HELM: Hierarchical Encoding for mRNA Language Modeling

Mehdi Yazdani-Jahromi [*] [†]
Department of Computer Science
University of Central Florida
yazdani@ucf.edu

Mangal Prakash [*]
Johnson & Johnson Innovative Medicine
mpraka12@its.jnj.com

Tommaso Mansi
Johnson & Johnson Innovative Medicine
tmansi@its.jnj.com

Artem Moskalev [‡]
Johnson & Johnson Innovative Medicine
amoskal2@its.jnj.com

Rui Liao [‡]
Johnson & Johnson Innovative Medicine
rliao2@its.jnj.com

Abstract

Messenger RNA (mRNA) plays a crucial role in protein synthesis, with its codon structure directly impacting biological properties. While Language Models (LMs) have shown promise in analyzing biological sequences, existing approaches fail to account for the hierarchical nature of mRNA's codon structure. We introduce Hierarchical Encoding for mRNA Language Modeling (HELM), a novel pre-training strategy that incorporates codon-level hierarchical structure into language model training. HELM modulates the loss function based on codon synonymity, aligning the model's learning process with the biological reality of mRNA sequences. We evaluate HELM on diverse mRNA datasets and tasks, demonstrating that HELM outperforms standard language model pre-training as well as existing foundation model baselines on six diverse downstream property prediction tasks on average by around 8%. Additionally, HELM enhances the generative capabilities of language model, producing diverse mRNA sequences that better align with the underlying true data distribution compared to non-hierarchical baselines.

1 Introduction

RNA analysis is becoming increasingly important in molecular biology [40; 27]. Messenger RNA (mRNA) is of particular interest due to its unique role in protein synthesis [60]. mRNA consists of triplets of nucleotides, called codons, which directly translate to protein amino acids through a surjective many-to-one mapping (Figure 1 *left*). This results in multiple synonymous mRNAs that

[*] Equal contribution as first authors.
[†] This work was done while the author was an intern at Johnson & Johnson.
[‡] Equal contribution as last authors.

AI for New Drug Modalities at NeurIPS 2024.

encode identical amino acid sequences while exhibiting distinct physical and biological properties at the nucleotide level [11]. The role of mRNA, serving as an intermediary between DNA and proteins, underlies its therapeutic potential in applications such as vaccines and gene therapies [12; 52], while also presenting challenges for analysis and optimization [43].

Language Models (LMs) have emerged as powerful tools for analyzing biological sequences, with notable successes in protein [23; 24; 39; 34] and DNA [47; 71] research. Despite the importance of mRNA, the field still lacks specialized LMs tailored for its analysis. Existing RNA LMs [38; 15] focus on non-coding sequences and do not account properly for codon hierarchy (Fig. 1 *right*) which, as we demonstrate, falls short when dealing with mRNA tasks. In this work, we aim to address this gap in mRNA language modeling by focusing specifically on the unique challenges presented by mRNA sequences.

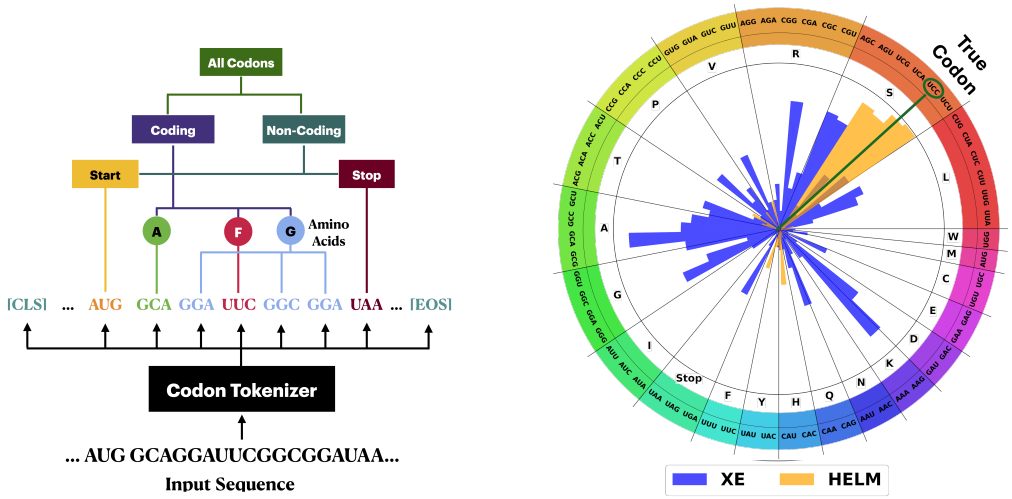

Figure 1: Hierarchical encoding of mRNA sequences as a biological prior. ***Left:*** Hierarchical structure of codons for HELM and codon tokenization. The tree diagram illustrates the codon hierarchy used in the HELM approach, categorizing codons into Start, Coding (grouped by amino acids), and Stop. This hierarchy informs the loss calculation. The codon tokenizer demonstrates the process of converting an mRNA input sequence into codon tokens for modeling. ***Right:*** Codon prediction probabilities on a amino acid codon wheel. Segments represent amino acids, bars represent codons. Orange: HELM approach; Blue: cross-entropy (XE) loss. Bar height indicates probability. Non-hierarchical XE model assigns high probabilities to non-synonymous codons for masked tokens, while HELM assigns high probabilities to synonymous codons, even when errors occur.

To address the limitations of existing bio-language modeling methods, we introduce Hierarchical Encoding for mRNA Language Modeling (HELM), a novel pre-training strategy for mRNA sequences. Rather than directly adopting natural-language pre-training methods such as Masked Language Modeling (MLM) [21; 44; 16] or Causal Language Modeling (CLM) [57; 58; 10], HELM begins by recognizing and formalizing the hierarchical structure of mRNA sequences based on codon synonymity. HELM modulates the CLM or MLM loss based on a codon's position in this hierarchy, treating errors between synonymous codons as less significant than those resulting in different amino acids. This approach requires minimal modifications to standard pipelines while aligning the model training process with the intrinsic hierarchical structure of mRNA data (Figure 1 *right*). In addition, to facilitate further development of mRNA LMs, we provide a consistent comparison of various tokenization methods, hierarchical and standard pre-training strategies, and different language model architectures, including recent Mamba and Hyena models [31; 56].

To demonstrate the practical advantages of HELM, we focus on the important domain of antibody mRNAs where codon distribution can significantly impact physical and therapeutic properties [70]. We conduct comprehensive experiments, evaluating our hierarchical model against its non-hierarchical counterparts pre-trained with natural-language CLM and MLM as well as against other state-of-the-art RNA foundation models. Our results show that hierarchical pre-training yields significantly better representations of antibody-encoding mRNA and even excels on non-antibody mRNA data. Our hierarchy-aware model outperforms non-hierarchical counterparts pre-trained with natural-language

CLM and MLM as well as other state-of-the-art RNA foundation models by around 8% on average across six diverse downstream property prediction tasks while requiring up to 2 times fewer model parameters. Beyond its excellent discriminative performance, we demonstrate that HELM-based CLM pre-training yields strong generative models capable of producing diverse mRNA sequences while preserving closer alignment with the true data distribution. Ultimately, HELM addresses the limitations of current bio-LMs by effectively capturing the hierarchical structure of mRNA as a strong biological prior, leading to improved performance in both predictive and generative tasks while requiring minimal modification of the pre-training.

To sum up, we make the following contributions:

- We highlight the importance of biological hierarchy when modeling mRNA sequences and demonstrate that standard natural-language pre-training methods do not account for this inherent biological structure, leading to suboptimal performance in mRNA tasks.
- We propose Hierarchical Encoding for mRNA Language Modeling (HELM), a novel pre-training strategy that incorporates mRNA hierarchical structure into language model pre-training. HELM modulates the CLM or MLM loss based on codon hierarchy, aligning the model's learning process with the biological structure of mRNA sequences.
- We provide a consistent comparison of various tokenization methods, hierarchical and standard pre-training strategies, and different LM architectures to facilitate further development of mRNA models.
- We demonstrate the advantages of representation learning with HELM where our method yields substantial downstream performance improvement on multiple diverse mRNA datasets spanning property prediction and sequence generation tasks.

2 Related works

Pre-training bio-language models Recent advancements in LMs have significantly improved the analysis of biological sequences such as DNA, RNA, and proteins [37]. The standard approach for bio-language modeling adopts pre-training strategies from natural language; state-of-the-art protein or DNA LMs rely on Masked Language Modeling [44; 39; 20] or Causal Language Modeling [10; 39] and pre-train representations on large bio-databases [4; 67; 19; 61]. In addition, the modeling choices of pre-training bio-LMs involve network architecture and tokenization strategy. Regarding architectures, transformer-based models have been widely adopted [23; 39], while recent works explore structured state-space [54; 63] and long-convolutional models [48; 47; 46] to better handle long-range dependencies in biological sequences. Tokenization strategies vary from character-level approaches [23; 39] to k-mer tokenization [20] and Byte Pair Encoding [71]. These existing pre-training and modeling choices primarily adapt natural language paradigms to biological sequences, overlooking the incorporation of inherent biological structures crucial for mRNA analysis. Moreover, the lack of consistent comparisons between these various approaches hinders the development of effective mRNA LMs. Our work addresses these limitations with HELM, a pre-training strategy incorporating mRNA's hierarchy into the training, and we also provide analysis of various pre-training and modeling choices to facilitate further development of mRNA LMs.

RNA language models Unlike the many protein and DNA LMs [39; 23; 20; 47], there is a noticeable lack of models specifically designed for mRNA due to the smaller, specialized datasets available. Existing RNA models focus on specific RNA types or regions, such as RNA-FM [14] and RINALMO [55] for non-coding RNAs, UTR-LM [17] and UTRBERT [69] for untranslated regions, and SpliceBERT [15] for precursor mRNA. While these models are valuable, they are not tailored for protein-coding mRNAs. Proprietary models such as CodonBERT [38] have been developed but they do not utilize the natural hierarchy of mRNA sequences. Our work addresses these limitations by introducing HELM to explicitly incorporate the biological knowledge of mRNA structure into the pre-training process of a LM.

Modeling hierarchical data Hierarchical relationships are important across various domains, and several methods have been proposed to model them effectively. In computer vision, approaches range from embedding-based methods like DeViSE [26], which maximizes cosine similarity between image and label embeddings, to geometric approaches that rely on hyperbolic spaces [42; 2] or

hyperspheres with structural constraints [6; 7]. Hierarchical loss functions, such as hierarchical cross-entropy (HXE) [8; 28], have been applied to tasks like image classification and astrophysical transient classification [68]. In natural language processing, hierarchical structures have been modeled both implicitly through contrastive learning [41; 29] and explicitly using hyperbolic losses [33] or hierarchical softmax [62]. While these methods show great promise in their respective domains, they are not tuned for biological data. Our HELM approach addresses this gap by extending the hierarchical learning to mRNA sequences, effectively capturing the codon hierarchy in the pre-training process. This demonstrates the potential of incorporating biological priors into language modeling pre-training, paving the way for more accurate and biologically relevant models in this domain.

3 Hierarchical Cross-Entropy Loss

Hierarchical cross-entropy (HXE) is a technique for training neural networks that incorporates hierarchical information into the learning process. HXE was originally proposed in computer vision for image classification [8] tasks, where the data labels can be organized in a hierarchical structure, such as a tree. When the hierarchy H is structured as a tree, HXE allows for a unique factorization of the categorical distribution $p(C)$ over the classes in terms of the conditional probabilities along the path from each class to the root of the tree. Specifically, for any leaf node/class C with a path $C^{(0)} = C, \ldots, C^{(h)} = R$, the probability of C can be written as $p(C) = \prod_{l=0}^{h-1} p(C^{(l)} \mid C^{(l+1)})$, where h is the height of the node C. The conditional probabilities can be expressed in terms of the class probabilities as:

$$p(C^{(l)} \mid C^{(l+1)}) = \frac{\sum_{A \in \text{Leaves}(C^{(l)})} p(A)}{\sum_{B \in \text{Leaves}(C^{(l+1)})} p(B)}, \tag{1}$$

where $\text{Leaves}(C)$ denotes the set of leaf nodes within the subtree rooted at C. To incorporate hierarchical information directly into the loss function, the classifier's output can be factorized according to the hierarchical structure, and the total loss can be defined as a reweighted sum of the cross-entropies of these conditional probabilities along the hierarchical path. This leads to the hierarchical cross-entropy (HXE) loss, effectively incorporating the hierarchical information into the learning process:

$$L_{HXE}(p, C) = -\sum_{l=0}^{h-1} \lambda(C^{(l)}) \log p(C^{(l)} \mid C^{(l+1)}), \tag{2}$$

where $\lambda(C^{(l)})$ is the weight associated with the edge between nodes $C^{(l+1)}$ and $C^{(l)}$.

Formalizing the hierarchcy for mRNA sequences We present a novel formalism for constructing a multi-level hierarchical tree $H = (V, E)$ that captures the structured relationships within mRNA sequences at the codon level (see Fig. 1 left, Appendix Fig. 4) and well-rooted in biology [18]. The root node R of this tree represents a node corresponding to all codons. This root node has two immediate child nodes: coding codons C_{coding} and non-coding codons $C_{\text{non-coding}}$. The non-coding codons $C_{\text{non-coding}}$ further branch into two child nodes: the start codon node C_{start} and the stop codon node C_{stop}. The start codon node C_{start} has a single leaf node corresponding to the codon AUG. The stop codon node C_{stop} has three leaf nodes corresponding to the codons UAA, UAG, and UGA. On the other side of the hierarchy, the coding codons C_{coding} are organized into multiple child nodes, each corresponding to a specific amino acid A_j. Each amino acid node A_j has a number of child nodes, each corresponding to a synonymous codon that encodes for A_j. Let $\text{Codons}(A_j) = \{C_{i_1}, C_{i_2}, \ldots, C_{i_m}\}$ denote the set of all synonymous codons for the amino acid A_j. These synonymous codons preserve the same amino acid, allowing for degeneracy in the genetic code. The complete hierarchical structure is thus represented by the tree $H = (V, E)$, where V includes the root node, all internal nodes (corresponding to coding and non-coding codons, start/stop codons, and amino acids), and all leaf nodes (corresponding to individual codons). The edges E define the hierarchical relationships between these nodes.

Hierarchical MLM and CLM pre-training for mRNA We adapt the Hierarchical Cross-Entropy to incorporate the biological structure of mRNA into Masked Language Modeling and Causal

Language Modeling pre-training strategies. For the hierarchical MLM, we randomly mask a portion of codons in the input sequence and train a model to predict the masked codons supervising it by the HXE loss. In the hierarchical CLM, a model predicts the next codon given the previous ones with the HXE loss function.

The hierarchical structure of mRNA informs the HXE objective, allowing the model's output token probabilities to be factorized according to the tree structure. Then, the HXE loss uses Eq.1 and Eq.2 weighting the MLM or CLM errors differently based on the codon's position in the hierarchy. We employ a practical weighting function $\lambda(C) = \exp(-\alpha h(C))$ where $h(C)$ is the height of the node C in the hierarchy and ($\alpha > 0$). The value of α determines how much weight is given to information at different hierarchical levels. This weighting function is applied in both MLM and CLM pre-training, allowing HELM to capture both local codon-level information and the global hierarchical structure of the genetic code. By incorporating this hierarchical loss into pre-training, HELM encourages the model to learn biologically meaningful representations of mRNA sequences, which closely aligns the model's learning with the biological structure of mRNA sequences. Note that our approach does not require modifying the architecture or tokenization of a model and permits operating on the standard codon vocabulary. mRNA hierarchy modeling with HXE will be referred as HELM from here on.

4 Experiments

In our experiments, we first compare various mRNA language modeling configurations by exploring different tokenization strategies and model architectures (Transformer, Mamba, Hyena) with MLM and CLM objectives. We demonstrate how HELM significantly improves mRNA property prediction over its non-hierarchical counterparts. Next, we investigate the effectiveness of hierarchical mRNA encoding, evaluating it through synonymous sequence clustering. Finally, we present generative evaluations include assessing generated sequence quality and diversity via Frechet Biological Distance (FBD), as well as testing the preservation of functional properties in generated mRNA sequences.

Datasets Unlike protein and DNA modality which boast many large-scale datasets, mRNA does not have many curated high-quality pre-training datasets. For this reason, we curated the OAS database [50] which contains antibody mRNA data from over 80 different studies with around 2 billion unpaired and 1.5 million paired sequences from various species. Although prior studies have curated this database on protein level [59; 65; 35] in the context of antibody-protein language modeling, a high-quality curated version of corresponding mRNA data does not exist. Due to its origin in high-throughput sequencing studies, the mRNA sequences in OAS exhibit varying levels of sequencing errors and noise. We conducted an extensive curation process to ensure high-quality mRNA data for pre-training. Dataset curation details can be found in Appendix A.8. Post-curation, we end up with 15.3 million mRNA sequences for pre-training mRNA LMs.

For downstream evaluation, we use the following six mRNA datasets, including antibody (Ab) encoding mRNA and general mRNA sequences:

- Ab1 [1] includes 1200 Ab-mRNA sequences with protein expression labels.
- Ab2 [1] contains 3442 Ab-mRNA sequences with protein expression labels.
- MLOS [38] has 167 viral mRNA sequences with expression in HeLa cells.
- iCodon [22] includes 65357 mRNA sequences with thermostability profiles from humans, mice, frogs, and fish.
- Tc-Riboswitches [30] consists of 355 riboswitch mRNA sequences with switching factor measurements.
- mRFP [49] has 1459 mRNA sequences with protein production levels for various gene variants in E. coli.

All downstream evaluation datasets provide regression labels for evaluating the quality of mRNA property prediction.

Evaluation We ran all models for 3 random data splits (train:val:test split of 70:15:15) and we report average performance across splits. When evaluating property prediction tasks, we use Spearman rank

correlation as a standard metric under commonly used probing methodology [45; 13; 51; 32] with a TextCNN [36] head to assess representation quality and transferability. For generative evaluation, we take the metrics used to evaluate generative models for DNA [66; 65] and we adopt them for mRNA data. We provide comprehensive experimental details in Appendix A.3.

4.1 Pre-training and modeling choices

We first aim to establish a strong non-hierarchical mRNA language model as a baseline. For this, we aim for consistent comparison between various standard pre-training strategies, different architectures, and various tokenization methods. We conduct the comparison on the property prediction mRNA tasks where we pre-train all our language language models on the same curated OAS data and test them on downstream property prediction tasks.

4.1.1 Experimental details

Pre-training objective To train an mRNA LM, we start with two standard pre-training objectives: Masked Language Modeling (MLM) and Causal Language Modeling (CLM). MLM is an inherently bidirectional pre-training method that relies on model learning to capture context from both directions of a sequence. CLM, on the other hand, is unidirectional, predicting each token based on previous tokens. We evaluate both objectives to determine their effectiveness in capturing the biological structure of mRNA sequences.

Model architectures We evaluate various state-of-the-art architectures for sequence modeling: Transformer [58], Hyena [56], and Mamba [31]. Transformers support both MLM and CLM objectives, while Hyena and Mamba are primary designed for CLM. All models use 50M parameters, balancing performance and efficiency. We found that models of this scale can outperforms larger existing models while maintaining reasonable run-times (see Appendix A.7 and Table 7). Detailed pre-training information is available in Appendix A.2.

Tokenization Inspired by protein and DNA language modeling, we explored various tokenization strategies for mRNA sequences: nucleotide, codon-level, and 6-mer. We observed that codon-level tokenization consistently outperforms other strategies (results in Appendix Sec. A.4). We hypothesize that the superior performance of codon tokenization stems from codons' role as the fundamental units of the genetic code, directly mapping to amino acids during protein synthesis. In addition, the codon level tokenization allows to naturally capture the variability in wobbling nucleotides, where changes in the third nucleotide do not always alter the amino acid produced [5; 51]. Here on, we adopt codon-level tokenization as the default strategy for our models. This way, our vocabulary includes $4^3 = 64$ codon tokens, representing all possible nucleotide combinations, plus special tokens for start, stop, padding, masking, and unknown characters.

Baselines We additionally compare trained models against three state-of-the-art public foundation models: RNA-FM, SpliceBERT, and CodonBERT. While RNA-FM and CodonBERT are 100M parameter models, SpliceBERT is a 20M parameter model. We choose these baselines as CodonBERT is the only existing mRNA LM publicly available while SpliceBERT is pre-trained on pre-mRNA, and RNA-FM is known to be one of the strongest baselines in recent works because of its reported strong performance across multiple domains [47; 25; 9]. Additionally, we include a simple one-hot encoding-based baseline to evaluate how much LMs performances surpass basic sequence representation methods commonly used [32; 9].

4.1.2 Results

We first compare the performance of the transformer-based mRNA language models to state-space Mamba and long-convolutional Hyena-based architectures. In Table 1, we can see that transformer-based models outperform Hyena and Mamba on 4 out of 6 datasets, with the transformer CLM trailing closely behind Mamba on the remaining 2 datasets. The superior performance of transformers is likely due to the relatively short sequence lengths ($\sim$ 1000 tokens) of mRNA sequences in both pre-training and downstream datasets. Transformers, with their explicit global attention mechanism between each pair of tokens, excel at capturing fine-grained interactions in these relatively short sequences. In contrast, Hyena and Mamba architectures are optimized for handling much longer

Table 1: Performance of mRNA LMs trained with standard cross-entropy loss on downstream tasks. Our models outperform general-purpose RNA baselines across various mRNA tasks. MLM and CLM show comparable results. Bold: best performance. Italics: second-best. Missing values: model unable to process due to sequence length limitations.

Model	Ab1	Ab2	MLOS	iCodon	Tc-Riboswitches	mRFP
Baseline Models						
One-hot	0.431	0.421	0.462	0.152	0.378	0.511
RNA-FM	0.595	0.515	-	-	0.504	0.527
SpliceBERT	0.652	0.542	-	-	0.418	0.596
CodonBERT	0.686	0.557	0.543	0.350	0.502	0.770
mRNA LMs (Ours)						
Transformer XE (MLM)	*0.748*	**0.599**	**0.653**	0.503	**0.569**	0.753
Transformer XE (CLM)	**0.752**	*0.597*	0.611	0.498	*0.531*	0.815
Hyena XE	0.743	0.594	*0.623*	**0.526**	0.517	**0.844**
Mamba XE	0.742	0.585	0.620	*0.509*	0.520	*0.823*

sequences via efficient convolutional or state-space mechanisms and may not fully leverage their strengths in this shorter sequence regime owing to their implicit handling of sequence dependencies, despite performing competitively. Given that transformer-based models generally show better performance than Hyena and Mamba models, we select transformer-based MLM and CLM models for all subsequent experiments.

Additionally, our results indicate that for the transformer models, both MLM and CLM perform comparably on 3 out of 6 datasets (Ab1, Ab2, and iCodon), while MLM outperforms CLM on MLOS and Tc-Riboswitches data, and CLM surpasses MLM on the mRFP dataset. The comparable performance of MLM and CLM across most cases suggests that the statistical and structural properties of mRNA are adequately captured by both unidirectional and bidirectional context modeling. This implies that the underlying biological information in mRNA sequences may be robust to the specific directional biases introduced by MLM or CLM pre-training, with neither approach consistently outperforming the other across varied bio-sequence contexts.

We then compare the performance of our transformer, Hyena, and Mamba models against various foundation model baselines. Table 1 illustrates that our transformer, Mamba, and Hyena-based LMs outperform all baselines by 5-17 percentage points across all downstream tasks which highlights the value of our curated mRNA pre-training dataset, tokenization, and architectural design choices.

4.2 HELM improves mRNA property prediction

Next, we evaluate the effect of learning hierarchical mRNA encoding with HELM for the mRNA property prediction tasks. As can be observed from Table 2, training with biological hierarchical prior consistently improves model performance for both MLM and CLM objectives on all six datasets compared to baseline models trained with standard non-hierarchical cross-entropy. For causal language modeling, HELM-CLM outperforms the non-hierarchical XE-CLM model in five out of six datasets. For the masked language modeling, HELM-MLM outperforms the non-hierarchical XE-MLM model in all datasets. Within hierarchy-aware models, we again observe the same trend that MLM and CLM HELM models perform comparably for most tasks with each performing best on three of the six datasets.

When and why learning hierarchical mRNA encoding is most effective? Hierarchical HELM modeling improves the performance in all predictive tasks we considered. However, the extent of improvement varies across datasets. Greater gains are seen in MLOS, Tc-Riboswitches, and mRFP datasets, while Ab1, Ab2, and iCodon show smaller improvements (see Table 2). We hypothesize that this variation is driven by synonymous codon usage bias which indicates the preference for certain synonymous codons (coding for the same amino acid) to be used more frequently than others due to factors like gene expression, tRNA availability, or evolutionary pressures [64; 53]. We hypothesize that datasets with more pronounced synonymous codon usage bias benefit more from learning with

Table 2: Performance comparison of HELM vs. XE models. HELM models encoding mRNA hierarchy outperform non-hierarchical XE models across all downstream tasks and datasets, highlighting the importance of hierarchy as a strong biological prior. Bold indicates the best performing model.

Model	Ab1	Ab2	MLOS	iCodon	Tc-Riboswitches	mRFP
Transformer XE (MLM)	0.748	0.599	0.653	0.503	0.569	0.753
Transformer HELM (MLM)	**0.767**	**0.609**	**0.701**	**0.525**	**0.626**	**0.822**
Transformer XE (CLM)	0.752	0.597	**0.611**	0.498	0.531	0.815
Transformer HELM (CLM)	**0.760**	**0.614**	0.592	**0.529**	**0.619**	**0.849**

hierarchy as HELM treats amino acids as parent nodes and their synonymous codons as child nodes, enabling the model to learn codon usage bias by capturing preferences at both the amino acid and codon levels.

To evaluate synonymous codon usage bias for any dataset, we first calculate the distribution of synonymous codons for each amino acid and then compute the entropy of codon usage for the five most frequent amino acids. Lower entropy indicates less uniform codon distribution hence the preference towards specific codons or codon bias. Appendix Fig. 5 and Appendix Table 5 demonstrate that MLOS, Tc-Riboswitches, and mRFP datasets have lower entropy, suggesting stronger codon bias, which aligns with the larger improvements seen in these datasets using HELM models. These findings suggest that HELM's hierarchical approach is particularly beneficial for datasets with pronounced codon usage biases, highlighting its potential for improved mRNA modeling in biologically diverse contexts.

4.3 HELM improves generative mRNA sequence design

Our HELM-CLM models are inherently generative and can be used for mRNA sequence generation. We evaluate their ability to produce biologically relevant and diverse mRNA sequences through two experiments: Frechet Biological Distance and functional properties preservation.

Evaluating sample quality of generated mRNA We adopt the Frechet Biological Distance (FBD) metric [66] to assess the quality of generated sequences. FBD is a standard generative evaluation metric which quantifies the similarity between generated and real sample distributions, and is used for biological sequences. Our experiment uses XE and HELM CLM models to generate 2000 mRNA sequences encoding antibody D and J regions, conditioned on signal peptide and V regions from the OAS hold-out dataset. Generated and real mRNAs are translated to protein sequences, from which we extract embeddings using the ESM-2 model [39]. FBD is computed as the Wasserstein distance between Gaussian distributions fitted to these embeddings, with lower scores indicating better alignment with real data. We focus on amino acid (and hence protein) level FBD since the encoding hierarchy reflects the codon-amino acid relationship.

In the absence of any other existing generative models for mRNA, we benchmark the HELM model against the non-hierarchical baseline trained with vanilla cross-entropy (XE) loss. As an additional control, we establish a random baseline by generating sequences randomly, following the approach of [66]. The random baseline serves as a control to assess the significance of the sequence generation capabilities of both our HELM model and the XE baseline.

Results in Fig. 2 demonstrate that both the HELM model and the XE baseline significantly outperform the random baseline, indicating that both models generate mRNA sequences that are meaningfully aligned with real data distributions. Unsurprisingly, the generative performance varies with generation temperature, with higher temperatures leading to greater diversity but worse FBD scores. Notably, HELM consistently achieves better FBD scores than XE across all temperatures, suggesting it produces sequences more representative of real mRNA data while maintaining generation diversity. These results demonstrate HELM's effectiveness in both predictive and generative tasks, highlighting its potential for improved mRNA sequence design.

Evaluating functional properties of generated mRNA We assess the models' ability to generate sequences that maintain functional properties across six datasets from our downstream predictive tasks

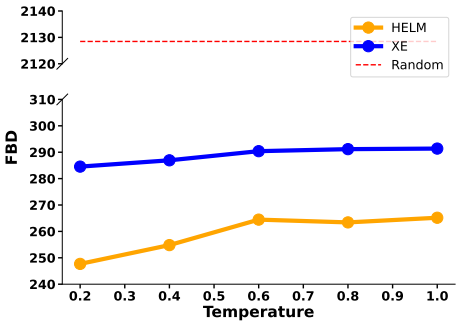

Figure 2: FBD comparison of generative HELM, XE, and random models across varying temperature. Higher temperatures increase diversity but worsen FBD scores. HELM consistently beats XE, suggesting better alignment with real mRNA data while maintaining diversity.

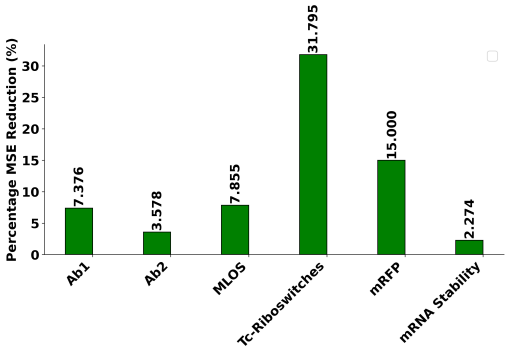

Figure 3: Percentage MSE reduction: HELM vs XE models. MSE compares the predicted properties of generated sequences to the predicted properties of original sequences. HELM consistently outperforms non-hierarchical models, indicating better retention of key mRNA properties in generated sequences.

(Sec. 4.1.2). For each dataset, we select up to 2000 sequences, condition the models on the first third of each sequence, and task them with generating the remaining two-thirds. To evaluate the retention of functional properties in generated sequences, we employ pre-trained property prediction models to estimate task-specific labels, following the protocol established in [66; 3]. We use Mean Squared Error (MSE) between predicted labels of generated sequences and true labels of corresponding real sequences as our performance metric, with lower MSE indicating better preservation of functional properties.

Fig.3 illustrates the relative improvement in MSE for HELM models compared to non-hierarchical XE models across all datasets. HELM consistently outperforms XE, with improvements ranging from 2% to 31%. This demonstrates that HELM's hierarchical encoding better preserves functional properties across diverse tasks and datasets. Notably, we observe greater improvements for MLOS, Tc-Riboswitches, and mRFP datasets, which aligns with the stronger synonymous codon usage bias observed in these datasets (as discussed in Sec.4.2). These results further underscore HELM's effectiveness in capturing and preserving the biological nuances of mRNA sequences during generation.

5 Conclusions

In this work, we highlight the importance of biological hierarchy in mRNA language modeling and show that standard natural language modeling approaches fall short in capturing the biological structure of mRNA sequences. We introduce HELM, a method to incorporate mRNA's hierarchical structure into bio-language model pre-training. By modulating loss based on codon hierarchy, HELM aligns the learning process with mRNA's inherent biological structure. In addition, we provide a consistent comparison of various tokenization methods, pre-training strategies, and model architectures to guide future mRNA LM development. HELM consistently outperforms non-hierarchical models by an average of 8% across six diverse downstream property prediction tasks with the biggest improvement observed on datasets with pronounced synonymous codon bias. Moreover, HELM models demonstrate significantly better generative capabilities than the non-hierarchical generative LMs enabling generation of more diverse and plausible mRNA sequences.

One limitation of our work is learning hierarchical relationships in Euclidean space, which may limit the model's ability to fully leverage the hierarchical prior. In contrast, hyperbolic spaces are naturally better suited for capturing hierarchical relationships, as they can effectively model the exponential growth of tree-like structures. This can be explored in future work to improve hierarchical learning and further enhance mRNA sequence modeling.

References

[1] A. Anonymous. Bridging biomolecular modalities for knowledge transfer in bio-language models. *Under review*, 2024.

[2] Mina Ghadimi Atigh, Julian Schoep, Erman Acar, Nanne Van Noord, and Pascal Mettes. Hyperbolic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4453–4462, 2022.

[3] Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pages 1276–1301. PMLR, 2023.

[4] Amos Bairoch, Rolf Apweiler, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. The universal protein resource (uniprot). *Nucleic acids research*, 33(suppl_1):D154–D159, 2005.

[5] Nils Aall Barricelli. On the origin and evolution of the genetic code i. wobbling and its potential significance. *Journal of Theoretical Biology*, 67(1):85–109, 1977.

[6] Björn Barz and Joachim Denzler. Hierarchy-based image embeddings for semantic image retrieval. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 638–647. IEEE, 2019.

[7] Samy Bengio, Jason Weston, and David Grangier. Label embedding trees for large multi-class tasks. *Advances in neural information processing systems*, 23, 2010.

[8] Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A Lord. Making better mistakes: Leveraging class hierarchies with deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12506–12515, 2020.

[9] Nicholas Boyd, Brandon M Anderson, Brent Townshend, Ryan Chow, Connor J Stephens, Ramya Rangan, Matias Kaplan, Meredith Corley, Akshay Tambe, Yuzu Ido, et al. Atom-1: A foundation model for rna structure and function built on chemical mapping data. *bioRxiv*, pages 2023–12, 2023.

[10] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[11] Florian Buhr, Sujata Jha, Michael Thommen, Joerg Mittelstaet, Felicitas Kutz, Harald Schwalbe, Marina V Rodnina, and Anton A Komar. Synonymous codons direct cotranslational folding toward different protein conformations. *Molecular cell*, 61(3):341–351, 2016.

[12] NJ Caplen. Gene therapy progress and prospects. downregulating gene expression: the impact of rna interference. *Gene therapy*, 11(16):1241–1248, 2004.

[13] Bo Chen, Xingyi Cheng, Pan Li, Yangli-ao Geng, Jing Gong, Shen Li, Zhilei Bei, Xu Tan, Boyan Wang, Xin Zeng, et al. xtrimopglm: unified 100b-scale pre-trained transformer for deciphering the language of protein. *arXiv preprint arXiv:2401.06199*, 2024.

[14] Jiayang Chen, Zhihang Hu, Siqi Sun, Qingxiong Tan, Yixuan Wang, Qinze Yu, Licheng Zong, Liang Hong, Jin Xiao, Tao Shen, et al. Interpretable rna foundation model from unannotated data for highly accurate rna structure and function predictions. *arXiv preprint arXiv:2204.00300*, 2022.

[15] Ken Chen, Yue Zhou, Maolin Ding, Yu Wang, Zhixiang Ren, and Yuedong Yang. Self-supervised learning on millions of pre-mrna sequences improves sequence-based rna splicing prediction. *bioRxiv*, pages 2023–01, 2023.

[16] Xingyi Cheng, Bo Chen, Pan Li, Jing Gong, Jie Tang, and Le Song. Training compute-optimal protein language models. *bioRxiv*, pages 2024–06, 2024.

[17] Yanyi Chu, Dan Yu, Yupeng Li, Kaixuan Huang, Yue Shen, Le Cong, Jason Zhang, and Mengdi Wang. A 5' utr language model for decoding untranslated regions of mrna and function predictions. *Nature Machine Intelligence*, 6(4):449–460, 2024.

[18] Suzanne Clancy and William Brown. Translation: Dna to mrna to protein. *Nature Education*, 1 (1):101, 2008.

[19] Genome Reference Consortium et al. Genome reference consortium human build 38 (grch38). *Database (GenBank or RefSeq)*, 2013.

[20] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, et al. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *BioRxiv*, pages 2023–01, 2023.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL `https://arxiv.org/abs/1810.04805`.

[22] Michay Diez, Santiago Gerardo Medina-Muñoz, Luciana Andrea Castellano, Gabriel da Silva Pescador, Qiushuang Wu, and Ariel Alejandro Bazzini. icodon customizes gene expression based on the codon composition. *Scientific Reports*, 12(1):12126, 2022.

[23] 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.

[24] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.

[25] Joerg KH Franke, Frederic Runge, Ryan Koeksal, Rolf Backofen, and Frank Hutter. Rnaformer: A simple yet effective deep learning model for rna secondary structure prediction. *bioRxiv*, pages 2024–02, 2024.

[26] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. *Advances in neural information processing systems*, 26, 2013.

[27] Xiang-Dong Fu. Non-coding rna: a new frontier in regulatory biology. *National science review*, 1(2):190–204, 2014.

[28] Ashima Garg, Depanshu Sani, and Saket Anand. Learning hierarchy aware features for reducing mistake severity. In *European Conference on Computer Vision*, pages 252–267. Springer, 2022.

[29] Francis Gosselin and Amal Zouaq. Sorbet: A siamese network for ontology embeddings using a distance-based regression loss and bert. In *International Semantic Web Conference*, pages 561–578. Springer, 2023.

[30] Ann-Christin Groher, Sven Jager, Christopher Schneider, Florian Groher, Kay Hamacher, and Beatrix Suess. Tuning the performance of synthetic riboswitches using machine learning. *ACS synthetic biology*, 8(1):34–44, 2018.

[31] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[32] Ameya Harmalkar, Roshan Rao, Yuxuan Richard Xie, Jonas Honer, Wibke Deisting, Jonas Anlahr, Anja Hoenig, Julia Czwikla, Eva Sienz-Widmann, Doris Rau, et al. Toward generalizable prediction of antibody thermostability using machine learning on sequence and structure features. In *Mabs*, volume 15, page 2163584. Taylor & Francis, 2023.

[33] Yuan He, Zhangdie Yuan, Jiaoyan Chen, and Ian Horrocks. Language models as hierarchy encoders. *arXiv preprint arXiv:2401.11374*, 2024.

[34] Brian L Hie, Varun R Shanker, Duo Xu, Theodora UJ Bruun, Payton A Weidenbacher, Shaogeng Tang, Wesley Wu, John E Pak, and Peter S Kim. Efficient evolution of human antibodies from general protein language models. *Nature Biotechnology*, 42(2):275–283, 2024.

[35] Henry Kenlay, Frédéric A Dreyer, Aleksandr Kovaltsuk, Dom Miketa, Douglas Pires, and Charlotte M Deane. Large scale paired antibody language models. *arXiv preprint arXiv:2403.17889*, 2024.

[36] Yoon Kim. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1181. URL https://aclanthology.org/D14-1181.

[37] Jon M Laurent, Joseph D Janizek, Michael Ruzo, Michaela M Hinks, Michael J Hammerling, Siddharth Narayanan, Manvitha Ponnapati, Andrew D White, and Samuel G Rodriques. Lab-bench: Measuring capabilities of language models for biology research. *arXiv preprint arXiv:2407.10362*, 2024.

[38] Sizhen Li, Saeed Moayedpour, Ruijiang Li, Michael Bailey, Saleh Riahi, Lorenzo Kogler-Anele, Milad Miladi, Jacob Miner, Dinghai Zheng, Jun Wang, et al. Codonbert: Large language models for mrna design and optimization. *bioRxiv*, pages 2023–09, 2023.

[39] 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.

[40] Chuang Liu, Qiangqiang Shi, Xiangang Huang, Seyoung Koo, Na Kong, and Wei Tao. mrna-based cancer therapeutics. *Nature Reviews Cancer*, 23(8):526–543, 2023.

[41] Hao Liu, Yehoshua Perl, and James Geller. Concept placement using bert trained by transforming and summarizing biomedical ontology structure. *Journal of Biomedical Informatics*, 112: 103607, 2020.

[42] Shaoteng Liu, Jingjing Chen, Liangming Pan, Chong-Wah Ngo, Tat-Seng Chua, and Yu-Gang Jiang. Hyperbolic visual embedding learning for zero-shot recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9273–9281, 2020.

[43] Yi Liu, Qian Yang, and Fangzhou Zhao. Synonymous but not silent: the codon usage code for gene expression and protein folding. *Annual review of biochemistry*, 90(1):375–401, 2021.

[44] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020. URL https://openreview.net/forum?id=SyxS0T4tvS.

[45] Céline Marquet, Michael Heinzinger, Tobias Olenyi, Christian Dallago, Kyra Erckert, Michael Bernhofer, Dmitrii Nechaev, and Burkhard Rost. Embeddings from protein language models predict conservation and variant effects. *Human genetics*, 141(10):1629–1647, 2022.

[46] Artem Moskalev, Mangal Prakash, Rui Liao, and Tommaso Mansi. Se(3)-hyena operator for scalable equivariant learning, 2024. URL https://arxiv.org/abs/2407.01049.

[47] Eric Nguyen, Michael Poli, Matthew G Durrant, Armin W Thomas, Brian Kang, Jeremy Sullivan, Madelena Y Ng, Ashley Lewis, Aman Patel, Aaron Lou, et al. Sequence modeling and design from molecular to genome scale with evo. *BioRxiv*, pages 2024–02, 2024.

[48] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36, 2024.

[49] Thijs Nieuwkoop, Barbara R Terlouw, Katherine G Stevens, Richard A Scheltema, Dick De Ridder, John Van der Oost, and Nico J Claassens. Revealing determinants of translation efficiency via whole-gene codon randomization and machine learning. *Nucleic acids research*, 51(5):2363–2376, 2023.

[50] Tobias H Olsen, Fergus Boyles, and Charlotte M Deane. Observed antibody space: A diverse database of cleaned, annotated, and translated unpaired and paired antibody sequences. *Protein Science*, 31(1):141–146, 2022.

[51] Carlos Outeiral and Charlotte M Deane. Codon language embeddings provide strong signals for use in protein engineering. *Nature Machine Intelligence*, 6(2):170–179, 2024.

[52] Norbert Pardi, Michael J Hogan, Frederick W Porter, and Drew Weissman. mrna vaccines—a new era in vaccinology. *Nature reviews Drug discovery*, 17(4):261–279, 2018.

[53] Sujatha Thankeswaran Parvathy, Varatharajalu Udayasuriyan, and Vijaipal Bhadana. Codon usage bias. *Molecular biology reports*, 49(1):539–565, 2022.

[54] Zhangzhi Peng, Benjamin Schussheim, and Pranam Chatterjee. Ptm-mamba: A ptm-aware protein language model with bidirectional gated mamba blocks. *bioRxiv*, 2024.

[55] Rafael Josip Penić, Tin Vlašić, Roland G Huber, Yue Wan, and Mile Šikić. Rinalmo: General-purpose rna language models can generalize well on structure prediction tasks. *arXiv preprint arXiv:2403.00043*, 2024.

[56] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.

[57] A Radford. Improving language understanding by generative pre-training. 2018.

[58] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[59] Jeffrey A Ruffolo, Jeffrey J Gray, and Jeremias Sulam. Deciphering antibody affinity maturation with language models and weakly supervised learning. *arXiv preprint arXiv:2112.07782*, 2021.

[60] Ugur Sahin, Katalin Karikó, and Özlem Türeci. mrna-based therapeutics—developing a new class of drugs. *Nature reviews Drug discovery*, 13(10):759–780, 2014.

[61] Conrad L Schoch, Stacy Ciufo, Mikhail Domrachev, Carol L Hotton, Sivakumar Kannan, Rogneda Khovanskaya, Detlef Leipe, Richard Mcveigh, Kathleen O'Neill, Barbara Robbertse, et al. Ncbi taxonomy: a comprehensive update on curation, resources and tools. *Database*, 2020:baaa062, 2020.

[62] Jetze Schuurmans and Flavius Frasincar. Global hierarchical neural networks using hierarchical softmax. *arXiv preprint arXiv:2308.01210*, 2023.

[63] Damiano Sgarbossa, Cyril Malbranke, and Anne-Florence Bitbol. Protmamba: a homology-aware but alignment-free protein state space model. *bioRxiv*, pages 2024–05, 2024.

[64] Paul M Sharp and Wen-Hsiung Li. The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic acids research*, 15(3): 1281–1295, 1987.

[65] Richard W Shuai, Jeffrey A Ruffolo, and Jeffrey J Gray. Iglm: Infilling language modeling for antibody sequence design. *Cell Systems*, 14(11):979–989, 2023.

[66] Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.

[67] Baris E Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H Wu. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

[68] V Ashley Villar, Kaylee de Soto, and Alex Gagliano. Hierarchical cross-entropy loss for classification of astrophysical transients. *arXiv preprint arXiv:2312.02266*, 2023.

[69] Yuning Yang, Gen Li, Kuan Pang, Wuxinhao Cao, Xiangtao Li, and Zhaolei Zhang. Deciphering 3'utr mediated gene regulation using interpretable deep representation learning. *bioRxiv*, pages 2023–09, 2023.

[70] He Zhang, Liang Zhang, Ang Lin, Congcong Xu, Ziyu Li, Kaibo Liu, Boxiang Liu, Xiaopin Ma, Fanfan Zhao, Huiling Jiang, et al. Algorithm for optimized mrna design improves stability and immunogenicity. *Nature*, 621(7978):396–403, 2023.

[71] Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. *arXiv preprint arXiv:2306.15006*, 2023.

A Appendix

A.1 Codon Hierarchy

Figure 4 shows the formalized codon hierarchy used in Helm pre-training.The root node R of this tree represents a node corresponding to all codons. This root node has two immediate child nodes: coding codons C_{coding} and non-coding codons $C_{\text{non-coding}}$. The non-coding codons $C_{\text{non-coding}}$ further branch into two child nodes: the start codon node C_{start} and the stop codon node C_{stop}. The start codon node C_{start} has a single leaf node corresponding to the codon AUG. The stop codon node C_{stop} has three leaf nodes corresponding to the codons UAA, UAG, and UGA. On the other side of the hierarchy, the coding codons C_{coding} are organized into multiple child nodes, each corresponding to a specific amino acid A_j. Each amino acid node A_j has a number of child nodes, each corresponding to a synonymous codon that encodes for A_j. Let $\text{Codons}(A_j) = \{C_{i_1}, C_{i_2}, \ldots, C_{i_m}\}$ denote the set of all synonymous codons for the amino acid A_j. These synonymous codons preserve the same amino acid, allowing for degeneracy in the genetic code. The complete hierarchical structure is thus represented by the tree $H = (V, E)$, where V includes the root node, all internal nodes (corresponding to coding and non-coding codons, start/stop codons, and amino acids), and all leaf nodes (corresponding to individual codons). The edges E define the hierarchical relationships between these nodes.

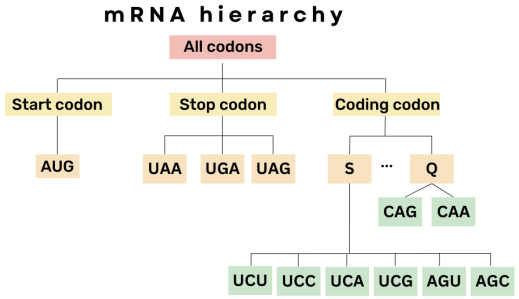

Figure 4: Formalized tree of the codon hierarchy used for pre-training HELM models

A.2 Pre-training details

We experimented with GPT-2, Mamba, and Hyena architectures pre-trained on a curated dataset. GPT-2 was trained with both MLM and CLM objective, while Mamba and Hyena were trained exclusively with CLM, given that Mamba is specifically designed for Causal Language Modeling, and recent findings show that CLM is superior to MLM for representation learning [48]. The models are pre-trained with either vanilla cross-entropy loss (vanilla XE) or hierarchical cross-entropy loss (HXE), also referred to as the HELM model (See Sec. 4.1 in main text).

With codon tokenization, the maximum context length for pre-training is 444 tokens, equivalent to the longest sequence of 1332 nucleotides in the dataset. However, the models are designed to support sequences up to 2048 tokens, thanks to a positional embedding layer that scales to 2048 positional embeddings, offering flexibility for future use with longer sequences without architectural changes. All models were pre-trained for 40 epochs on 8 Nvidia A100 GPUs. For downstream tasks, a 1 million-parameter TextCNN[36] head was used for fine-tuning, following common probing techniques[45; 13; 51; 32].

All models are designed to have approximately 50 million parameters. GPT-2 used 10 layers with a hidden size of 640 and an intermediate size of 2560. For position embedding, we follow the strategy used in GPT-2. We employ an embedding layer and add the embedded positional embeddings to the token embeddings. This approach is consistent across all models in our study.

Although GPT-2 supports CLM by design, we use attention without causal masking for creating MLM variant of it. We employ the AdamW optimizer with a learning rate scheduler using linear warmup and cosine decay. Initially, we use a learning rate of 1e-3 for all models. For hierarchical HELM models trained with HXE loss, we reduced the learning rate to 1e-4 for all models to accommodate the smaller scale of the loss compared to XE. We experimented with three different α values for HXE: 0.2, 0.4, and 0.6 (See Sec. 3 in main text for details) with $\alpha = 0.2$ performing the best overall.

All models are trained using 8 NVIDIA A100 GPUs, each with 80GB of GPU memory. We utilized mixed-precision training (AMP) to improve computational efficiency. The distributed training setup allowed us to effectively use a batch size of 1024 (128 per GPU across 8 GPUs).

We initially experimented with longer training durations but found no significant improvement beyond 40 epochs. Therefore, all models are trained for 40 epochs to ensure consistent comparison and efficient use of computational resources. The hyperparameters used for training are summarized in the following table:

Table 3: Hyperparameters for trained LM Models

Hyperparameter	**GPT-2**	**Mamba**	**Hyena**
Number of layers	10	40	7
Hidden size	640	256	768
Intermediate size	2560	1024	3072
Batch size	1024	1024	1024
Learning rate (XE)	1e-3	1e-3	1e-4
Learning rate (HXE)	1e-4	1e-4	1e-4
Minimum learning rate	1e-5	1e-5	1e-6
Weight decay	0.1	0.1	0.1
Number of epochs	40	40	40
Vocabulary size	70	70	70

A.3 Downstream task finetuning details

To evaluate the effectiveness of our pre-trained LMs on downstream tasks, we employ probing approach [45; 13; 51; 32] using TextCNN [36]. This allows us to assess the quality and transferability of the learned representations across various downstream predictive tasks.

In our implementation of TextCNN, we set the embedding size to 640 to match the hidden size of our models, ensuring consistency across all probing experiments. The convolutional layer size is fixed at 100, which we find to provide a good balance between model capacity and computational efficiency. To focus solely on evaluating the quality of the learned representations, we freeze the weights of the pre-trained LMs, using them as fixed feature extractors to generate embeddings for the probing experiments. We conduct a comprehensive hyperparameter search to optimize the performance of the TextCNN probe on each downstream task. Our search strategy employs a grid search methodology, exploring a range of learning rates and batch sizes. Specifically, we investigate learning rates of 3e-4, 1e-4, and 1e-5, which span a reasonable range for finetuning neural networks. For batch sizes, we examine values of 8, 16, 32, and 64, allowing us to balance between update frequency and the stability of gradient estimates. The combination of three learning rates and four batch sizes results in twelve distinct configurations for each downstream task. We train the TextCNN probe using each of these configurations and evaluate their performance on a hold-out validation set to choose the best hyperparameters. For our final results, we report the performance as measured on a separate test set. This approach ensures that we are comparing the most optimized versions of each probing model, providing a fair assessment of the underlying representations learned by our pre-trained models. For all datasets, whether they come with predefined splits or not, we maintained a consistent data partitioning strategy. We divide the data into training, validation, and test sets with ratios of 0.75, 0.15, and 0.15, respectively. This split ensures a substantial portion of data for training while allocating sufficient samples for validation and testing. In cases where the datasets for downstream tasks does not come with predefined train-validation-test splits, we implement a robust splitting strategy. We randomly divide the data into training, validation, and test sets, maintaining consistent proportions across all tasks. To account for potential variability introduced by this random splitting, we repeat the entire process using three different random seeds. The reported results for these tasks represent the average performance across these three splits, providing a more reliable estimate of the model's generalization capabilities. This comprehensive finetuning and evaluation process allows us to systematically and fairly assess the transferability and quality of the representations learned by our pre-trained models.

A.4 Tokenization Strategy Experiments

We conducted experiments to compare the performance of different tokenization strategies on various mRNA datasets, including nucleotide-level, 6-mer, and codon-level tokenization. The experiments were performed using a 50M parameter GPT-2 model with a Masked Language Modeling (MLM) objective. Other experimental setups, such as data processing, training pipelines, and evaluation metrics, were kept consistent across all models to ensure a fair comparison of tokenization strategies.

A.4.1 Tokenization Strategies

- **Nucleotide-level Tokenization:** This strategy treats each nucleotide (A, C, G, U) as a single token, resulting in a vocabulary size of 4.
- **6-mer Tokenization:** In this approach, sequences are split into overlapping or non-overlapping chunks of six nucleotides, generating a more complex vocabulary.
- **Codon-level Tokenization:** Codon-level tokenization represents sequences as triplets of nucleotides, resulting in a vocabulary of 64 codons, corresponding directly to amino acids.

A.4.2 Results

Table 4 presents a comparison of the performance of different tokenization strategies across several mRNA datasets in terms of Spearman rank correlation (higher the better). Codon-level tokenization generally performs best across most datasets, except for mRFP, where 6-mer tokenization outperformed codon-level tokenization.

Table 4: Comparison of tokenization strategies across different mRNA datasets. Codon-level tokenization generally performs best, with 6-mer outperforming on mRFP.

Dataset	Nucleotide-level	Codon-level	6-mer
Ab1	0.699	**0.747**	0.733
Ab2	0.569	**0.599**	0.582
MLOS	0.528	**0.654**	0.625
Tc-Riboswitches	0.533	**0.570**	0.562
mRFP	0.820	0.754	**0.857**

The results show that codon-level tokenization captures the biological significance of mRNA sequences and is more suitable for most tasks. However, on MRFP and iCodon datasets, 6-mer tokenization yielded better performance.. The 50M parameter GPT-2 model with the MLM objective was used in all experiments, and all other experimental setups remained the same to facilitate fair comparison. For further experimental details and setup, please refer to this appendix.

A.5 Entropy Details

Table 5 reveals that MLOS, Tc-Riboswitches, and mRFP datasets exhibit lower entropy values, indicating stronger codon usage bias compared to Ab1, Ab2, and iCodon datasets which have higher entropy and more uniform codon distributions. This observation aligns with our hypothesis, as HELM models achieve greater improvements in datasets with skewed codon usage, suggesting that hierarchical learning better captures and leverages these codon biases for enhanced accuracy.

A.6 Impact of the choice of α for HXE loss

In this section, we present a detailed analysis of the impact of the hyperparameter α on the performance of models trained using the Hierarchical Cross-Entropy (HXE) loss. The α parameter in the HXE loss plays a crucial role in weighting the importance of hierarchical relationships during model training. Specifically, α controls how much weight is given to mistakes that violate the hierarchical structure of mRNA sequences, where codons are organized based on their synonymous relationships.

As explained in Sec. 3, the HXE loss leverages the hierarchical nature of mRNA sequences by applying differential penalties to the prediction errors based on their position within the hierarchy.

Table 5: Average entropy values of synonymous codon distributions for the top five most frequent amino acids in each dataset. Datasets with lower entropy values, such as MLOS, Tc-Riboswitches, mRFP, and iCodon, exhibit stronger synonymous codon usage bias, correlating with the enhanced performance of hierarchical HELM models on these datasets (see Fig. 5 and Table 2).

Dataset	Average Entropy (top-5)
Ab1	1.81
Ab2	1.81
MLOS	1.62
iCodon	1.77
Tc-Riboswitches	1.63
mRFP	1.20

The weighting function $\lambda(C)$ is $\lambda(C) = \exp(-\alpha h(C))$, where $h(C)$ is the height of the node C and $\alpha > 0$, determines the extent to which errors are penalized. Higher values of α increase the penalty for mistakes higher up in the hierarchy (e.g., confusing codons that code for different amino acids), while lower values of α result in a more uniform penalty across the hierarchy.

Table 6: Ablation study on HXE loss α across downstream tasks

Hyperparameter	**Ab1**	**Ab2**	**iCodon**	**mRFP**	**Tc-Riboswitches**
Transformer HELM (MLM)					
$\alpha = 0.2$	0.767	0.609	0.525	0.822	0.626
$\alpha = 0.4$	0.767	0.608	0.511	0.831	0.653
$\alpha = 0.6$	0.762	0.599	0.508	0.796	0.582
Transformer HELM (CLM)					
$\alpha = 0.2$	0.760	0.614	0.529	0.849	0.619
$\alpha = 0.4$	0.762	0.608	0.527	0.841	0.551
$\alpha = 0.6$	0.752	0.592	0.526	0.798	0.580

To assess the impact of α we evaluated the performance of the transformer models trained with the HXE loss using three different values of α: 0.2, 0.4, and 0.6. These values were chosen to explore a range of hierarchical weighting, from mild to strong penalties for hierarchical errors. Table 6 summarizes the performance of the models on each downstream task. We observe the following trends:

- α =0.2: At this lower value of α, the model applies relatively mild penalties for hierarchical mistakes. The performance across most tasks is stable, indicating that even a modest incorporation of hierarchical weighting can enhance the model's performance by aligning with the biological structure of mRNA sequences.

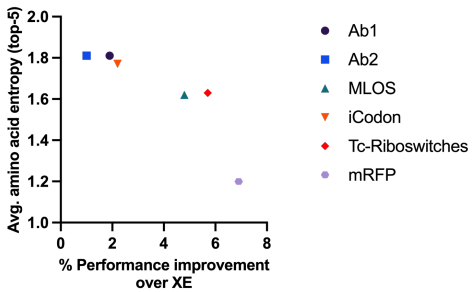

Figure 5: Average entropy of synonymous codon distributions correlates with HELM model performance improvement. Lower entropy, indicating stronger codon bias, is observed in datasets like MLOS, Tc-Riboswitches, and mRFP, where HELM outperforms XE models.

- α =0.4: Increasing α to 0.4 generally results in improved performance for MLM, particularly for the mRFP and Tc-Riboswitches tasks. This suggests that at this level, the model benefits from a stronger alignment with the hierarchical structure, which is critical for these tasks. However, the performance on some tasks (e.g., Ab1 and Ab2) remains stable, indicating that the benefit of increased α might be task-dependent.
- α =0.6: When α is further increased to 0.6, we observe a decline in performance for several tasks, including mRFP and Tc-Riboswitches. This suggests that an overly strong hierarchical penalty can restrict the model's flexibility, leading to overfitting to the hierarchical relationships and a loss of generalization across diverse tasks.

Thus, our experiments show that a moderate value of α (0.2 to 0.4) generally yields the best results across the diverse set of tasks. This value strikes a balance between enforcing hierarchical structure and maintaining model flexibility.

A.7 Scalability Experiments

In this section, we provide the details of our experiments with three GPT-2-based models: 20 million (20M), 50 million (50M), and 100 million (100M) parameters. All models were trained using the HELM framework with a Causal Language Modeling (CLM) objective. The performance was evaluated on downstream mRNA property prediction tasks.

A.7.1 Model Configurations

- **20M Model:** This model consists of 12 transformer layers, each with hidden size of 384 and intermediate size of 1536.
- **50M Model:** This model consists of 10 transformer layers, each with a hidden size of 640 and intermediate size of 2560.
- **100M Model:** This model consists of 14 transformer layers, each with a hidden size of 768 and intermediate size of 3072.

All models were trained on the same mRNA dataset using the HELM-modified CLM objective, and we maintained a consistent training pipeline across all experiments.

A.7.2 Results

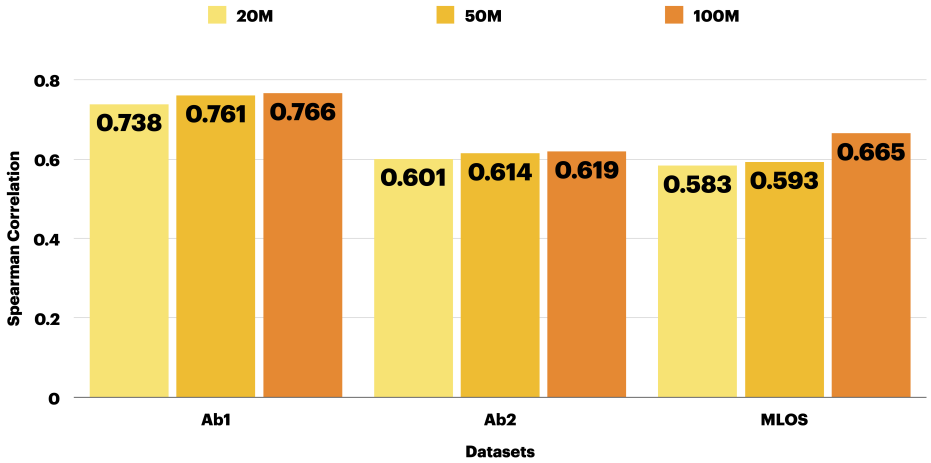

Figure 6: Performance comparison of GPT-2 models with 20M, 50M, and 100M parameters trained using the HELM framework.

This analysis illustrates the scalability of HELM and highlights the balance between model size and performance efficiency (see Fig. 6. The diminishing returns beyond 50M parameters suggest that optimal model sizes for mRNA tasks may lie within the 50M-100M parameter range.

The training duration for each model configuration is illustrated in Figure 7.

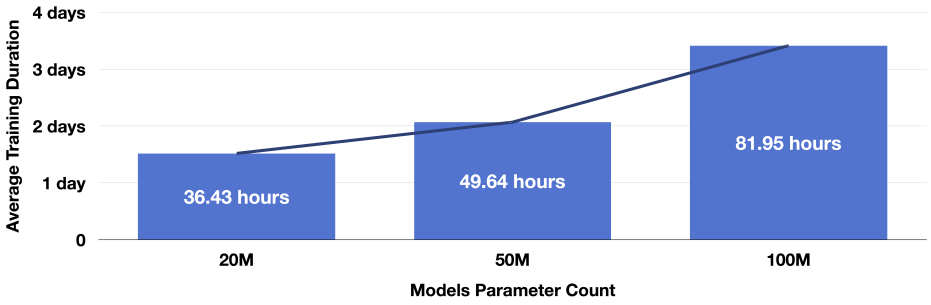

Figure 7: Average Pre-training Time Across Models with Varying Parameter Counts

A.8 Pre-training data curation and statistics

As discussed in Sec. 4 in main text, we ensure that the mRNA data used for pre-training our models in this study is of the highest quality, with careful attention to preserving the biological diversity and representativeness of the sequences. We conduct statistical analyses to confirm that the curated dataset maintains a balanced representation of gene types, comparable to the original OAS human dataset and did not introduce any biases in the gene representation.

First, we filtered sequences based on the *ANARCI status* annotation in the dataset, excluding those with unusual residues, indels, truncations, or a lack of conserved cysteines, which are often problematic. We further refined our selection by focusing only on sequences with a V and J identity greater than 0.7, ensuring a high degree of similarity to known reference sequences. Additionally, we retained only those sequences labeled as *productive* and *complete vdj*, indicating they are fully functional and contain complete variable, diversity, and joining regions. For the purpose of this study, we restricted the dataset to human sequences only. Recognizing that the paired sequences in the database are significantly fewer in number compared to unpaired ones, we unpaired these sequences to increase the dataset's size and diversity. We then performed sequence similarity analysis separately on paired and unpaired sequences to eliminate redundancy with a threshold of 0.5. To maintain a balanced representation, we manually adjusted the number of heavy and light chains. Finally, we compared the gene frequencies of heavy and light chains in our curated dataset against the original human OAS dataset to ensure that our curation process did not introduce artificial overrepresentation or underrepresentation of any gene types. This process yields 15.3 million sequences with 7.7 million heavy and 7.6 million light chains each.

The following figures present the distribution of gene types for heavy and light chains in both the original and curated datasets, as well as a breakdown of heavy and light chain numbers, including Kappa (K) and Lambda (L) isotypes.

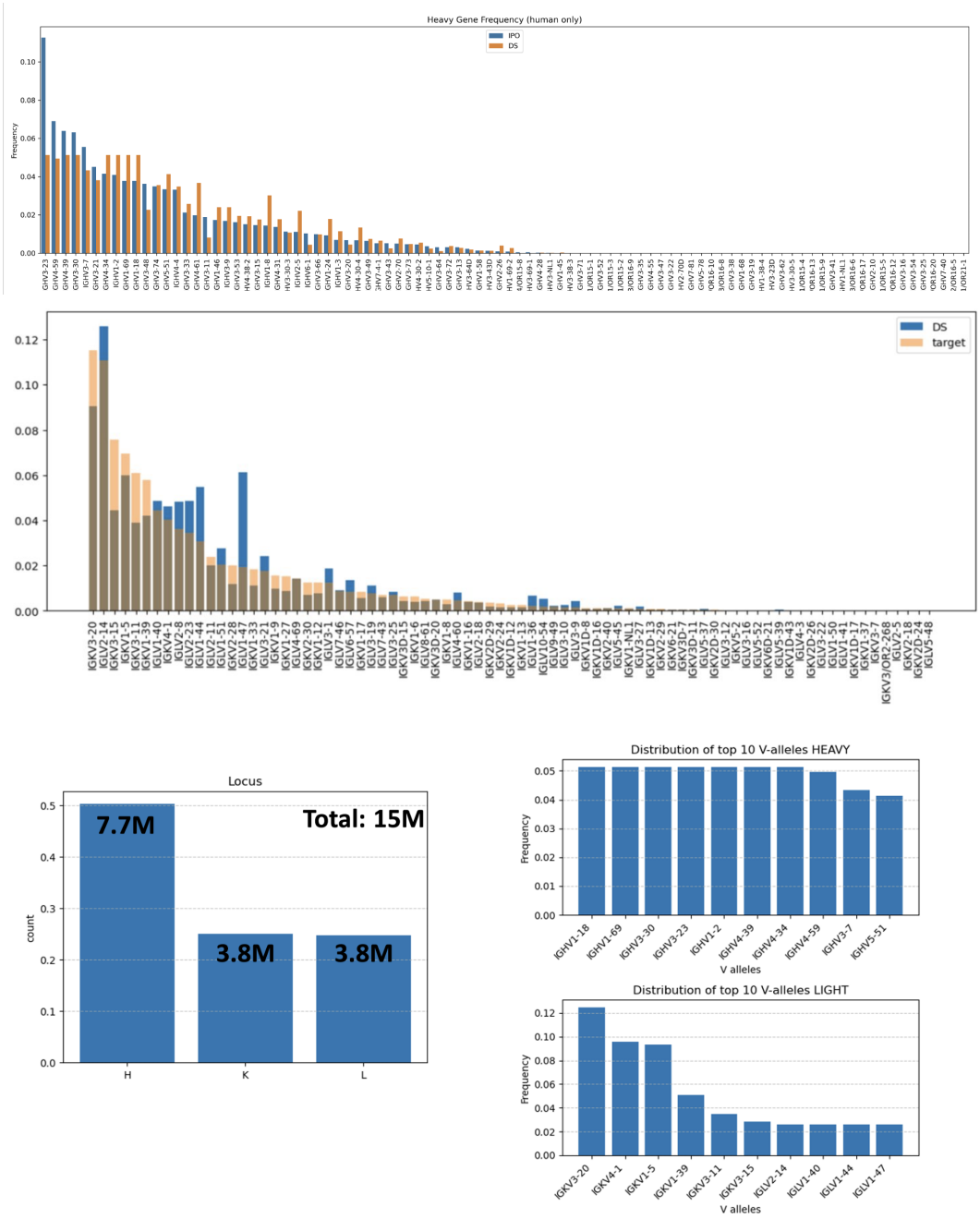

Figure 8: Comparison of gene distributions between the original OAS human data and the curated dataset. (a) shows the gene distribution for heavy chains, and (b) shows the gene distribution for light chains. (c) illustrates the balanced numbers of heavy and light chains, including the distribution of Kappa and Lambda isotypes, along with V-allele distribution.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction section carefully convey the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Limitations of the work was discussed in the conclusion section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: There is no theoretical analysis in this work.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification:

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental settings and details are availble in the appendix of this work.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All compute resources are mentioned in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: No possible violation exists in this study.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: [NA]

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.