REFLECTION PRETRAINING ENABLES TOKEN-LEVEL SELF-CORRECTION IN *De Novo* PEPTIDE SEQUENCING

Anonymous authors
Paper under double-blind review

ABSTRACT

De novo peptide sequencing infers a peptide's amino-acid sequence directly from its tandem mass spectrum (MS/MS). State-of-the-art models decode the peptide autoregressively, one residue at a time, and cannot revise a residue once it has been emitted: a single misread residue, caused for instance by a missing fragment ion, corrupts the whole peptide. We propose *reflection pretraining*, which augments the amino-acid vocabulary with a reflection token ⟨REFLECT⟩ and trains the decoder to revise its own residues during generation. *De novo* sequencing is unusually well suited to this: every training spectrum is paired with a gold-label peptide, so the position and identity of an injected error are known exactly, and the spectrum itself (the fragment-ion ladder and the precursor mass) contains the evidence needed to verify each residue. During pretraining, we inject synthetic residue errors into target peptides and require the model to emit ⟨REFLECT⟩ followed by the correct residue. Errors are drawn either uniformly or from later residues of the same peptide, which mimics the residue-order confusions typical of real spectra. Gradients are blocked at error positions, so the injected error serves as context rather than as a learning target. On the 9-species benchmark, reflection-pretrained Transformers detect and correct their own mistakes at inference time and improve amino-acid and peptide-level precision over a standard Transformer and, on average, over domain-specific *de novo* methods, without any task-specific architectural modules. The same mechanism regularizes training and lets experts insert ⟨REFLECT⟩ at suspect residues. Code, data and pretrained model weights can be found in the repo.

1 INTRODUCTION

De novo peptide sequencing determines the amino-acid sequence of a peptide from its tandem mass spectrum without consulting a reference database (Aebersold & Mann, 2003; Ng et al., 2023). It is the only way to identify peptides that are absent from existing databases, which is essential for antibody characterization (Beslic et al., 2022), neoantigen discovery (Karunratanakul et al., 2019), and metaproteomics (Hettich et al., 2013). Deep learning now dominates the task (Tran et al., 2017; Yilmaz et al., 2022; 2023; Eloff et al., 2023; Xia et al., 2024; Yang et al., 2024; Jin et al., 2024; Zhang et al., 2025a). Most models cast it as conditional sequence generation: an encoder reads the observed peaks, and an autoregressive decoder emits the peptide one residue at a time.

These decoders produce only **answer tokens**, namely amino-acid residues. A residue is committed as soon as it is emitted. Even when later evidence in the spectrum contradicts an earlier choice, for example when the next fragment peak does not line up with the predicted prefix mass or the remaining precursor mass cannot be filled by any valid residues, the model has no way to say so, and it must continue decoding from a prefix it has reason to doubt. Because every residue fixes the prefix mass that all later residues must agree with, one early mistake usually invalidates the entire peptide. This is visible in the gap between residue-level and peptide-level accuracy: a standard Transformer reaches 0.717 amino-acid precision but only 0.443 peptide precision on the Mouse test set (Table 1). Protein sequence models more broadly share this answer-only output format (Rives et al., 2021; Rao et al., 2021; Meier et al., 2021; Brandes et al., 2022; Wu et al., 2021; Guntuboina et al., 2023; Xiao et al., 2025; Zhang et al., 2025b), but in *de novo* sequencing its cost is directly measurable, because every committed error can be checked against a gold label.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

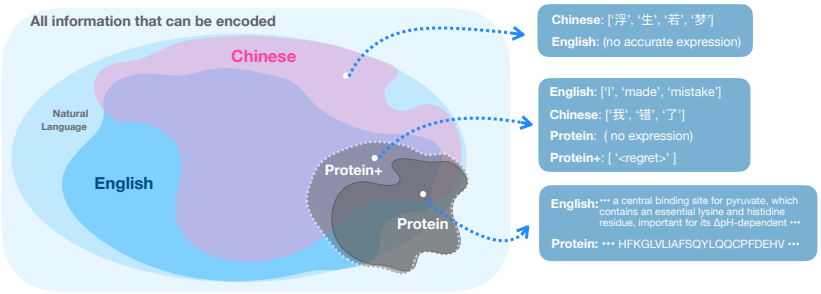


Figure 1: Expressiveness of languages. The peptide vocabulary cannot express intermediate states such as "the previous residue was wrong".

Natural language models overcome the same limitation with Chain-of-Thought (CoT) reasoning (Wei et al., 2022; Hu et al., 2025). Instead of mapping the input directly to answer tokens, $\boldsymbol{x_1 : x_n} \Rightarrow$ <answer$_1$> <answer$_2$> $\cdots$ <answer$_m$>, CoT interleaves **non-answer tokens** that are discarded from the final output:

$$x_1{:}x_n \Rightarrow \texttt{<answer_1>}\ [\texttt{non-answer_1}]\ [\texttt{non-answer_2}]\ \texttt{<answer_2>}\ \cdots\ [\texttt{non-answer_k}]\ \texttt{<answer_m>}.$$

These tokens let the model store intermediate results, perform additional computation, and *correct earlier errors*. Theoretical work shows that answer-only generation is strictly less expressive than generation with intermediate tokens (Li et al., 2024; Delétang et al., 2022) (see Appendix). The benefit, however, depends on the output language being able to express such intermediate states. English can express "I made a mistake"; the vocabulary of a peptide decoder, which contains only residues, cannot (Section 2).

In this work, we give *de novo* peptide sequencing models a non-answer token for self-correction. We add a reflection token $\langle\texttt{reflect}\rangle$ to the residue vocabulary and pretrain the model on peptides with injected errors, each followed by $\langle\texttt{reflect}\rangle$ and the correct residue. *De novo* sequencing is not only an important application but also an unusually clean setting in which to teach a model to reflect (Section 3.1). Every training spectrum comes with a **gold-label peptide**, so we know exactly where an injected error sits and what its correction must be, and no human annotation or reward model is required. Every residue is **verifiable against the input**, because consecutive fragment ions differ by exactly one residue mass and all residue masses must sum to the measured precursor mass, so a wrong residue leaves a trace in the spectrum the model is conditioned on. Finally, evaluation is **exact**, since predictions are scored against the gold peptide at the residue and peptide level.

Our experiments on the 9-species benchmark yield the following findings: **(1)** With error injection during pretraining, a standard encoder–decoder *de novo* model learns to **detect and revise its own residues** with $\langle\texttt{reflect}\rangle$, and also learns to keep residues that are already correct; **(2)** this ability **emerges only through pretraining**, since a model that is reflection-finetuned after standard pretraining never emits $\langle\texttt{reflect}\rangle$; **(3)** raising the error-injection ratio from 60% to 90% increases both reflection usage and peptide precision, with no further gain at 99%; **(4)** reflection pretraining improves peptide precision by up to 30% (relative) over the same Transformer trained without reflection, through **self-correction** and a **counter-overfitting** effect of dynamic error injection, and it outperforms domain-specific *de novo* methods on average; **(5)** the model supports **human-in-the-loop decoding**: a proteomics expert can insert $\langle\texttt{reflect}\rangle$ at a suspect residue and let the model re-decode from there.

2 Why Answer-Only Peptide Decoders Cannot Reflect

2.1 Computation Depends on the Output Tokens

The computational power of a sequence model, that is, the class of problems it can solve, is bounded by what it is allowed to output. Transformers that emit only answer tokens correspond to low complexity classes such as TC_0 (Li et al., 2024; Delétang et al., 2022). Their reasoning depth is fixed by the number of layers, their total computation is tied to the length of the answer, and they

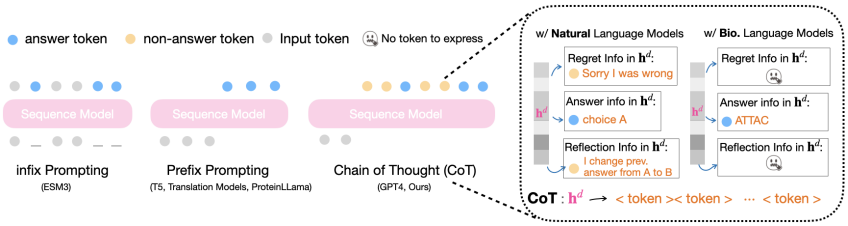


Figure 2: Prompting in natural language models versus *de novo* peptide sequencing models. A peptide decoder can only emit residues, so intermediate states such as self-correction have no token to be expressed with.

have no place to store intermediate results. CoT relaxes all three constraints: re-reading its own intermediate tokens simulates recurrence (**depth**), extra tokens buy extra computation before an answer is committed (**complexity**), and the tokens themselves act as an external **memory** tape. Under suitable assumptions, CoT-enhanced Transformers can even approach Turing completeness (Li et al., 2024), a level of expressiveness unattainable by direct-answer generation alone (Delétang et al., 2022). These analyses, however, implicitly assume that the output vocabulary is rich enough to carry such intermediate information, which holds for natural language but not for a peptide decoder.

2.2 Language Expressiveness

We make this assumption explicit.

Definition: Language Expressiveness

A language is defined as $\boldsymbol{L} = (\mathcal{G}, \mathcal{V})$, where $\mathcal{G}$ denotes the language's grammar specifying valid token sequences, and $\mathcal{V} = \{\mathtt{t}_1, \mathtt{t}_2, \ldots, \mathtt{t}_M\}$ is the vocabulary of usable tokens in this language. The set of all meanings expressible (valid sequences) in language $\boldsymbol{L}$ is denoted $\mathbb{S}_{\boldsymbol{L}}$. For any syntactically valid token sequence $\boldsymbol{s} = (\mathtt{t}_{i_1}, \mathtt{t}_{i_2}, \ldots)$, grounded in $\mathcal{G}$, it encodes certain information (meaning) $\boldsymbol{s} \in \mathbb{S}_{\boldsymbol{L}}$. The **expressiveness** of $\boldsymbol{L}$ is defined as the cardinality:

$$\text{Expressiveness}(\boldsymbol{L}) := |\mathbb{S}_{\boldsymbol{L}}|\,.$$

Although $|\mathbb{S}_{\boldsymbol{L}}|$ is usually infinite, languages can still be compared: if $\mathbb{S}_{\boldsymbol{L}_1} \subsetneq \mathbb{S}_{\boldsymbol{L}_2}$, then $\boldsymbol{L}_2$ is strictly **more expressive** than $\boldsymbol{L}_1$ (Fig. 1). The output language of a *de novo* sequencing model is $\boldsymbol{L}_{\text{peptide}} = (\mathcal{G}_{\text{peptide}}, \mathcal{V}_{\text{peptide}})$, where $\mathcal{V}_{\text{peptide}}$ contains the 20 canonical amino acids and a few modified residues (e.g., oxidized methionine and carbamidomethylated cysteine). Every sequence in this language denotes a candidate peptide and nothing else. Statements such as [``I'', ``am'', ``wrong''] in English, or "the residue I just emitted does not match the next fragment peak", have no expression in $\mathbb{S}_{\text{peptide}}$.

2.3 Language Expressiveness Determines CoT Expressiveness

Non-answer tokens are not random. They are discretized forms of information held in the decoder's hidden state $\mathbf{h}$, which stores everything the model has computed, including counters, intermediate results and uncertainty about earlier outputs. If we view the latent space as a language $\boldsymbol{\Theta} = (\mathcal{G}_\theta, \mathcal{V}_\theta)$ with meanings $\mathbb{S}_{\boldsymbol{\Theta}}$, each CoT step is a map from latent states to output tokens:

Essence of CoT

Each step of CoT can be viewed as a mapping from the latent language space to the output language: $\mathbb{S}_{\boldsymbol{\Theta}} \to \mathbb{S}_{\boldsymbol{L}}$, where the latent state $\mathbf{h} \in \mathcal{V}_\theta$ is decoded into a sequence of non-answer tokens $([\mathtt{na}_1], [\mathtt{na}_2], \ldots, [\mathtt{na}_k])$, with $[\mathtt{na}_k] \in \mathcal{V}_{\boldsymbol{L}}$ (Fig. 2, right panel).

In English, a latent "*regret*" state can be decoded as ['I', 'made', 'a', 'mistake']. A peptide decoder may well reach the same state: after emitting residue a_t, cross-attention to the spectrum can reveal that the prefix mass no longer matches the fragment-ion ladder. But the only tokens it can emit are residues, so the regret state has to be decoded into another amino acid or suppressed. **With**



Figure 3: *De novo* peptide sequencing from tandem mass spectrometry. Consecutive fragment ions differ by one residue mass, and all residue masses must sum to the precursor mass, so each predicted residue can be checked against the spectrum.

$\mathcal{V}_{\text{peptide}}$ **alone, the CoT expressiveness of a *de novo* decoder is effectively** 0. Section 3 adds the missing token.

3 Reflection Pretraining for *De Novo* Peptide Sequencing

3.1 Why *De Novo* Peptide Sequencing

Reflection training needs to know, at every position, whether a token is wrong and what it should have been. Open-ended generation such as protein design cannot provide this, because it has no single correct sequence. *De novo* peptide sequencing (Fig. 3) can:

1. **Gold labels at every position.** Each training spectrum is paired with a database-identified peptide, so errors can be injected anywhere with an exact correction target, at the scale of MassIVE-KB (30M spectra (Wang et al., 2018)), and predictions are scored exactly against the same labels.
2. **Every residue is verifiable from the input.** Consecutive fragment ions differ by one residue mass, and residue masses must sum to the precursor mass m. A wrong residue therefore contradicts the spectrum the decoder attends to, so learning when to reflect is well posed.
3. **Errors are local and structured.** Each residue is one reasoning step, and typical errors are similar-mass substitutions and order swaps ("AB" vs. "BA") caused by missing fragment ions, which motivates our error-injection strategies (Section 3.2.3).

Problem formulation. The model is given a spectrum $\mathbf{H} = \{\mathbf{I}, c, m\}$ produced by a mass spectrometer (Fig. 3), consisting of

- $\mathbf{I} = \{(\text{m/z}_1, i_1), (\text{m/z}_2, \text{i}_2), \ldots, (\text{m/z}_k, \text{i}_k)\}$, a set of k observed mass-to-charge and intensity pairs, filtered by a signal threshold;
- $c \in \mathbb{Z}^+$, the charge state of the precursor ion; and
- $m \in \mathbb{R}^+$, the total measured mass of the peptide.

The objective is to **predict the peptide** $\mathbf{A} = \{a_1, a_2, \ldots, a_n\}$ with $a_i \in \mathcal{V}_{\text{peptide}}$. The mapping $\mathbf{H} \mapsto \mathbf{A}$ requires the model to explain the peaks in $\mathbf{I}$ with a residue sequence whose total mass matches m.

3.2 Reflection-Token-Augmented Peptides for Next-Token Prediction

A decoder that has just emitted a residue inconsistent with the spectrum may encode this in its hidden state, $\mathbf{h}_{\text{regret}}$, but it cannot decode $\mathbf{h}_{\text{regret}}$ into any $a \in \mathcal{V}_{\text{peptide}}$. We therefore augment the vocabulary with a reflection token, $\mathcal{V}_{\text{peptide+}} = \mathcal{V}_{\text{peptide}} \oplus \langle\texttt{reflect}\rangle$. Using the gold labels, we construct a modified training set $\mathcal{D}_{\text{de novo+}}$ by injecting a wrong residue into the target peptide, followed by $\langle\texttt{reflect}\rangle$ and the correct residue. At inference, every $\langle\texttt{reflect}\rangle$ and the residue before it are removed to obtain the final peptide.

Below we describe the model, two error-injection strategies (Fig. 4), and our dynamic injection and gradient-blocking scheme.

3.2.1 MODEL ARCHITECTURE

We adopt a Transformer-based encoder-decoder architecture, following standard designs for this task (Yilmaz et al., 2023). The input spectrum is denoted as $\boldsymbol{H} = \{\mathbf{I}, c, m\}$, where $\mathbf{I} = \{(m/z_k, i_k)\}_{k=1}^{K}$ consists of K mass-to-charge and intensity pairs. Each $(m/z_k, i_k)$ is embedded via a sinusoidal encoding and projected into a d-dimensional vector. The resulting sequence of embeddings forms the encoder output $\boldsymbol{E} \in \mathbb{R}^{K \times d}$.

The decoder is an autoregressive Transformer with causal self-attention and cross-attention to $\boldsymbol{E}$. Let $\boldsymbol{h}_t^{(i)}$ denote the hidden state at decoding step t and decoder layer i. Causal self-attention ensures that each position only attends to past outputs: $\boldsymbol{h}_t^{(i)} = \text{Attn}\left(\boldsymbol{h}_t^{(i-1)}, \{\boldsymbol{h}_{1:t-1}^{(i-1)}\}\right)$. Each $\boldsymbol{h}_t^{(i)}$ then attends to the spectrum features via cross-attention over $\boldsymbol{E}$. The final decoder output at step t, denoted $\boldsymbol{h}_t^{(L)}$, is projected onto the vocabulary $\mathcal{V}_{\text{peptide+}}$: $P_t(\cdot \mid \boldsymbol{S}, \boldsymbol{y}_{<t}) = \text{softmax}\left(\boldsymbol{W}\boldsymbol{h}_t^{(L)}\right)$; Token $\boldsymbol{y}_t$ is sampled from $P_t(\cdot \mid \boldsymbol{S}, \boldsymbol{y}_{<t})$. Training proceeds via standard next-token prediction (NTP), minimizing the cross-entropy loss over ground-truth sequences.

3.2.2 RANDOM POSITION, RANDOM ERROR (RPRE) INJECTION

For a given input-spectrum and ground-truth sequence pair $(\boldsymbol{H}, \boldsymbol{A} = \{a_1, a_2, \ldots, a_n\}) \in \mathcal{D}_{\text{de novo}}$, we use the gold-label peptide to construct reflection trajectories that teach the model to revisit its previous residue and correct it.

In the RPRE strategy, we randomly sample a position $t \sim \text{Uniform}(1, n)$ and a random residue $\tilde{a}_t \sim \mathcal{V}_{\text{peptide}}$. We then replace the token at position t with $\tilde{a}_t$, regardless of correctness, and insert a reflection token $\langle\texttt{reflect}\rangle$ immediately afterward, followed by the original correct token a_t. The modified training target becomes:

$$\boldsymbol{A}' = \{a_1, \ldots, a_{t-1}, \tilde{a}_t, \langle\texttt{reflect}\rangle, a_t, a_{t+1}, \ldots, a_n\}$$

Because the gold label fixes a_t, the correction target is always exact. Notably, when $\tilde{a}_t = a_t$ (i.e., the random residue happens to be correct), the reflection becomes a no-op, which teaches the model to keep a residue that is consistent with the spectrum. As a result, RPRE encourages both correction and retention behavior through the same mechanism.

3.2.3 RANDOM POSITION, LOOKAHEAD ERROR (RPLE) INSERTION

A uniformly sampled residue usually has a mass that clearly contradicts the local fragment ladder, which makes it easy to detect. Real *de novo* errors are subtler: when the fragment ion between two adjacent residues is missing, the decoder often emits a residue that belongs later in the peptide, producing an order swap with the same total mass. To mimic these errors, we propose ***Random Position, Lookahead Error (RPLE)*** insertion, which samples the replacement residue from the gold peptide itself. Given a spectrum-label pair $(\boldsymbol{H}, \boldsymbol{A} = \{a_1, a_2, \ldots, a_n\}) \in \mathcal{D}_{\text{de novo}}$, we randomly select a position $t \sim \text{Uniform}(1, n)$ and choose an error token $\tilde{a}_t$ from a set of true tokens that appear later in the sequence: $\tilde{a}_t \sim \{a_{t+1}, a_{t+2}, \ldots, a_n\}$. We then replace a_t with $\tilde{a}_t$, append the reflection token $\langle\texttt{reflect}\rangle$, and restore the correct token a_t. The injected residue is present in the peptide and consistent with the precursor mass, so it can only be recognized from position-specific fragment evidence. This teaches the model to correct premature residues and order swaps (e.g., predicting "BA" instead of "AB").

3.2.4 BATCH-LEVEL ONLINE DYNAMIC ERROR INJECTION

To prevent the model from memorizing a fixed error for each spectrum and to encourage generalizable reflection behavior, we adopt *batch-level online dynamic error injection* during training. For each mini-batch randomly sampled from $\mathcal{D}_{\text{de novo}}$, we apply error injection in real-time to the sequences.

Given an injection ratio $\alpha \in [0, 1]$, a fraction α of sequences in each batch are modified using either the RPRE or RPLE strategy. The remaining $(1 - \alpha)$ fraction are left unchanged:

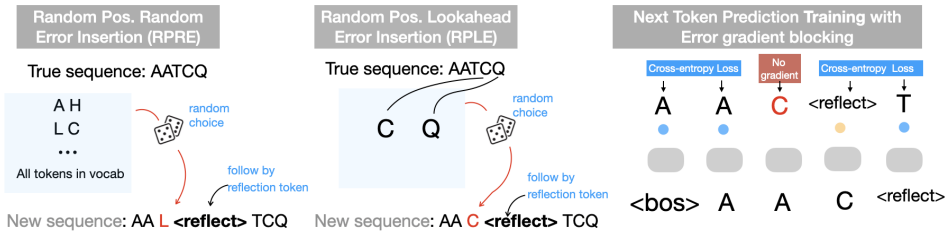


Figure 4: Error injection and reflection training for *de novo* peptide sequencing.

Algorithm 1 Online Dynamic Reflection-Error Injection During Training

Require: Batch $\mathcal{B} = \{(\boldsymbol{H}^{(i)}, \boldsymbol{A}^{(i)})\}_{i=1}^{B}$, injection ratio $\alpha \in [0, 1]$
1: **for** each $(\boldsymbol{H}, \boldsymbol{A})$ in $\mathcal{B}$ **do**
2: **if** Uniform$(0, 1) < \alpha$ **then**
3: Sample $t \sim$ Uniform$(1, |\boldsymbol{A}|)$, choose RPRE or RPLE
4: $\tilde{a}_t \sim$ **RPRE**: $\mathcal{V}_{\text{peptide}}$ **RPLE**: $\{a_{t+1}, \ldots, a_n\}$
5: $\boldsymbol{A}' \leftarrow \{a_1, \ldots, a_{t-1}, \tilde{a}_t, \langle\texttt{reflect}\rangle, a_t, \ldots, a_n\}$
6: Replace $\boldsymbol{A}$ with $\boldsymbol{A}'$ in $\mathcal{B}$
7: **end if**
8: **end for**

3.2.5 Error Position Gradient Blocking

To ensure that injected errors act as *contextual signals* rather than learning targets, we apply **gradient blocking** at error positions during training. Formally, given a modified target sequence $\boldsymbol{A}' = \{a_1, \ldots, \tilde{a}_t, \langle\texttt{reflect}\rangle, a_t, \ldots\}$, we exclude the loss term at position t–corresponding to the injected error token $\tilde{a}_t$–from the training objective. This allows the model to condition on the error via causal attention and learn to generate the reflection token $\langle\texttt{reflect}\rangle$ in response, without learning to generate the incorrect residue itself (Fig. 4 right panel).

4 Experiments

4.1 Experimental Setup

Datasets. Following established protocols (Yilmaz et al., 2023), we use the MassIVE-KB dataset (Wang et al., 2018), which contains **30 million peptide-spectrum matches (PSMs)** collected from diverse mass spectrometry platforms. Evaluation is conducted on the widely adopted 9-species-v1 and 9-species-v2 benchmarks.

Implementation. Input peaks and amino acids are embedded into 512-dimensional vectors. Both the spectrum encoder and peptide decoder are 9-layer Transformers with 8 attention heads and 1024-dimensional hidden states. Models are trained on eight NVIDIA A100 80GB GPUs using AdamW ($\alpha_0 = 5 \times 10^{-4}$) with a 10k-step linear warmup followed by cosine decay.

Evaluation Metrics. We report two metrics. **Amino acid-level (AA) Precision** considers a prediction correct if the amino acid token aligns with the target position. Accuracy is defined as $\boldsymbol{M}_{\text{AA}}/\boldsymbol{T}_{\text{AA}}$, where $\boldsymbol{M}_{\text{AA}}$ is the number of matched amino acids and $\boldsymbol{T}_{\text{AA}}$ is the total number of predicted amino acids. **Peptide-level precision** considers a prediction correct only if the entire predicted peptide matches the ground truth sequence exactly. Precision is defined as $\boldsymbol{M}_{\text{pep}}/\boldsymbol{T}_{\text{pep}}$, where $\boldsymbol{M}_{\text{pep}}$ is the number of matched peptides and $\boldsymbol{T}_{\text{pep}}$ is the total number of predicted peptides.

4.2 Results

Results Analysis. As shown in Table 1, reflection-based pretraining leads to substantial improvements in both amino acid and peptide precision across all 9 species. In contrast, **finetuning** shows negligible gains over baselines, and reflection tokens are never used during inference (Table 3), indicating that reflection behavior is not learned through finetuning alone.

Table 1: Comparison of different settings of reflection pretraining, a placeholder-token ablation, and the baseline on the 9-species test set.

Method (beam size = 1)	*Mouse*	*Human*	*Yeast*	*M.mazei*	*Honeybee*	*Tomato*	*R.bean*	*Bacillus*	*C.bacteria*
Standard Pretrain				**Amino Acid Precision**					
Transformer (Δ baseline)	0.717	0.649	0.752	0.713	0.706	0.763	0.714	0.753	0.66
Placeholder Token Pretrain									
Transformer + 60% Placeholder	0.721	0.644	0.740	0.716	0.699	0.763	0.711	0.740	0.658
Reflect. Finetune w/ *RPRE*									
Transformer + 60% Error	0.737	0.672	0.736	0.739	0.690	0.784	0.749	0.774	0.684
Reflect. Pretrain w/ *RPRE*									
Transformer + 60% Error	0.765	0.713	0.796	0.772	0.721	0.806	0.795	0.811	0.713
Reflect. Pretrain w/ *RP(RE +LE)*									
Transformer + 60% Error	0.784	0.735	0.808	0.786	0.743	0.813	0.808	0.820	0.723
+ 90% Error	**0.792** (+10.5%)	**0.752** (+15.9%)	**0.809** (+7.6%)	**0.790** (+10.8%)	**0.744** (+5.4%)	**0.822** (+7.7%)	**0.817** (+14.4%)	**0.826** (+9.7%)	**0.737** (+11.2%)
+ 99% Error	0.786	0.739	0.807	0.789	0.742	0.819	0.814	0.823	0.732
Standard Pretrain				**Peptide Precision**					
Transformer (Δ baseline)	0.443	0.433	0.584	0.522	0.460	0.606	0.652	0.580	0.413
Placeholder Token Pretrain									
Transformer + 60% Placeholder	0.447	0.431	0.578	0.523	0.458	0.601	0.646	0.589	0.412
Reflect. Finetune w/ *RPRE*									
Transformer + 60% Error	0.443	0.434	0.589	0.530	0.465	0.607	0.563	0.583	0.416
Reflect. Pretrain w/ *RPRE*									
Ours (60% ER, 6 AA/PEP)	0.485	0.494	0.632	0.575	0.507	0.644	0.629	0.635	0.458
Reflect. Pretrain w/ *RP(RE +LE)*									
Transformer + 60% Error	0.513	0.535	0.651	0.592	0.534	0.658	0.665	0.662	0.470
+ 90% Error	**0.533** (+20.3%)	**0.563** (+30.0%)	**0.661** (+13.2%)	**0.605** (+15.9%)	**0.544** (+18.3%)	**0.668** (+10.2%)	**0.657** (+0.8%)	**0.674** (+16.2%)	**0.490** (+18.6%)
+ 99% Error	0.515	0.537	0.648	0.595	0.534	0.660	0.657	0.660	0.474

Case Study: Self-Reflection and Error-Correction During Inference Time

Raw prediction: `R<reflect>KYFHNELM+15.995NYVQEC+57.021QFDSETSL$`
Post-processed output: `KYFHNELM+15.995NYVQEC+57.021QFDSETSL`
Ground-truth label: `KYFHNELM+15.995NYVQEC+57.021QFDSETSL`

Notably, increasing the error ratio from 60% to 90% further boosts peptide precision in all species, while also increasing the frequency of reflection token usage from 2.3% to 4.46% during inference (Table 3). Despite most `<reflect>` tokens retaining the original answer (e.g., 67.8% in the 90% error model), this indicates a learned capacity for self-assessment. Moreover, incorporating RPLE during pretraining introduces more realistic, confusing errors, pushing peptide precision further up.

Larger Gains at the Peptide Level. The relative improvement from reflection pretraining is typically larger for peptide precision than for amino-acid precision (Table 1). On Human, for example, amino-acid precision improves by 15.9% and peptide precision by 30.0%; on Honeybee, by 5.4% and 18.3%. This matches the failure mode that motivates our approach. A peptide is counted as correct only if every residue is correct, and in answer-only decoding a single early error propagates through the rest of the sequence. Correcting one residue therefore often turns an entirely wrong peptide into a correct one, which affects peptide precision far more than residue-level precision.

Placeholder-Token Ablation. Reflection pretraining lengthens the target sequence with extra non-residue tokens, so part of its gain could come from the added tokens alone rather than from reflection. To rule this out, we pretrain the same Transformer with a placeholder token inserted at random positions of the target peptide. The placeholder carries no reflection or correction semantics and is removed from the output at inference. As shown in Table 1, simply adding placeholder tokens during training does not improve amino-acid or peptide precision, and even degrades performance on some species. The gains of reflection pretraining therefore come from the functional role of ⟨reflect⟩, not from additional non-answer tokens.

Outperforming Domain-Specific *De Novo* Methods Without Domain Modules. As shown in Table 2, our reflection-pretrained model achieves the best average amino-acid precision and peptide recall among the compared *de novo* methods, despite using **no task-specific architectural modules**. Baselines such as HelixNovo encode fragmentation rules (complementary b/y ions) into the input,

Table 2: Comparison with *de novo* peptide sequencing methods on the 9-species benchmark.

Method	*Mouse*	*Human*	*Yeast*	*M.mazei*	*Honeybee*	*Tomato*	*R.bean*	*Bacillus*	*C.bacteria*	**Average**
					AA Precision					
***De Novo* Methods**										
Peaks	0.600	0.639	0.748	0.673	0.633	0.728	0.644	0.719	0.586	0.663
DeepNovo	0.623	0.610	0.750	0.694	0.630	0.731	0.679	0.742	0.602	0.673
PointNovo	0.626	0.606	0.779	0.712	0.644	0.733	0.730	0.768	0.589	0.688
InstaNovo	0.703	0.636	0.691	0.712	0.660	0.732	0.711	0.739	0.619	0.689
Casanovo, Beam = 1	0.717	0.649	0.752	0.713	0.706	0.763	0.714	0.753	0.663	0.704
HelixNovo, Beam = 1	0.750	0.648	0.758	0.766	0.699	0.757	0.771	0.799	0.661	0.734
Reflection Pretrain										
90% Error, Beam = 1	**0.792**	**0.752**	**0.809**	**0.790**	**0.744**	**0.822**	**0.817**	**0.826**	**0.737**	**0.788**
90% Error, Beam = 3	**0.799**	**0.763**	**0.818**	**0.800**	**0.755**	**0.827**	**0.832**	**0.836**	**0.745**	**0.797**
90% Error, Beam = 5	**0.805**	**0.774**	**0.826**	**0.810**	**0.766**	**0.831**	**0.846**	**0.846**	**0.753**	**0.806**
					Peptide Recall					
***De Novo* Methods**										
Peaks	0.197	0.277	0.428	0.356	0.287	0.403	0.362	0.387	0.203	0.333
DeepNovo	0.286	0.293	0.462	0.422	0.330	0.454	0.436	0.449	0.253	0.387
PointNovo	0.355	0.351	0.534	0.478	0.396	0.513	0.511	0.518	0.298	0.439
InstaNovo	0.471	0.455	0.559	0.528	0.466	0.732	0.564	0.576	0.416	0.530
Casanovo	0.443	0.433	0.584	0.522	0.460	0.606	0.652	0.580	0.413	0.521
Reflection Pretrain										
90% Error, Beam=1	**0.533**	**0.563**	**0.661**	**0.605**	**0.544**	**0.668**	**0.657**	**0.674**	**0.490**	**0.600**
90% Error, Beam=3	**0.540**	**0.573**	**0.669**	**0.615**	**0.555**	**0.674**	**0.667**	**0.685**	**0.497**	**0.608**
90% Error, Beam=5	**0.546**	**0.582**	**0.676**	**0.624**	**0.565**	**0.680**	**0.676**	**0.696**	**0.504**	**0.617**

whereas our model uses the same standard Transformer as Casanovo and differs only in reflection pretraining. This suggests that the ability to correct an early residue matters as much as additional spectral priors.

Case Study: Self-Reflection but no-op During Inference time

Raw prediction: KDFFTYME<reflect>E$
Post-processed output: KDFFTYME
Ground-truth label: KDFFTYME

Table 3: Reflection usage during inference. **Absolute Use**: % sequences with <reflect> among all sequences; **Corr.**: % reflections that corrected errors; **Same**: % reflections that retained the original prediction. Others % include that reflection change the previous error to another error as well as from correct answer to an error (rarely). **Actual Use**: % sequences with <reflect> among all incorrect sequences, notice that originally correctly predicted sequences (roughly 60%) don't even need to rely on reflection tokens.

Model	Use	Corr.	Same	Others	Actual Use
Baseline	0.0%	–	–	–	–
Reflect FT	0.0%	0.0%	0.0%	–	0.0%
PT 60% Error	2.3%	13.6%	63.2%	23.2%	4.1%
PT 90% Error	4.5%	12.7%	67.8%	19.5%	7.2%

Reflection Behavior Case Study. As shown in Table of Case studies, the model exhibits **strong reflection** capabilities during inference. In the first example, the model emits an incorrect first residue (R), reflects, and then decodes the full peptide correctly, including both modified residues. In the second, it reflects on the final residue but keeps it. This balance between correction and retention shows that the model uses ⟨reflect⟩ selectively rather than as a default action.

Impact of Error Injection on Generalization. Figure 5 shows the effect of different reflection error ratios on validation loss during pretraining. Without error injection (blue curve), the model exhibits clear signs of overfitting: validation loss initially decreases but begins to rise after 100 steps. In contrast, injecting reflection errors–especially at 90% and 99% rates–leads to more stable and monotonic decreases in validation loss. This improvement stems from the fact that our **batch-dynamic error** alters sequences throughout the training, preventing the model from memorizing fixed input-output mappings. As a result, the model is forced to generalize across perturbed inputs, reducing overfitting.

Human-in-the-Loop Reflection. We manually picked 10 sequences where the model failed to detect errors. By manually inserting <reflect> tokens at the error positions and then generating from there, we found that 2 out of 10 sequences were successfully corrected during inference. Locating errors without a gold label is still an open problem, but in practice proteomics experts can often flag

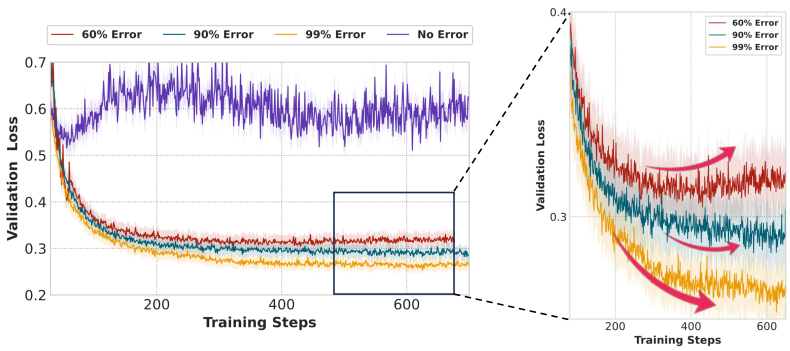


Figure 5: Error Injection Rate on Validation Loss. Loss Plotting and Logging by Neptune.ai (Before it was taken down).

suspect residues, for example where fragment-ion coverage is missing, and let the model re-decode from there.

Case Study: Manual Reflection Improves Prediction

Raw prediction:	RLANLYWL$
Manual intervention:	RL<reflect>
Output after intervention:	RL<reflect>MNFYGFL
Ground-truth label:	RMNFYGFL

The case study above illustrates such an intervention. The model first predicts "RLANLYWL". After a <reflect> token is inserted following the second residue, it produces "MNFYGFL", and the final peptide matches the ground truth "RMNFYGFL". The correction cascades beyond a single token: the model revises not only the second residue but the entire downstream sequence. Because each residue fixes the prefix mass that all later residues must agree with, repairing one residue lets the model re-read the fragment ladder consistently from that point on.

Additional results, including evaluation on extended benchmark datasets, the effect of beam size, and further case studies with analysis, are provided in Appendix.

Analysis of Beam Size Influence. We examined the effect of beam search width on the performance of the Reflection Pretrain model. As shown in Table 2, increasing the beam size consistently improves model performance across both Amino Acid (AA) Precision and Peptide Recall metrics. Specifically, AA Precision increased from 0.788 at beam size 1, to 0.797 at beam size 3, and further to 0.806 at beam size 5. This reflects an absolute improvement of 0.018, corresponding to a relative increase of approximately 2.28%. A similar trend was observed for Peptide Recall, which rose from 0.600 (beam size 1) to 0.608 (beam size 3), and reached 0.617 at beam size 5–an absolute gain of 0.017 and a relative improvement of approximately 2.83%. These results suggest that larger beam sizes enable the model to explore a broader hypothesis space during decoding, thereby reducing the risk of prematurely discarding promising peptide candidates. Nevertheless, the performance benefits must be balanced against the increased computational cost in both time and memory–an inherent trade-off in beam search.

5 Conclusion

We introduced reflection pretraining, which gives *de novo* peptide sequencing models a way to revise residues they have already emitted. The approach relies on two properties of the task: gold-label peptides make the position and correction of every injected error exact, and the fragment-ion ladder and precursor mass make each residue verifiable from the spectrum the model conditions on. A standard Transformer trained this way learns to reflect selectively, improves amino-acid and peptide precision on all nine species, overfits less, and lets experts trigger corrections at suspect residues. Other generation tasks with gold labels and input-verifiable outputs are natural next candidates for the same recipe.

AI USE STATEMENT

We used generative AI tools to edit the wording of the manuscript. The work is original and not generated by AI. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning. We do not identify any ethical concerns that require specific discussion.

REPRODUCIBILITY STATEMENT

Code, data and pretrained model weights are available at the anonymous repository linked in the abstract. The datasets (MassIVE-KB and the 9-species benchmark), model architecture and optimization settings are described in Section 4.1, and further training details and baseline descriptions are given in Appendix F. The error-injection procedure is specified in Algorithm 1.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Ruedi Aebersold and Matthias Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928): 198–207, 2003.

Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33 (8):831–838, 2015.

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Denis Beslic, Georg Tscheuschner, Bernhard Y Renard, Michael G Weller, and Thilo Muth. Comprehensive evaluation of peptide de novo sequencing tools for monoclonal antibody assembly. *Briefings in Bioinformatics*, 5:1–12, dec 2022. ISSN 1467-5463.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. Proteinbert: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110, 2022.

Jürgen Cox and Matthias Mann. MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification. *Nature Biotechnology*, 26 (12):1367–1372, 2008. ISSN 10870156.

Vlado Dančík, Theresa A Addona, Karl R Clauser, James E Vath, and Pavel A Pevzner. De novo peptide sequencing via tandem mass spectrometry. *Journal of computational biology*, 6(3-4): 327–342, 1999.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, et al. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*, 2022.

Zhi-Luo Deng, Philipp C Münch, René Mreches, and Alice C McHardy. Rapid and accurate identification of ribosomal rna sequences via deep learning. *Nucleic acids research*, 50(10): e60–e60, 2022.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

Kevin Eloff, Konstantinos Kalogeropoulos, Oliver Morell, Amandla Mabona, Jakob Berg Jespersen, Wesley Williams, Sam PB van Beljouw, Marcin Skwark, Andreas Hougaard Laustsen, Stan JJ Brouns, et al. De novo peptide sequencing with instanovo: Accurate, database-free peptide identification for large scale proteomics experiments. *bioRxiv*, pp. 2023–08, 2023.

J K Eng, A L McCormack, and J R Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*, 5(11):976–989, nov 1994. ISSN 1044-0305 (Print).

Ari Frank and Pavel Pevzner. Pepnovo: de novo peptide sequencing via probabilistic network modeling. *Analytical chemistry*, 77(4):964–973, 2005.

Erik Garrison. Memory makes computation universal, remember? *arXiv preprint arXiv:2412.17794*, 2024.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Chakradhar Guntuboina, Adrita Das, Parisa Mollaei, Seongwon Kim, and Amir Barati Farimani. Peptidebert: A language model based on transformers for peptide property prediction. *The Journal of Physical Chemistry Letters*, 14(46):10427–10434, 2023.

Robert L Hettich, Chongle Pan, Karuna Chourey, and Richard J Giannone. Metaproteomics: harnessing the power of high performance mass spectrometry to identify the suite of proteins that control metabolic activities in microbial communities. *Analytical chemistry*, 85(9):4203–4214, may 2013. ISSN 1520-6882 (Electronic).

Ming Hu, Chenglong Ma, Wei Li, Wanghan Xu, Jiamin Wu, Jucheng Hu, Tianbin Li, Guohang Zhuang, Jiaqi Liu, Yingzhou Lu, et al. A survey of scientific large language models: From data foundations to agent frontiers. *arXiv preprint arXiv:2508.21148*, 2025.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023.

Zhi Jin, Sheng Xu, Xiang Zhang, Tianze Ling, Nanqing Dong, Wanli Ouyang, Zhiqiang Gao, Cheng Chang, and Siqi Sun. Contranovo: A contrastive learning approach to enhance de novo peptide sequencing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 144–152, 2024.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Korrawe Karunratanakul, Hsin-Yao Tang, David W Speicher, Ekapol Chuangsuwanich, and Sira Sriswasdi. Uncovering Thousands of New Peptides with Sequence-Mask-Search Hybrid De Novo Peptide Sequencing Framework. *Molecular & cellular proteomics : MCP*, 18(12):2478–2491, dec 2019. ISSN 1535-9484 (Electronic).

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 1, 2024.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

Liuzhenghao Lv, Zongying Lin, Hao Li, Yuyang Liu, Jiaxi Cui, Calvin Yu-Chian Chen, Li Yuan, and Yonghong Tian. Prollama: A protein large language model for multi-task protein language processing. *IEEE Transactions on Artificial Intelligence*, 2025.

Bin Ma, Kaizhong Zhang, Christopher Hendrie, Chengzhi Liang, Ming Li, Amanda Doherty-Kirby, and Gilles Lajoie. Peaks: powerful software for peptide de novo sequencing by tandem mass spectrometry. *Rapid communications in mass spectrometry*, 17(20):2337–2342, 2003.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in neural information processing systems*, 34:29287–29303, 2021.

William Merrill and Ashish Sabharwal. The expresssive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

Cheuk Chi A Ng, Yin Zhou, and Zhong-Ping Yao. Algorithms for de-novo sequencing of peptides by tandem mass spectrometry: A review. *Analytica Chimica Acta*, pp. 341330, 2023.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021.

D N Perkins, D J Pappin, D M Creasy, and J S Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20, 1999. URL `http://dx.doi.org/3.0.CO;2-2`.

Rui Qiao, Ngoc Hieu Tran, Lei Xin, Xin Chen, Ming Li, Baozhen Shan, and Ali Ghodsi. Computationally instrument-resolution-independent de novo peptide sequencing for high-resolution devices. *Nature Machine Intelligence*, 3(5):420–425, 2021.

Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*, pp. 8844–8856. PMLR, 2021.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *arXiv preprint arXiv:2308.10379*, 2023.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024.

Mirac Suzgun and Adam Tauman Kalai. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *arXiv preprint arXiv:2401.12954*, 2024.

Ngoc Hieu Tran, Xianglilan Zhang, Lei Xin, Baozhen Shan, and Ming Li. De novo peptide sequencing by deep learning. *Proceedings of the National Academy of Sciences*, 114(31):8247–8252, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Mingxun Wang, Jian Wang, Jeremy Carver, Benjamin S Pullman, Seong Won Cha, and Nuno Bandeira. Assembling the community-scale discoverable human proteome. *Cell systems*, 7(4): 412–421, 2018.

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. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Zachary Wu, Kadina E Johnston, Frances H Arnold, and Kevin K Yang. Protein sequence design with deep generative models. *Current opinion in chemical biology*, 65:18–27, 2021.

Jun Xia, Shaorong Chen, Jingbo Zhou, Tianze Ling, Wenjie Du, Sizhe Liu, and Stan Z Li. Adanovo: Adaptive\emph {De Novo} peptide sequencing with conditional mutual information. *arXiv preprint arXiv:2403.07013*, 2024.

Yijia Xiao, Wanjia Zhao, Junkai Zhang, Yiqiao Jin, Han Zhang, Zhicheng Ren, Renliang Sun, Haixin Wang, Guancheng Wan, Pan Lu, et al. Protein large language models: A comprehensive survey. *arXiv preprint arXiv:2502.17504*, 2025.

Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):11407–11427, 2023.

Fan Yang, Wenchuan Wang, Fang Wang, Yuan Fang, Duyu Tang, Junzhou Huang, Hui Lu, and Jianhua Yao. scbert as a large-scale pretrained deep language model for cell type annotation of single-cell rna-seq data. *Nature Machine Intelligence*, 4(10):852–866, 2022.

Tingpeng Yang, Tianze Ling, Boyan Sun, Zhendong Liang, Fan Xu, Xiansong Huang, Linhai Xie, Yonghong He, Leyuan Li, Fuchu He, et al. Introducing π-helixnovo for practical large-scale de novo peptide sequencing. *Briefings in Bioinformatics*, 25(2):bbae021, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Melih Yilmaz, William Fondrie, Wout Bittremieux, Sewoong Oh, and William S Noble. De novo mass spectrometry peptide sequencing with a transformer model. In *International Conference on Machine Learning*, pp. 25514–25522. PMLR, 2022.

Melih Yilmaz, William E Fondrie, Wout Bittremieux, Carlo F Melendez, Rowan Nelson, Varun Ananth, Sewoong Oh, and William Stafford Noble. Sequence-to-sequence translation from mass spectra to peptides with a transformer model. *BioRxiv*, pp. 2023–01, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Jing Zhang, Lei Xin, Baozhen Shan, Weiwu Chen, Mingjie Xie, Denis Yuen, Weiming Zhang, Zefeng Zhang, Gilles A Lajoie, and Bin Ma. PEAKS DB: de novo sequencing assisted database search for sensitive and accurate peptide identification. *Molecular & cellular proteomics : MCP*, 11(4): M111.010587, apr 2012. ISSN 1535-9484 (Electronic).

Jinny X Zhang, Boyan Yordanov, Alexander Gaunt, Michael X Wang, Peng Dai, Yuan-Jyue Chen, Kerou Zhang, John Z Fang, Neil Dalchau, Jiaming Li, et al. A deep learning model for predicting next-generation sequencing depth from dna sequence. *Nature communications*, 12(1):4387, 2021.

Xiang Zhang, Tianze Ling, Zhi Jin, Sheng Xu, Zhiqiang Gao, Boyan Sun, Zijie Qiu, Jiaqi Wei, Nanqing Dong, Guangshuai Wang, et al. π-primenovo: an accurate and efficient non-autoregressive deep learning model for de novo peptide sequencing. *Nature Communications*, 16(1):267, 2025a.

Xiang Zhang, Jiaqi Wei, Zijie Qiu, Sheng Xu, Zhi Jin, ZhiQiang Gao, Nanqing Dong, and Siqi Sun. Bidirectional representations augmented autoregressive biological sequence generation: Application in de novo peptide sequencing. *arXiv preprint arXiv:2510.08169*, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

A Related Work

Chain-of-Thought Reasoning in Natural Language Models. Traditional natural language processing (NLP) systems predominantly followed an *answer-only* generation paradigm, directly mapping inputs to final outputs. This changed with the introduction of Chain-of-Thought (CoT) prompting (Wei et al., 2022), which enables models to emit intermediate reasoning steps-non-answer tokens-prior to producing a final answer. CoT significantly improves performance on tasks requiring compositional reasoning (Nye et al., 2021; Kojima et al., 2022), by allowing models to retain transient memory, perform iterative updates, and correct earlier errors during generation. The success of CoT has led to a surge in structured reasoning approaches, including tree-structured (Yao et al., 2024; Long, 2023), graph-based (Besta et al., 2024; Sel et al., 2023), and decompositional methods (Zhou et al., 2022; Drozdov et al., 2022; Khot et al., 2022). Further extensions (Zelikman et al., 2022; Madaan et al., 2024; Suzgun & Kalai, 2024) demonstrate that the effectiveness of these methods hinges on the high expressiveness of human language-capable of articulating logic, uncertainty, abstraction, and reflection of which enrich the non-answer space.

Answer-Only Decoding in *De Novo* Peptide Sequencing. Deep learning models have advanced sequence modeling in biology across DNA (Alipanahi et al., 2015; Zhang et al., 2021), RNA (Yang et al., 2022; Deng et al., 2022), and proteins (Rives et al., 2021; Jumper et al., 2021; Elnaggar et al., 2021; Lin et al., 2023). Yet, despite these advances, generation in biological domains-particularly in *de novo* peptide sequencing (Yilmaz et al., 2022; Eloff et al., 2023; Xia et al., 2024)-remains constrained to answer-only prediction. These models operate over token spaces (e.g., 20 amino acids) with inherently limited semantic richness, producing final sequence tokens without externalized reasoning traces. We argue that this limitation is not merely architectural but stems from the low expressive capacity of biological sequence languages. Unlike natural language, biological alphabets are poorly suited for encoding intermediate states, error attribution, or hypothetical exploration capabilities central to CoT-style reasoning.

Expressiveness, Computational Power, and Generative Capacity. While Transformers with CoT prompting can theoretically simulate Turing-complete machines (Pérez et al., 2021; Merrill & Sabharwal, 2023; Strobl et al., 2024), their practical reasoning capacity is constrained by input length, memory bandwidth (Arora & Barak, 2009; Garrison, 2024), and crucially, the expressiveness of the output token space. In biological models, the inability to produce diverse non-answer tokens impedes the model's ability to represent internal state transitions or conduct hypothesis-driven exploration. This constitutes a fundamental bottleneck, distinct from issues of model size or architecture. In contrast, natural language models such as GPT (Achiam et al., 2023) benefit from an output space rich in semantics, enabling a flexible reasoning substrate. Biological models like ESM (Rives et al., 2021) or ProLLaMA (Lv et al., 2025) lack this capacity, as their generation is grounded in symbolic alphabets that are not optimized for reasoning abstraction.

Embedding Reflection and Self-Correction into Biological Models. Recent advances in NLP promote self-reflection and critique during inference (Madaan et al., 2024), but these methods are predominantly post hoc and not integrated into model training. Our work is the first to incorporate reflection and error correction directly into the pretraining phase of biological sequence models. Specifically, we propose a reflection-based pretraining strategy for *de novo* peptide sequencing, where models are exposed to corrupted prediction traces and explicitly trained to identify and amend errors. This goes beyond inference-time prompting, embedding reasoning patterns into model weights. Complementary efforts in scaling Transformer context (Beltagy et al., 2020; Zhang et al., 2023; Jiang et al., 2023) address sequence length limitations but do not address the deeper issue of representational constraints. By introducing a structured set of non-answer tokens and designing tasks that invoke reasoning within biological domains, our framework redefines the generative capacity of biological models to support more intelligent, interpretable, and compositional generation.

B Model Expressive Power: From MLPs to Recurrent Architectures and Transformers

The **expressive power** of a neural network architecture dictates its fundamental capacity to represent or approximate complex functions, thereby determining its suitability for a diverse range of computational tasks. This capacity is intrinsically linked to the architectural design, especially its

mechanisms for information propagation and transformation across processing stages. Architectures with constrained expressive power may falter on tasks demanding deep sequential reasoning or intricate dependency modeling, whereas more expressive models can capture a richer set of complex patterns and computational processes.

B.1 Defining Effective Computational Depth

We propose to formalize a critical dimension of expressive power via the concept of *Effective Computational Depth*, denoted $D_{\text{eff}}(\mathcal{M})$. This metric quantifies the maximal length L of a sequence of non-trivial transformations $(\mathcal{T}_i)_{i=1}^{L}$ that a model $\mathcal{M}$ (with parameters θ) can compose to map an input x to an output $y^*(x)$, potentially averaged over a distribution of tasks $\mathcal{P}_{\text{task}}$. Each $\mathcal{T}_i$ represents a distinct computational step, possibly conditioned on intermediate results generated by preceding transformations $\mathcal{T}_{j<i}$.

$$D_{\text{eff}}(\mathcal{M}) := \sup\Big\{ L \in \mathbb{N} \;\Big|\; \exists\, \mathbf{T}^{(L)} \subseteq \mathcal{M} \text{ s.t. } \forall\, P \in \mathcal{P}_{\text{task}}^{(L)},\ \mathcal{R}_{\min}\big(P, \mathbf{T}^{(L)}\big) \leq \epsilon \Big\}. \tag{1}$$

Here, $\mathcal{P}_{\text{task}}^{(L)}$ signifies tasks inherently requiring L sequential computational steps. $\mathcal{R}_{\min}$ represents the minimum achievable risk (or loss) for task P using the sequence of L transformations $\mathbf{T}^{(L)}$ afforded by model $\mathcal{M}$, and ϵ is a small tolerance indicating successful task resolution. A higher $D_{\text{eff}}(\mathcal{M})$ signifies a greater capacity for deep sequential processing, a cornerstone of tackling complex computations. The depth complexity, as discussed in prior work, measures the number of sequential steps after considering all parallel processing a model performs. $D_{\text{eff}}(\mathcal{M})$ aims to capture this notion of maximal sequential transformative capacity.

B.2 Multi-Layer Perceptrons (MLPs)

Multi-Layer Perceptrons (MLPs) are structured with a fixed number of layers, say m. Computation proceeds sequentially through these layers: $h^{(i)} = \sigma(W^{(i)}h^{(i-1)})$. Each layer's operation constitutes a transformation $\mathcal{T}_i$. Since m is a constant, independent of the input sequence length n, the effective computational depth $D_{\text{eff}}(\text{MLP})$ is $O(m)$. This effectively becomes $O(1)$ with respect to n, as m does not scale with input size. This inherently fixed and often shallow depth restricts MLPs from adeptly solving tasks that necessitate iterative refinement or the processing of sequences with variable and potentially long-range dependencies.

B.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) incorporate recurrent connections, enabling the output from a previous time step h_{t-1} to be an input to the current step: $h_t = g_\theta(h_{t-1}, x_t)$. This architectural feature allows information to persist and be transformed across a number of time steps that can scale with the input sequence length n. In this context, each transformation $\mathcal{T}_i$ corresponds to the recurrent computation at a given time step. Consequently, the effective computational depth $D_{\text{eff}}(\text{RNN})$ can reach $O(n)$. This substantial increase in sequential depth capacity empowers RNNs to effectively model sequential data and tasks characterized by temporal dependencies, such as string manipulations or sequential decision-making, far surpassing the capabilities of standard MLPs. RNNs are considered "recurrence-complete" as they possess the capability to simulate any one-term recurrent function $h_t = g'(h_{t-1})$, given adequate network capacity and appropriate non-linear activation functions, a property underpinned by the Universal Approximation Theorem.

B.4 Transformers

Standard Transformer architectures, despite their considerable success driven by the attention mechanism, operate with a fixed number of layers, m, akin to MLPs. While attention allows for parallel processing of tokens within each layer and sophisticated information integration from the entire input sequence $x_{1:t}$, the number of sequential transformation steps (i.e., layers) remains constant. The final layer output $h_t^{(m)}$ is a function of the input $x_{1:t}$ but not directly of a temporally prior hidden

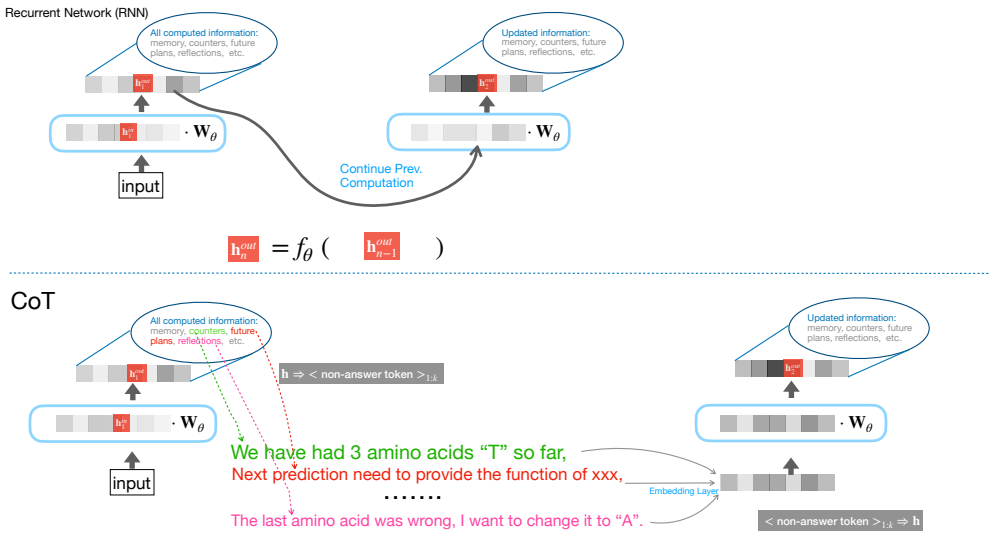


Figure 6: Illustration of how CoT approximate recurrent computation in LMs.

state $h_{t-1}^{(m)}$ in the same way as in an RNN. The transformations $\mathcal{T}_i$ in a Transformer correspond to the operations within each of its m layers. Therefore, the $D_{\text{eff}}(\text{Transformer})$ is $O(m)$, which is $O(1)$ with respect to the input sequence length n. This characteristic limits their inherent ability to solve tasks that require a computational depth greater than m without architectural modifications or prompting strategies that simulate deeper recurrence.

C Chain of Thought: Towards Unbounded Expressive Power in Theory

The advent of Chain of Thought (CoT) prompting has been empirically shown to significantly enhance the reasoning capabilities of Large Language Models (LLMs). Beyond a mere prompting strategy, CoT can be interpreted from a computational theory perspective as a mechanism that fundamentally reshapes the Transformer's operational dynamics at runtime. In theory, CoT endows autoregressive LLMs with the potential for unbounded expressive power by dynamically altering their computational depth, time complexity, and effective memory.

C.1 Augmenting Computational Depth via Simulated Recurrence

Standard Transformer architectures possess a fixed number of layers, m, constraining their *Effective Computational Depth*, $D_{\text{eff}}(\text{Transformer})$, to $O(m)$, or $O(1)$ with respect to input sequence length n. This architectural limitation inherently restricts their capacity for tasks demanding extensive sequential reasoning steps exceeding m.

Chain of Thought circumvents this limitation by simulating a recurrent computational process. The core mechanism involves:

1. Encoding the model's internal computational state (hidden state h) at the end of a reasoning step into a sequence of natural language tokens $(o_1, o_2, ..., o_k)$ that form the "thought". This can be represented as $h \to o_{1:k}$.
2. Appending these tokens to the ongoing input sequence, effectively creating an extended context $x_{n+k} = (x_1, ..., x_n, o_1, ..., o_k)$.
3. The model then processes this extended sequence, and the CoT string $o_{1:k}$ is decoded back into a computational state h', effectively $o_{1:k} \to h'$.

This iterative process, $h^{(t)} \to o_{1:k}^{(t)} \to h^{(t+1)}$, emulates the recurrent connection $h_t = g(h_{t-1}, x_t)$ found in RNNs, where the model can resume and refine its computation based on the explicitly articulated intermediate state. This transformation allows the internal hidden state h, containing comprehensive computational information, to be first externalized into a sequence of "non-answer tokens" (the CoT string). These tokens, which can articulate ongoing reasoning, memory states, or even self-corrections, are then embedded and re-processed by the model to inform subsequent computations, thereby achieving the $o_{1:k} \to h'$ step. Consequently, the effective computational depth is no longer rigidly tied to the number of physical layers but rather to the length of the generated thought sequence, $T(n)$. The D_{eff}(LLM+CoT) can thus become $O(T(n))$, theoretically allowing for arbitrarily deep sequential computation, limited primarily by factors like context window length or coherence of the generated thought, rather than by fixed architectural depth. This approximated recurrence is what notably improves the model's computational capacity.

C.2 Impact on Computational Time

The simulation of recurrence through CoT inherently impacts computational time. Standard autoregressive generation processes one token at a time. By extending the input sequence with intermediate thought tokens $o_{1:k}$, the CoT process necessitates additional computational steps for both generating these thought tokens and subsequently processing them as part of the augmented input. If an input of length n requires $T(n)$ CoT steps (where each step might involve generating multiple tokens), the overall time complexity for processing is increased. The time complexity is enhanced to approximately $O(n + T(n)_{\text{tokens}})$, where $T(n)_{\text{tokens}}$ represents the total number of tokens generated as part of the chain of thought. While this increases the computational cost, it is this very extension that facilitates the enhanced reasoning depth.

C.3 Expanding Effective Memory through Articulated States

Transformers, through their attention mechanism, can access all previous tokens in their context window. However, the critical information from intermediate computational steps might be diffusely encoded within hidden states and not directly accessible or interpretable for subsequent, distinct reasoning phases that exceed the fixed-layer depth.

CoT provides a mechanism to externalize these intermediate computational states. By articulating the hidden state h (or critical aspects of it, such as memory, counters, or reflections) into natural language strings $o_{1:k}$, these strings act as an explicit, persistent memory trace within the context window. The model can then attend to these textual representations of its prior states in subsequent steps. This allows for a form of "read-write" memory capability where the model "writes" its state as text and "reads" it back to continue computation. This capability is crucial for tasks that require maintaining and manipulating information over extended reasoning chains, effectively emulating more powerful memory structures. For instance, LLMs with CoT can achieve tape-like memory access via attention on the generated text, enabling them to tackle tasks at higher levels of the Chomsky hierarchy, such as context-sensitive tasks, which are typically beyond the reach of standard Transformers without CoT. The expressiveness of natural language is posited to be sufficiently universal to encode diverse types of information, including reasoning states, memory, and intermediate results, which is a critical assumption for this mechanism's success.

In essence, CoT theoretically empowers Transformers by allowing them to dynamically construct deeper computational graphs and manage more explicit memory traces at runtime, thereby transcending some of their inherent architectural limitations.

D Background on *De Novo* Peptide Sequencing

Peptide sequencing from tandem mass spectrometry (MS/MS) data is a cornerstone of proteomics (Aebersold & Mann, 2003), driving advancements from fundamental biological discovery to drug development (Aebersold & Mann, 2003; Ng et al., 2023). The goal is to determine the amino acid sequence of peptides directly from their mass spectra. Traditional database search algorithms (Eng et al., 1994; Perkins et al., 1999; Cox & Mann, 2008; Zhang et al., 2012), though widely used, cannot identify novel peptides absent from reference databases. This limitation impedes progress in emerging applications such as *de novo* antibody characterization (Beslic et al., 2022), neoantigen

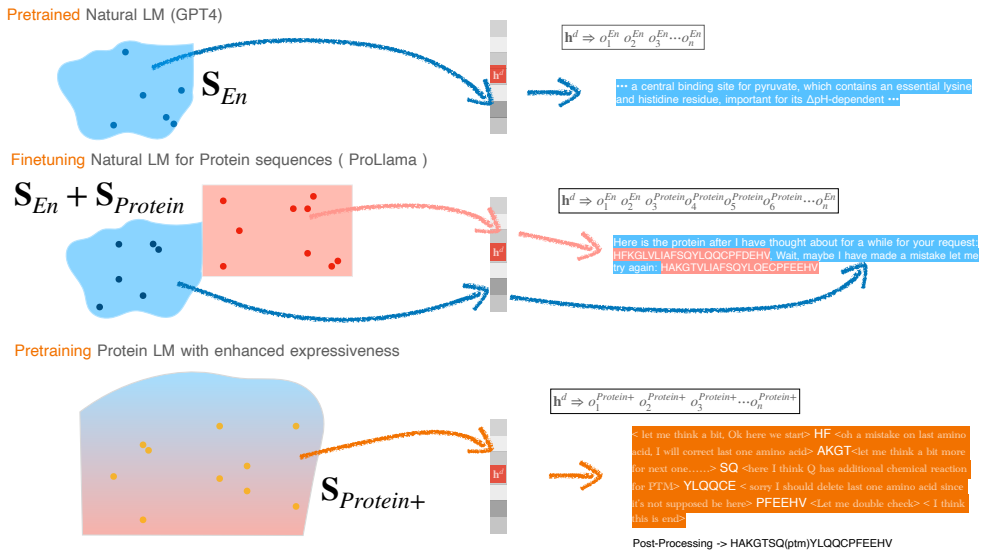


Figure 7: Illustration of the disjoint expressive spaces between natural language (NL) and protein sequences under fine-tuning. CoT reasoning capabilities learned in $\boldsymbol{L}_{\text{NL}}$ do not transfer into $\boldsymbol{L}_{\text{protein}}$, resulting in no shared reasoning ability.

discovery (Karunratanakul et al., 2019), and metaproteomics (Hettich et al., 2013). In contrast, *de novo* peptide sequencing infers sequences without relying on any database, making it indispensable for comprehensive and unbiased proteomic analysis.

The advent of deep learning (LeCun et al., 2015) has transformed *de novo* sequencing, surpassing traditional algorithmic approaches (Dančík et al., 1999; Ma et al., 2003; Frank & Pevzner, 2005). Early models such as DeepNovo (Tran et al., 2017) employed CNNs and LSTMs, while more recent efforts have shifted to Transformer-based architectures (Vaswani et al., 2017), which offer improved scalability and performance. These models generally fall into two categories: autoregressive (AT) and non-autoregressive (NAT). AT model–such as Casanovo (Yilmaz et al., 2022; 2023) and its derivatives (e.g., AdaNovo (Xia et al., 2024), HelixNovo (Yang et al., 2024), InstaNovo (Eloff et al., 2023), ContraNovo (Jin et al., 2024))–predict one amino acid at a time, often achieving high precision through architectural refinements. NAT models like PrimeNovo (Zhang et al., 2025a), on the other hand, generate sequences in parallel, enabling bidirectional context and faster inference (Gu et al., 2017; Xiao et al., 2023). Despite this progress, both paradigms largely lack the capacity for explicit intermediate reasoning, which may limit generalization in noisy or uncertain scenarios.

Section 3.1 explains why these properties—gold labels at every position, residues verifiable from the spectrum, and local, structured errors—make the task a natural testbed for reflection pretraining, and gives the formal problem definition.

E Why Finetuning Fails to Transfer Reasoning to Protein Sequences

Let $\boldsymbol{L}_{\text{NL}}$ denote the natural language space (e.g., English), and $\boldsymbol{L}_{\text{protein}}$ denote the protein language space, comprising amino acid tokens with limited compositional structure. Large language models (LLMs) such as LLaMA or GPT-4 are pretrained to reason over $\boldsymbol{L}_{\text{NL}}$ and can perform advanced tasks through CoT prompting–generating intermediate reasoning steps, storing temporary variables, performing error correction, and maintaining self-reflection, all in natural language.

In approaches like **ProLlama**, a pretrained natural language model is fine-tuned on protein sequences, introducing a new vocabulary $\mathcal{V}_{\text{protein}}$ and dataset distribution $\mathbb{S}_{\text{protein}} \subset \mathcal{X}^*_{\text{protein}}$. While the transformer architecture remains shared, the representations fine-tuned on proteins diverge significantly from

those learned in natural language. As illustrated in Figure 7, the model's learned expressiveness splits into two largely non-overlapping subspaces:

$$\mathbb{S}_{\text{NL}} \cap \mathbb{S}_{\text{protein}} \approx \emptyset \tag{2}$$

This results in a model that performs well on surface-level protein generation but cannot leverage its natural language reasoning skills–because those abilities remain confined to the $\boldsymbol{L}_{\text{NL}}$ space and are inaccessible in $\boldsymbol{L}_{\text{protein}}$.

To understand this intuitively, consider a bilingual LLM pretrained on English. If we fine-tune it on Chinese texts without joint training or alignment, the model learns to generate Chinese tokens fluently, but cannot reason about Chinese using English thinking patterns. For example, tokens like "let me double check" or "I made a mistake" have no meaningful grounding in the Chinese token space unless the model was trained to use them cross-lingually. The model ends up with two islands of knowledge–English and Chinese–that cannot communicate or share reasoning capacity.

The same phenomenon occurs in protein modeling. ProLlama retains its CoT reasoning ability in English but cannot express or apply that reasoning in $\boldsymbol{L}_{\text{protein}}$. As a result, it cannot generate intermediate reasoning tokens like `<reflect>` or revise outputs mid-generation–capabilities that are essential for self-correction, debugging, or guided sequence exploration.

In contrast, our reflection-pretraining approach explicitly introduces auxiliary reasoning tokens $\mathcal{V}_{\text{reflect}}$ and trains the model to use them within the protein generation process. This expands the language from $\boldsymbol{L}_{\text{protein}}$ to an augmented form:

$$\boldsymbol{L}_{\text{protein+}} = (\mathcal{G}_{\text{protein}}, \mathcal{V}_{\text{protein}} \cup \mathcal{V}_{\text{reflect}}) \tag{3}$$

and correspondingly enlarges the expressive space from $\mathbb{S}_{\text{protein}}$ to $\mathbb{S}_{\text{protein+}}$, allowing the model to generate and interpret internal reasoning sequences natively in the protein domain:

$$|\mathbb{S}_{\text{protein+}}| \gg |\mathbb{S}_{\text{protein}}| \tag{4}$$

F EXPERIMENTS

Model Training. We adopt the same architecture and optimization configuration as in prior work of training Transformer for De novo Peptide Sequencing. All inputs–including MS/MS peaks, precursor features, and amino acid tokens–are first embedded into a shared 400-dimensional latent space.

The model is built on a 9-layer Transformer, with each layer comprising 8 attention heads and a 1024-dimensional feedforward network. This architecture forms the backbone for both the encoder and decoder components.

Training is conducted with a batch size of 1600 and an initial learning rate of 4e-4. The learning rate linearly warms up during the first epoch before decaying via a cosine schedule. Optimization is performed using AdamW (Kingma & Ba, 2014), and all models are trained for 30 epochs on eight A100 GPUs.

These hyperparameters–embedding dimension, transformer depth, attention head count, and learning rate policy–are kept fixed across all experiments unless explicitly stated. Additional implementation details can be found in our released codebase.

***De Novo* Baselines.** We evaluate our approach against a range of *de novo* peptide sequencing methods, each of which incorporates domain-specific knowledge or architectural modifications tailored to biological data:

- **Peaks** (Ma et al., 2003) employs a protein database search strategy that combines tandem mass spectrometry with homology-based alignment. This approach excels in navigating ambiguity in peptide identification using prior biological information.
- **DeepNovo** (Tran et al., 2017) combines convolutional and recurrent neural networks with peptide-specific preprocessing and adopts a species-wise cross-validation scheme.
- **PointNovo** (Qiao et al., 2021) builds on point-cloud processing ideas to handle varied-resolution spectra without increasing model complexity, showcasing robustness through biologically-informed design.

- **InstaNovo** (Eloff et al., 2023) introduces a diffusion model that improves performance through iterative refinement of predicted sequences.
- **HelixNovo** (Yang et al., 2024) extends CasaNovo by modifying the MS/MS input to explicitly encode complementary b/y ion relationships–information that is biologically grounded in peptide fragmentation rules.

In contrast to these models, our reflection-pretrained Transformer introduces no biological heuristics or domain-specific preprocessing. Instead, it augments a standard model's reasoning capacity via token-level reflection and self-correction. This purely cognitive enhancement allows the model to outperform bio-inspired baselines without incorporating any explicit biochemical priors.

G LIMITATIONS

Our approach has several limitations. First, it relies on gold-label peptides to place errors and supervise their corrections. Labels from database search carry a small false discovery rate, and extending reflection pretraining to tasks without an exact reference sequence (e.g., protein design) would require a different source of correction targets. Second, the computational cost of pretraining with reflection mechanisms is higher than that of traditional methods, as the inclusion of non-answer tokens increases sequence length and training complexity. Lastly, while our framework facilitates human-in-the-loop interactions, its practical adoption in real-world biological workflows hinges on proteomics experts' ability to identify suspect residues and interpret the reflection outputs, which may require additional training or domain expertise. These limitations highlight several avenues for future research, including improved dataset curation, optimization of computational efficiency, and the development of more user-friendly interfaces to enhance the accessibility and impact of biological reasoning models.

H BROADER IMPACT

The capacity to give *de novo* peptide sequencing models self-correction holds transformative potential for society, primarily by accelerating biomedical discovery for new drugs and therapies, and advancing biotechnology through the design of novel enzymes and materials. This advancement is not without significant ethical responsibilities, chief among them mitigating dual-use risks related to the creation of harmful biological agents, ensuring the safety of AI-designed sequences, and promoting equitable access to these powerful tools. Consequently, the path forward requires a concerted effort towards responsible innovation, incorporating robust safety protocols, ethical guidelines, and mechanisms for human oversight-such as the human-in-the-loop generation. Our work supports harnessing these capabilities for broad societal benefit while safeguarding against potential harms.