \in SO(3),\ t \in \mathbb{R}^3, \tag{2}$$

acting on a point $x \in \mathbb{R}^3$ as

$$T \circ x = Rx + t. \tag{3}$$

Composition of transforms follows

$$(R_1, t_1) \circ (R_2, t_2) = (R_1 R_2,\ R_1 t_2 + t_1). \tag{4}$$

This formalism is not just mathematical style: it encodes the fundamental fact that the absolute global coordinate system is arbitrary. Good models should behave consistently under global rotations/translations of a structure.

1.4 Evaluation: CASP and the GDT family

Progress in structure prediction is tracked by community benchmarks. The CASP competition evaluates blind predictions for proteins whose experimental structures are not yet public. Informally, the CASP competition uses the "Global Distance Test" (GDT) metric that behaves as follows: given a predicted structure and a reference structure, first superimpose them. For a distance threshold τ, compute the fraction of residues whose Cα atoms are within τ of the reference. GDT scores summarize this fraction across several thresholds. The superimposing process is done iteratively. The key idea is to score structural similarity after removing arbitrary global rigid motions via superposition.

1.5 AlphaFold2 at a high level

AlphaFold2 is an end-to-end pipeline that maps sequence-derived features to 3D structure. A helpful mental model is three stages3:

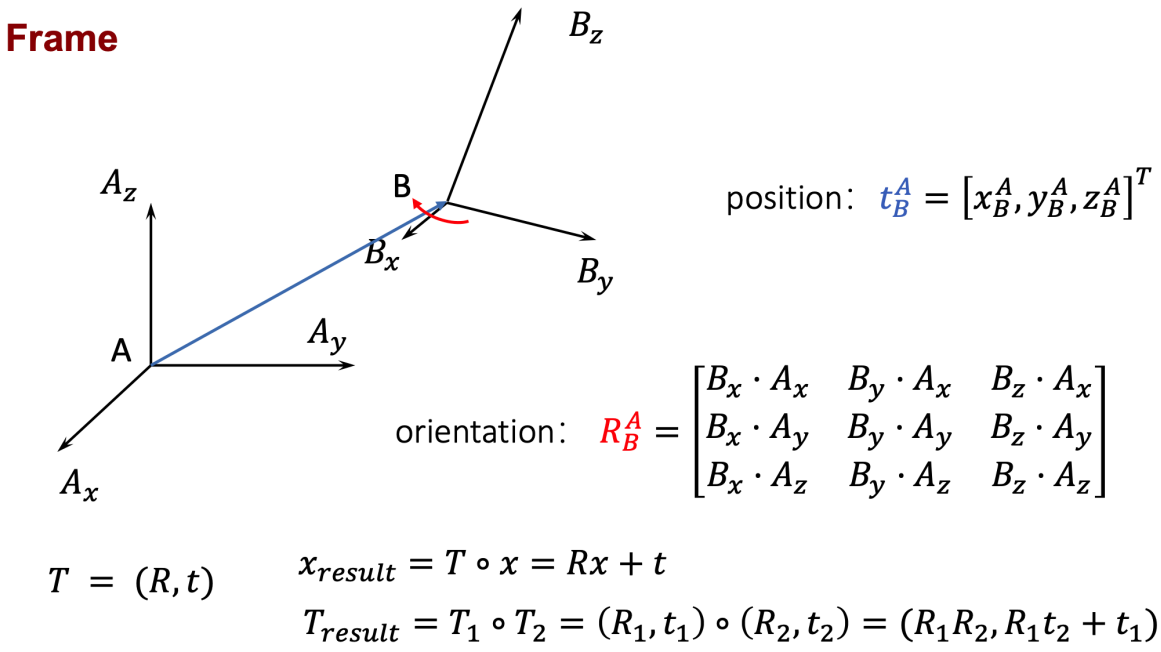


Figure 2: Rigid transformations (lecture PPT p.13).

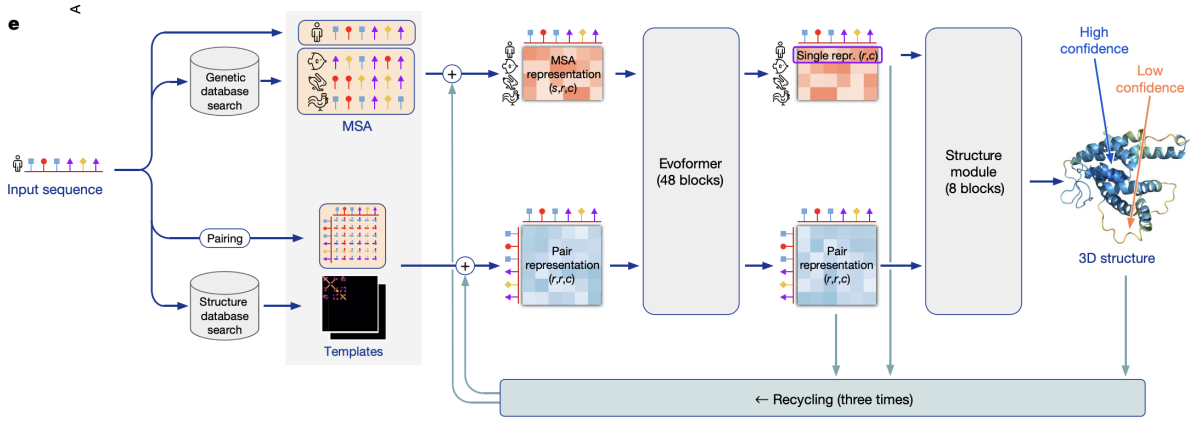


Figure 3: AlphaFold Pipeline (Jumper et al. [2021])

1. **Embedding / Feature Construction**: Generates rich representations by integrating the target amino acid sequence, database search results, and structural templates.
2. **Evoformer**: A deep, transformer-like architecture that refines protein features through iterative updates along two coupled representation tracks.
3. **Structure Module**: Translates the refined features into three-dimensional atomic coordinates by predicting backbone frames and side-chain torsions using geometry-aware attention mechanisms.

The architecture is also characterized by *recycling*: the model can re-use its own outputs as inputs for multiple refinement passes.

A note on scaling. In addition to standard attention costs (quadratic in sequence length), AlphaFold2 performs operations on the $N_{\text{res}} \times N_{\text{res}}$ pair grid, including triangle-style updates that can scale cubically in N_{res}. Practical training therefore uses strategies such as cropping long sequences and limiting the number of MSA sequences per example.

1.6 Inputs and embeddings

1.6.1 Target sequence features

Two basic inputs are standard in transformer-style sequence models:

- a categorical encoding of each residue (e.g., one-hot over 20 amino acids plus an "unknown" token)
- a residue index / positional signal.

1.6.2 Multiple sequence alignment (MSA) pipeline

AlphaFold2 extracts evolutionary information by searching large sequence databases for homologs, then building a multiple sequence alignment (MSA). A practical pipeline is:

1. **Search**: retrieve many similar sequences.
2. **Subsampling/deletion**: remove overly redundant blocks (to reduce bias and cost).
3. **Clustering**: pick representative sequences.
4. **Feature computation**: build per-position statistics and related MSA-derived features.

MSA processing is critical because it converts the "too little structure data" problem into an advantage: sequence databases are enormous compared to solved structures.

1.6.3 Two coupled representations

AlphaFold2 uses two main learned tensors:

- **MSA representation** $\in \mathbb{R}^{N_{\text{seq}} \times N_{\text{res}} \times c_m}$ (Token embeddings for each aligned sequence position);
- **Pair representation** $\in \mathbb{R}^{N_{\text{res}} \times N_{\text{res}} \times c_z}$ (Embeddings for residue–residue relationships).

The pair representation is especially important because it is the natural home for predicted geometry (distances, orientations, contact patterns) and serves as a bias term in several attention operations.

1.7 Evoformer: learning from evolution and geometry

The Evoformer is a deep stack of blocks that update the MSA and pair tracks in an intertwined way.

1.7.1 MSA updates: row-wise and column-wise attention

Think of the MSA representation as a matrix (sequences $\times$ positions) with feature channels. Two complementary attentions are used:

- **Row-wise self-attention**: operates along positions within each aligned sequence, enabling long-range dependencies along the protein chain.
- **Column-wise self-attention**: operates across sequences at a fixed position, pooling evolutionary variation.

Row-wise attention typically incorporates a *pair bias* derived from the pair representation, injecting geometric context into sequence processing.

1.7.2 MSA-to-pair: outer product mean

To update pair features using MSA information, AlphaFold2 uses a form of outer-product aggregation (followed by averaging) across sequences. Intuitively, if two residue positions co-vary across evolution, their embeddings should interact to strengthen a relationship in the pair track.

1.7.3 Pair updates: triangle operations

Pair features form a complete graph over residues (a dense $N_{\text{res}} \times N_{\text{res}}$ grid). AlphaFold2 performs specialized updates that reflect how 3D constraints propagate through triangles in a graph:

- **Triangle multiplicative updates**: combine edges (i, k) and (k, j) to update (i, j), analogous to message passing through an intermediate residue.

- **Triangle self-attention**: an attention mechanism over the third node k, capturing multiple geometric "paths" between residues.

These operations encourage consistency among residue relationships and are widely viewed as one of the distinctive engineering choices behind AlphaFold2's performance.

1.8 Structure module and Invariant Point Attention

After Evoformer processing, the model must produce a concrete 3D structure.

1.8.1 From features to frames and torsions

The structure module first constructs a *single* representation per residue (derived from MSA features), then iteratively predicts:

- backbone rigid frames T_i for each residue i,
- side-chain torsion angles (often denoted $\chi_1, \ldots, \chi_4$ depending on residue type).

Given T_i and the torsions, standard chemistry templates can reconstruct atom coordinates.

1.8.2 Invariant Point Attention (IPA): the key geometric ingredient

Standard attention compares query and key vectors using dot products. IPA augments this with *points* attached to residues and measured in 3D, while ensuring invariance to global rigid motions.

One way to view IPA is:

1. Predict query/key/value feature vectors as usual.
2. Predict a small set of 3D query/key points in each residue's local frame.
3. Map these points into a common frame using the current rigid transforms T_i.
4. Define attention logits using both feature similarity and *Euclidean distances* between transformed points.

Because Euclidean distances are preserved under a shared global rigid motion, the distance term is invariant. Moreover, when outputs are mapped back to local frames, a global motion cancels out. This yields a structure-update rule that is consistent with SE(3) symmetries: global rotations/translations of the current structure do not change the internal reasoning.

1.9 Training objectives and confidence estimation

AlphaFold2 trains with a mixture of structure and auxiliary losses.

1.9.1 Structure loss

A major component is the *Frame Aligned Point Error* (FAPE), which measures discrepancies between predicted and true points after aligning via local frames. FAPE is attractive because it is sensitive to local geometry while avoiding the need to commit to an arbitrary global frame.

Side-chain accuracy is encouraged through torsion-angle and atom-position terms, and additional chemical constraints penalize physically implausible geometries.

1.9.2 Auxiliary losses

Auxiliary heads improve training stability and representation quality:

- **Distogram prediction**: predict distance distributions between residue pairs.
- **Masked-MSA**: reconstruct masked residues in the MSA (a self-supervised objective).
- **Confidence prediction**: estimate per-residue confidence (pLDDT), useful for downstream interpretation.

1.10 Ablations and engineering lessons

AlphaFold2 is a deeply engineered system; many components contribute materially to accuracy.

Common takeaways from ablation studies include:

- **MSA information matters**: removing or weakening MSA processing reduces accuracy.
- **Geometry-aware modules matter**: triangle updates and IPA are not cosmetic; they encode inductive biases that help generalization.
- **End-to-end gradients matter**: coupling the whole pipeline (instead of a modular, non-differentiable assembly) improves results.

At a broader level, AlphaFold2 illustrates a pattern in scientific ML: progress often comes from combining domain knowledge (symmetries, geometry, chemistry) with large-scale optimization and careful systems design.

1.11 Beyond single structures: ensembles and dynamics

Proteins are not rigid objects. Thermal motion and conformational heterogeneity mean that a single "static" structure is often an approximation. Functional mechanisms—transport, signaling, catalytic cycles, binding and unbinding—may rely on rare or transient conformations.

From a statistical mechanics viewpoint, one may care about an equilibrium distribution over conformations and derived quantities such as free energies. This motivates methods that go beyond predicting one best structure:

- **Ensemble prediction**: generate multiple plausible conformations consistent with the sequence and physical constraints.
- **Dynamics**: model time-dependent trajectories or transitions between metastable states.
- **MSA-free prediction**: in regimes with few homologs (or for speed), alternative approaches reduce dependence on MSAs; this also connects to questions about how to model diverse conformational ensembles.

2 AlphaFold Meets Flow Matching for Protein Ensembles

This section mainly discusses Jing et al. [2024].

2.1 Motivation: Proteins as Structural Ensembles

Protein function often depends on *conformational heterogeneity*: proteins occupy structural ensembles with multiple states and fluctuations rather than a single rigid structure. A single-state predictor (e.g.,AlphaFold Jumper et al. [2021]) can be extremely accurate for a representative structure, but it does not directly provide the underlying ensemble.

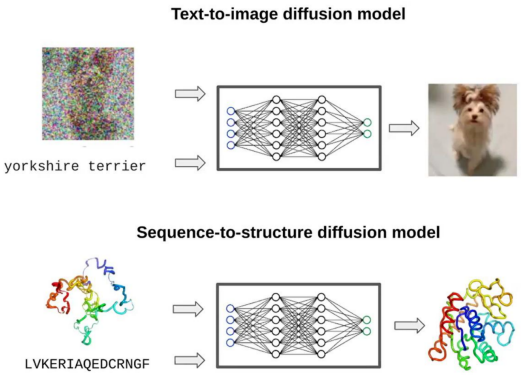
Existing practice: MSA ablation / subsampling. A common way to induce diversity is to perturb the multiple sequence alignment (MSA) at inference time (subsampling, mutagenesis, clustering), producing different predictions. However:

- It is tied to MSA-based pipelines and does not naturally transfer to predictors relying on protein language models (PLMs) such as ESMFold.
- It is an *inference-time hack*: it does not provide a general training framework to learn from ensembles beyond the PDB (e.g., MD trajectories).

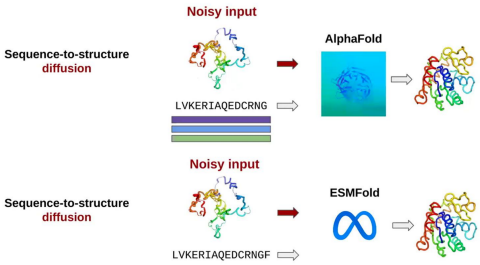
2.2 Problem statement

Given a protein sequence (and optionally MSA/template-like inputs), we want a *distribution* over 3D structures:

$$p(x \mid A), \qquad x \in \mathbb{R}^{3\times N},$$



(a) Diffusion analogy (lecture PPT p.50–54).



(b) Injecting noisy structure input (lecture PPT p.55–58).

Figure 4: Lecture explanation: predictive models can be repurposed as denoising networks, enabling generative sampling.

where A is the amino-acid sequence and N is the number of residues. The output is an ensemble (samples) rather than a single point estimate.

The lecture emphasizes a key conceptual distinction:

> **Ensemble, not trajectory.** The goal is to model equilibrium-like structural variability (a distribution), not to reproduce time-ordered MD trajectories.

2.3 Core idea: turn AlphaFold into a denoiser, then into a generator

The paper leverages a standard generative-modeling recipe: if we can train a network to *denoise* (map noisy x_t toward clean x_1), then iterative denoising defines a sampling procedure.

2.3.1 AlphaFold as a denoising model

The lecture uses a text-to-image analogy:

- A diffusion (or flow) model repeatedly denoises an image from noise to data.
- Similarly, if AlphaFold can take a *noisy structure* as input and predict a *clean structure*, then repeated denoising yields a *sequence-to-structure generative model*.

2.3.2 Minimal architectural modification

AlphaFold already has a notion of *template input*. The method adds an embedding module (similar to template embedding) so the model can ingest a noisy structure x_t plus a time embedding t; the rest of the AlphaFold/ESMFold trunk is reused. Conceptually:

$$\widehat{x}_1 = f_\theta(A, M, x_t, t),$$

where M denotes MSA/PLM-derived context (depending on AlphaFold vs ESMFold).

2.4 Flow matching formulation

2.4.1 Flow matching in one picture

Flow matching Lipman et al. [2023] defines a probability path that transports samples from a prior distribution to the data distribution, by learning a vector field (or equivalently, a denoising target along the path).

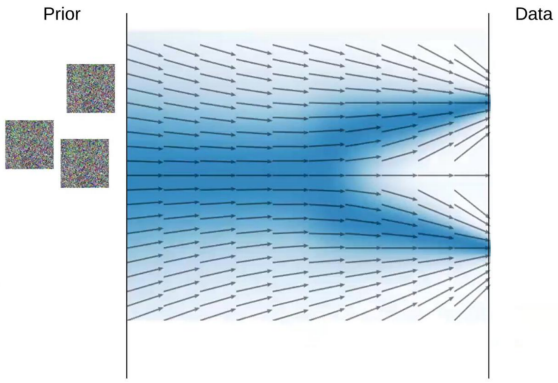


Figure 5: Flow matching intuition (lecture PPT p.59–61): transport from prior to data along an interpolant, guided by a denoising model.

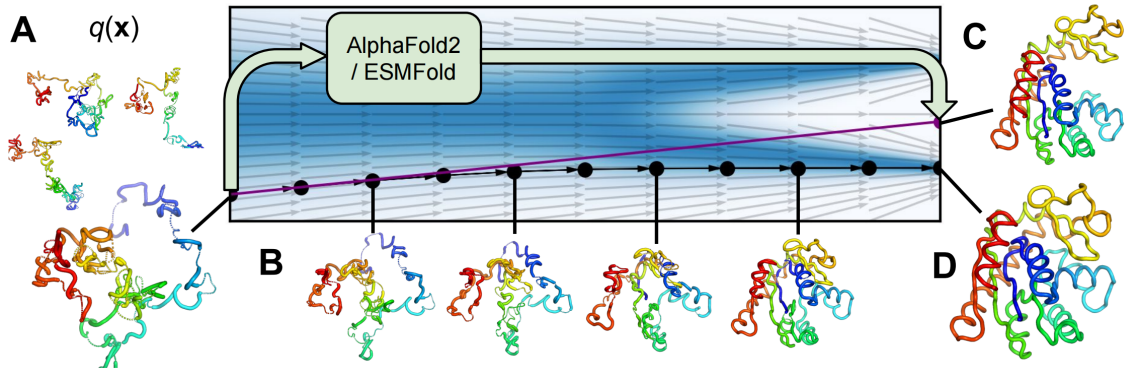


Figure 6: Conceptual overview of AlphaFLOW/ESMFLOW (paper Fig.1): sample from polymer prior and iteratively denoise/refine to a structure sample.

2.4.2 Conditional probability path via interpolation

A simple and effective path is linear interpolation between a prior sample x_0 and a data sample x_1:

$$x_t = (1-t)x_0 + tx_1, \quad t \in [0,1].$$

In standard flow matching, this induces a conditional vector field

$$u_t(x \mid x_1) = \frac{x_1 - x}{1-t}.$$

Rather than predicting the vector field directly, the paper reparameterizes the learned field via a denoiser $\widehat{x}_1(x,t)$:

$$\widehat{v}(x,t) = \frac{\widehat{x}_1(x,t) - x}{1-t}.$$

This mirrors how diffusion models train a network to predict a clean target from noisy input.

2.4.3 Protein-specific prior: harmonic (polymer-like) prior

Protein backbones are polymeric; the paper uses a harmonic prior (over Cβ coordinates; Cα for glycine) to keep noisy samples physically plausible:

$$q(x) \propto \exp\left(-\frac{\alpha}{2}\sum_{i=1}^{N-1}\|x_i - x_{i+1}\|^2\right).$$

This ensures x_0 and intermediate x_t resemble a connected chain rather than arbitrary Gaussian noise.

2.4.4 SE(3) invariance and the quotient-space fix

A naive MSE denoising objective is incompatible with AlphaFold-style invariances: the model ingests *SE(3)-invariant* features (e.g., distograms), so it cannot recover absolute orientation/translation needed by a plain coordinate MSE loss.

The paper resolves this by operating in the quotient space $\mathbb{R}^{3N}/SE(3)$ and:

- aligning structures via RMSD (rigid superposition) before interpolation,
- using a squared FAPE loss as a metric-compatible objective, effectively guiding the vector field towards the target in the quotient geometry.

Practically, training samples (x_0, x_1) are RMSD-aligned before forming x_t, and training uses FAPE^2 on the full all-atom prediction.

2.5 Algorithms

2.5.1 Training

For each training structure S with sequence A (and MSA M when applicable):

1. extract Cβ coordinates x_1 from S,
2. sample $x_0 \sim q(\cdot)$ from the harmonic prior of matching length,
3. RMSD-align x_0 to x_1,
4. sample $t \sim \text{Uniform}[0, 1]$, set $x_t = (1-t)x_0 + tx_1$,
5. predict $\widehat{S} = f_\theta(A, M, x_t, t)$ and minimize $\text{FAPE}^2(\widehat{S}, S)$.

2.5.2 Inference (sampling)

Initialize $x_0 \sim q(\cdot)$ and iteratively update x_t toward the model's denoised prediction $\widehat{x}_1$ using a discrete-time schedule (e.g., 10 steps). The final forward pass returns an all-atom structure sample.

Controlling diversity. The paper and lecture both highlight that truncating/merging early steps (or using distilled one-step models) allows trading off precision vs diversity, analogous to tuning MSA depth in subsampling baselines.

2.6 Experiments and results

2.6.1 PDB ensembles: precision–diversity trade-off

Setup: fine-tune AlphaFold/ESMFold on PDB structures under the flow-matching objective to obtain AlphaFLOW / ESMFLOW. Evaluate on heterogeneous proteins with multiple experimentally observed conformations. Metrics include:

- **Precision**: how close each prediction is to the nearest crystal structure (e.g., lDDT-Cα).
- **Diversity**: dissimilarity among generated samples.
- **Recall**: how well generated samples cover known conformations.

Key result: AlphaFLOW traces a *better precision–diversity frontier* than MSA subsampling; it can increase diversity without drifting as far away from true conformations.

2.6.2 MD ensembles (ATLAS): learning beyond PDB

The method is further fine-tuned on ATLAS all-atom MD ensembles, enabling evaluation against simulated conformational distributions (with test proteins structurally dissimilar to training).

The paper groups evaluations into three increasing-difficulty questions:

1. **Flexibility prediction:** does the ensemble diversity correlate with MD flexibility (pairwise RMSD, RMSF)?

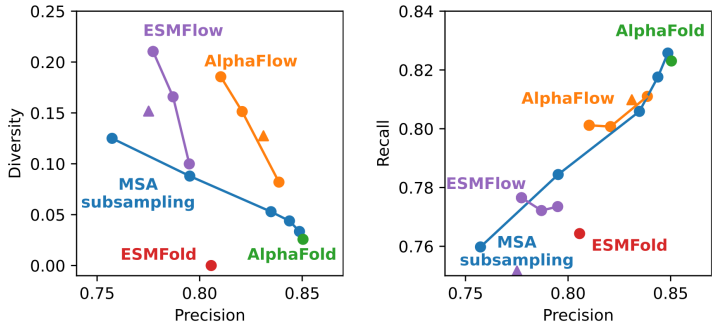


Figure 7: PDB evaluation (paper Fig.3): AlphaFLOW vs MSA subsampling on precision–diversity and precision–recall curves.

		AlphaFLOW-MD		MSA subsampling					AFMD+Templates	
		Full	Distilled	32	48	64	256	AlphaFold	Full	Distilled
Predicting flexibility	Pairwise RMSD (= 2.90)	**2.89**	1.94	4.40	2.34	1.67	0.72	0.58	2.18	1.73
	Pairwise RMSD r ↑	**0.48**	**0.48**	0.03	0.12	0.22	0.15	0.10	0.94	0.92
	All-atom RMSF (=1.70)	**1.68**	1.28	5.38	2.29	1.17	0.49	0.31	1.31	1.00
	Global RMSF r	**0.60**	0.54	0.13	0.23	0.29	0.26	0.21	0.91	0.89
	Per-target RMSF r	**0.85**	0.81	0.51	0.52	0.51	0.55	0.52	0.90	0.88
Distributional accuracy	Root mean $\mathcal{W}_2$-dist. ↓	**2.61**	3.70	6.15	5.32	4.28	3.62	3.58	1.95	2.18
	↪ Translation contrib. ↓	**2.28**	3.10	5.22	3.92	3.33	2.87	2.86	1.64	1.74
	↪ Variance contrib. ↓	**1.30**	1.52	3.55	2.49	2.24	2.24	2.27	1.01	1.25
	MD PCA $\mathcal{W}_2$-dist. ↓	**1.52**	1.73	2.44	2.30	2.23	1.88	1.99	1.25	1.41
	Joint PCA $\mathcal{W}_2$-dist. ↓	**2.25**	3.05	5.51	4.51	3.57	3.02	2.86	1.58	1.68
	% PC-sim > 0.5 ↑	**44**	34	15	18	21	21	23	44	43
Ensemble observables	Weak contacts J ↑	**0.62**	0.52	0.40	0.40	0.37	0.30	0.27	0.62	0.51
	Transient contacts J ↑	**0.41**	0.28	0.23	0.26	0.27	0.27	0.28	0.47	0.42
	Exposed residue J ↑	**0.50**	0.48	0.34	0.37	0.37	0.33	0.32	0.50	0.47
	Exposed MI matrix ρ ↑	**0.25**	0.14	0.14	0.11	0.10	0.06	0.02	0.25	0.18

Figure 8: MD evaluation summary (paper Table 1): AlphaFLOW(-MD) vs MSA subsampling across flexibility, distributional metrics, and ensemble observables.

2. **Distributional accuracy:** are atomic position distributions accurate (e.g., Wasserstein-type distances; PCA-space comparisons)?
3. **Ensemble observables:** does the model reproduce higher-order properties such as intermittent contacts and solvent exposure statistics?

Key result: AlphaFLOW substantially outperforms MSA subsampling on distributional and observable-level metrics, indicating it captures meaningful ensemble structure rather than merely injecting noise.

2.7 Discussion

2.7.1 Why this is conceptually clean

- **Training-time generative framework:** not just inference-time MSA perturbations.
- **Model reuse:** minimal changes; leverages AlphaFold/ESMFold accuracy as a denoiser.
- **Generalizes across inputs:** works for both MSA-based (AlphaFold) and PLM-based (ESMFold Lin et al. [2023]) predictors.
- **Ensembles beyond PDB:** can train on MD datasets to approximate equilibrium distributions and observables.

2.8 Limitations / open questions

- The output is an **ensemble** (samples) but not a **time-ordered trajectory**; dynamics/kinetics are not modeled directly.

- Evaluations are bounded by MD timescales in ATLAS Vander Meersche et al. [2023]; slower conformational transitions remain challenging.
- Sampling requires multiple forward passes (mitigated by distillation), so runtime remains a practical consideration.

3 Classical Learning Theory vs Theory for LLM

3.1 Theory Falls Far Behind Practice

In the rapid evolution of deep learning and large language models, theoretical understanding consistently lags behind empirical advancements. While classical learning theory provides foundational principles for shallow networks and i.i.d. data, modern LLMs operate under fundamentally different paradigms that challenge traditional assumptions.

3.2 Expressiveness

3.2.1 Classical Theory: Universal Approximation

- **Theoretical Foundation:** Early neural network theory established the *universal approximation theorem*, proving that even shallow networks with sigmoidal activations can approximate any continuous function given sufficient width.
- **Cybenko's Contribution:** George Cybenko's seminal work *Approximation by superpositions of a sigmoidal function* (1989) provided rigorous mathematical foundations for neural network expressivity.
- **Limitations:** While theoretically powerful, these guarantees assume infinite width and don't address practical training dynamics or computational constraints.

3.2.2 LLM Theory: AutoRegressive and Non-i.i.d.

- **AutoRegressive Nature:** Modern LLMs process sequences *auto-regressively*, where each token prediction depends on previous tokens, creating complex conditional dependencies.
- **Non-i.i.d. Data:** Unlike classical assumptions of independent identically distributed data, LLM training data exhibits strong temporal, semantic, and contextual dependencies.
- **Sequential Generation:** The autoregressive framework enables coherent multi-step generation but introduces challenges in theoretical analysis due to the compounding of errors.
- **Contextual Dependencies:** The attention mechanism in Transformers creates dense interactions between all tokens, resulting in highly non-linear and non-i.i.d. internal representations.

3.3 Optimization

3.3.1 Classical Theory: Convergence Analysis

- **Gradient-Based Optimization:** Classical theory focuses on gradient descent dynamics, analyzing convergence rates and stability under convexity assumptions.
- **Gradient Explosion/Vanishing:** Early deep networks faced optimization challenges from unstable gradients during backpropagation, hindering training of deeper architectures.
- **Convergence Guarantees:** Traditional analysis assumes multiple passes over data and provides convergence bounds under various regularity conditions.

3.3.2 LLM Theory: Limited Data Passes and New Challenges

- **Few Epochs Training:** Modern large-scale models often see training data only **3-4 times**, making traditional convergence analysis less meaningful.
- **Emergent Optimization Behaviors:** LLMs exhibit unexpected optimization phenomena that classical theory doesn't adequately explain:

  - **Scaling Laws:** Performance improves predictably with model size, data, and compute despite theoretical complexity.
  - **Grokking:** Models suddenly generalize after prolonged training on seemingly memorized data.
  - **Double Descent:** Risk decreases then increases then decreases again with model capacity.
- **New Research Directions:**
  - **Incremental Learning:** How to effectively incorporate **new data** without catastrophic forgetting.
  - **Efficient Adaptation:** Methods for rapid learning from limited data passes.
  - **Dynamic Optimization:** Algorithms that handle continuously evolving data distributions and objectives.
- **Practical Reality:** Reference paper `https://arxiv.org/pdf/1709.02540` argues that analyzing convergence for models trained only 3-4 epochs has limited practical relevance today.

3.4 Generalization

3.4.1 Classical Theory: Uniform Convergence and VC Dimension

- **Uniform Convergence:** Classical theory bounds generalization error through uniform convergence of empirical risk to expected risk.
- **VC Dimension and Rademacher Complexity:** These measures characterize model capacity and provide generalization bounds independent of data distribution.
- **Well-Studied Gap:** The discrepancy between training and test performance has been extensively analyzed under i.i.d. assumptions with fixed task distributions.
- **Structural Risk Minimization:** Theory guides model selection through bias-variance tradeoffs and regularization.

3.4.2 LLM Theory: Multi-Task, Open-Ended Generalization

- **Thousands of Diverse Tasks:** LLMs must generalize across **thousands of diverse, non-stationary tasks** that emerge dynamically from user interactions.
- **Open-Ended Evaluation:** Performance assessment requires measuring **multi-task, few-shot, and zero-shot generalization** rather than single-task accuracy.
- **Non-Stationary Distributions:** Task distributions shift continuously as models interact with users and environments, violating traditional i.i.d. assumptions.
- **Emergent Capabilities:** LLMs exhibit unexpected generalization abilities not present in smaller models, including:
  - **Few-shot Learning:** Rapid adaptation from minimal examples without parameter updates.
  - **Instruction Following:** Understanding and executing complex instructions beyond training distribution.
  - **Chain-of-Thought Reasoning:** Multi-step reasoning through explicit intermediate steps.
- **Theory Gap:** While traditional generalization theory is mature, new frameworks are urgently needed to understand LLM behavior in open-ended, multi-task environments where classical measures like VC dimension fail to capture true capabilities.

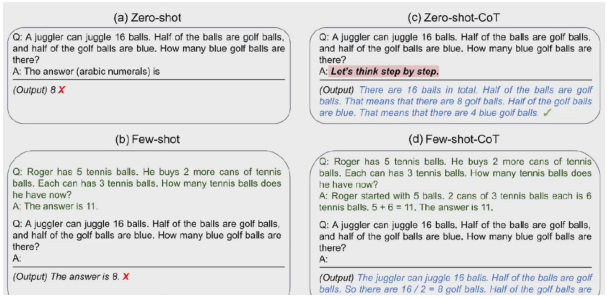


Figure 9: Chain of thought example

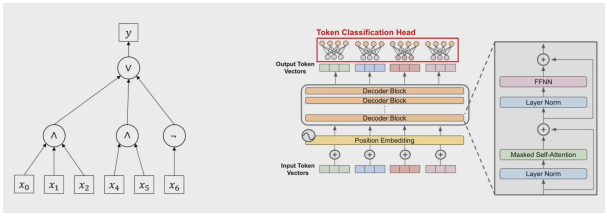


Figure 10: Circuit and Transformer

4 Chain-of-Thought Reasoning from the Perspective of Expressive Capability

4.1 The Evolution of Model Problem-Solving

The initial release of ChatGPT in 2022 demonstrated capabilities in tasks like writing short articles or replying to emails, yet its application performance was notably poor. It frequently erred on basic arithmetic and general problem-solving. Subsequent rapid advancements, particularly following the merger of Google Brain and DeepMind, led to significantly improved models like Gemini. The chain-of-thought (CoT) method emerged to address these early shortcomings. The following simple arithmetic problem exemplifies a task that large models initially failed—a fact that seems remarkable given the immense progress over three years.

4.2 The Emergence and Impact of Chain-of-Thought

In 2022, researchers from Google Brain discovered that modifying the prompt to include "Letś think step by step" led to significant performance improvements. (Wei et al. [2022]) Prior to CoT, models generated answers in a single step. Another related technique is few-shot, or in-context, learning, where the model is provided with example solutions. Even for modern models, bypassing a reasoning trace for moderately difficult problems (e.g., solving linear equations) results in a substantially higher error rate compared to employing a step-by-step thinking mode. Contemporary CoT is more sophisticated, involving training on human-written reasoning traces and techniques like reinforcement learning. However, considering a model purely with and without CoT prompting reveals a fundamental difference in expressive capability, which we aim to understand.

4.3 Analyzing Modern Model Capability

Understanding a modern large model requires analyzing not just the Transformer structure in isolation, but its use in an auto-regressive manner *plus* the chain-of-thought process. The Transformer structure maps an input sequence of length N tokens into an embedding space with positional encoding, processes it through stacked blocks, and produces an output. Auto-regressive generation appends this output to the input, creating a sequence of length $N+1$, and iteratively feeds it back until a stop token is generated.

4.4 Transformers as Circuits: A Complexity Perspective

We adopt a technical assumption of log precision, where a Transformer uses $O(\log n)$ bits of precision for its inputs and intermediate values. To understand the Transformer's essence, we compare it to a logical

circuit diagram. A Transformer is fundamentally a type of computational circuit. Circuit complexity studies the resources—such as circuit size (number of gates) and depth (number of layers)—required to compute problems, contrasting with classical computational complexity based on Turing machines. Relevant complexity classes include NC and its subclasses (e.g., NC^0, NC^1), where the superscript relates to depth. The class TC^0 consists of problems computable by constant-depth circuits using threshold gates (which activate based on a weighted sum of inputs) alongside basic gates. A key point is that a Transformer, in its essence, can be understood as a circuit within the TC^0 class.

5 Transformer with Chain-of-Thought Reasoning: beyond TC^0 Expressive Capability

5.1 The Expressiveness of Transformer is Equivalent to the Circuit Complexity TC^0

It can be easily understood that the ReLU function can serve as the threshold gate, and transformer is intuitively similar to the TC^0 circuit. Given a certain circuit within the TC^0 class, it can be simulated by a transformer, and vice versa.

The connected circuits perform parallel computation, where the order of inputs has no impact on the computation result. Same with the circuit, the transformer process the input data using a parallel structure, through we know it receives and executes serial tasks. To deal with serial tasks in parallel, positional encoding is introduced to make transformer aware of the serial order of inputs.

Thus, for a transformer with constant depth, its expressive capability is exactly TC^0, and handling computations out of TC^0 class is hard to accomplish.

5.2 Arithmetic Operations: Simple as Serial computations but complex as Parallel Computations

Unfortunately, the complexity of the four arithmetic operations is not in TC^0 class, so you can never get perfect results of even the simplest arithmetic operation questions with shallow transformers. The reason behind is that arithmetic operations are inherently serial computations and they cannot be properly parallelized. So it generalizes, as most planning and reasoning problems' complexities are above TC^0, transformer do not have capability to directly generate the solution of these math and reasoning problems unless parameter is super-polynomial.

5.3 Power of Chain-of-Thought: Increases Expressive Capability

As transformer is auto-regressive, adding its outputs into inputs continuously, it is considered that the usage of chain-of-thought makes transformer be called repeatedly, improving its expressive capability far above TC^0.

Constant-size transformer have capacity to generate the final solution step-by-step auto-regressively, and with chain-of-thought it can solve P-complete problems.

References

Bowen Jing, Bonnie Berger, and Tommi Jaakkola. Alphafold meets flow matching for generating protein ensembles. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

John Jumper, Richard Evans, Alexander Pritzel, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Zeming Lin, Halil Akin, Roshan Rao, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Yannick Vander Meersche et al. Atlas: a database of protein conformational ensembles from molecular dynamics simulations. *Nucleic Acids Research*, 51(D1):D323–D329, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in neural information processing systems (NeurIPS 2025)*, volume 35, pages 24824–24837, 2022.