Large Language Models — the Future of Fundamental Physics?

Caroline Heneka[1], Florian Nieser[2,3], Ayodele Ore[1], Tilman Plehn[1,3], and Daniel Schiller[1]

1 Institut für Theoretische Physik, Universität Heidelberg, Germany
2 Heidelberg Center for Digital Humanities (HCDH), Universität Heidelberg, Germany
3 Interdisciplinary Center for Scientific Computing (IWR), Universität Heidelberg, Germany

July 29, 2025

Abstract

For many fundamental physics applications, transformers, as the state of the art in learning complex correlations, benefit from pretraining on quasi-out-of-domain data. The obvious question is whether we can exploit Large Language Models, requiring proper out-of-domain transfer learning. We show how the Qwen2.5 LLM can be used to analyze and generate SKA data, specifically 3D maps of the cosmological large-scale structure for a large part of the observable Universe. We combine the LLM with connector networks and show, for cosmological parameter regression and lightcone generation, that this Lightcone LLM (L3M) with Qwen2.5 weights outperforms standard initialization and compares favorably with dedicated networks of matching size.

Contents

1 Introduction **3**

2 Large Language Models **4**
2.1 Tokenization 4
2.2 Autoregressive pretraining 4
2.3 Network architecture 5
2.4 Finetuning 8
2.5 Efficient training 9

3 Lightcone Large Language Model (L3M) **9**
3.1 Architecture 9
3.2 21cm lightcone data 11

4 Parameter regression with frozen backbone **13**
4.1 Data and connector architecture 13
4.2 Results 16

5 Generation with finetuned backbone **17**
5.1 Data and connector architecture 17
5.2 Results 21

6 Conclusion **22**

A Specifics about Qwen2.5 **25**

B Generation dataset 25

References 27

1 Introduction

The complexity and volume of experimental data in fundamental physics is increasing dramatically right now, while our lives are simultaneously transformed by modern machine learning (ML). Cutting-edge ML-methods allow us to make optimal use of this data, combining meaningful complexity, huge data volumes, fast precision simulations, and simulation-based inference into the scientific methodology of the coming decades [1–4]. Here, the fundamental paradigm shift is that *complexity is a feature, not a problem.*

To extract complex correlations, modern network architectures like transformers are extremely powerful. This is true for data acquisition, data reconstruction, first-principle simulation, and optimal inference. Initially, using existing architectures from the ML literature proved a promising path to scientific progress. Transformers with their unprecedented expressivity have brought us to the point where performance can only be improved sustainably by working toward physics-specific requirements and by using domain-specific knowledge. The prime example for physics domain knowledge are (slightly broken) symmetries, with networks built to guarantee equivariance [5–23].

For complex data representations, symmetries can be challenging to encode explicitly. An alternative approach, learning structures and symmetries inspired by foundation models, has recently gained interest in astrophysics [24–29] and particle physics [30–39]. After imposing minimal bias on the network architecture, the goal is to learn appropriate and ideally symmetry-aware data representations from generic, large datasets. The key premise is that out-of-domain data can be leveraged to scaffold a base representation for downstream finetuning on specialized data. Transformers have been shown to be the best-suited architecture for, both, representation learning (encoding) and generation (decoding). Pretraining on quasi-out-of-domain data allows for extremely data-efficient finetuning, even across network tasks.

Thinking this pretraining strategy to the end, there remains a gap between physics research and industry in terms of the network and dataset sizes. Even in particle physics applications with cheap and precise simulations, the largest open datasets used for pretraining contain around 100M jets [40, 41]. For SKA studies we are typically limited to tens of thousands simulated realizations of tomographic sky maps with semi-numerical codes. For fully hydrodynamical simulators we are even more limited in terms of open datasets [42, 43]. Large Language Models (LLMs) are comprised of over 100B parameters and trained on trillions of words. An obvious question is whether these networks can be exploited for physics [44]. Specifically, can the extreme gap in scale between LLMs and the typical physics networks compensate for the shift in the modality of the data. Unlike in existing particle physics and astrophysics studies, using a pretrained LLM implies a proper out-of-domain pretraining.

In this paper, we explore this question for the first time quantitatively and in detail. We begin by reviewing state-of-the-art LLMs for a physics audience in Sec. 2. Then, in Sec. 3, we outline how the LLM is adapted for numerical data. We apply Qwen2.5-0.5B [45–47] to simulations of the cosmological 21cm signal. We develop a Lightcone LLM (L3M), attaching two connector networks to the pretrained LLM. In Sec. 4 we target with L3M a 6-dimensional regression problem of astrophysical and cosmological parameters and compare the L3M performance for pretrained and randomized LLM backbones with two reference networks, one large and one with the same number of trainable parameters as the L3M connector networks. Especially the pretrained L3M fine-tuning is extremely data-efficient and it outperforms the small reference networks, showing that the LLM with out-of-domain pretraining indeed works. Finally, in Sec. 5 we go a step further and finetune the LLM backbone itself. Here, the randomized LLM backbone do not gain anything, but a pretrained and finetuned LLM outperform dedicated networks of matching size.

2 Large Language Models

We review the elements of state-of-the-art LLMs from a physics perspective, beginning with the data representation via tokenization in Sec. 2.1, followed by the pretraining Sec. 2.2. Then, we describe the network architecture in Sec. 2.3 and introduce finetuning methods in Sec. 2.4 and 2.5. For in-depth reviews, we recommend Refs. [48–50].

2.1 Tokenization

Tokenization is a crucial step in natural language processing. It introduces a representation of language by converting a string of characters s into a sequence of tokens,

$$s \longleftrightarrow (t_1, \ldots, t_n) \qquad t_i \in V \,. \tag{1}$$

Because the vocabulary V is a finite set, each token can be assigned a unique token-id. Tokens can be considered a generalization of characters. A concrete tokenizer defines the grammar for an LLM.

There exist many algorithms to create a vocabulary of lexical tokens, Byte Pair Encoding [51] being a wide-spread choice. It starts with a base vocabulary, which can tokenize all strings in the training data. This can be all characters or, alternatively, all bytes. The most frequent adjacent token pairs are then iteratively merged and added to the vocabulary as a new token. This stops once a specified vocabulary size is reached, typically of order 10^5. WordPiece tokenization [52, 53] also extends a base vocabulary, but instead of merging tokens by frequency, it merges them based on high mutual information between them. Once a vocabulary is created, it remains fixed and forms the latent representation of the training text. For this study, we represent physics (simulated SKA data) as non-linguistic, numeric, tokens by embedding our data with additional networks, see Sec. 3.1.

In addition, special tokens can be added or removed afterwards to indicate non-linguistic meta-information. Typically, a special token is introduced for the start, <|im_start|>, and the end, <|im_end|>, of messages, defining the chat template. It also encodes the source of a message as: (i) the system defining the broad task of the LLM, for instance a chat bot; (ii) the user whose queries prompt the LLM; and (iii) the assistant defined by the LLM's responses. The source is appended to the start token, for example as

```
<|im_start|>system
You are a wise physics AI.<|im_end|>
<|im_start|>user
What is your favorite astrophysical experiment?<|im_end|>
<|im_start|>assistant
It is the Square Kilometer Array.<|im_end|>
```

2.2 Autoregressive pretraining

A language generator encodes the probability of sequences of tokens, $p(t_1, \ldots, t_n)$ in a factorized, autoregressive form,

$$p(t_1, \ldots, t_n) = \prod_{i=1}^{n} p(t_i | t_1, \ldots, t_{i-1}) \,, \tag{2}$$

LLMs are most commonly pretrained to approximate these conditionals

$$p_\theta(t_i|t_1,\dots,t_{i-1}) \approx p(t_i|t_1,\dots,t_{i-1})\,, \tag{3}$$

for next-token prediction [54]. The LLM is trained by minimizing the log-likelihood of a dataset, leading to a cross-entropy loss

$$\mathcal{L} = -\sum_{i=2}^{N} \Big\langle \log p_\theta(t_i|t_1,\dots,t_{i-1}) \Big\rangle_{p_{\text{data}}(t_i\,|\,t_1,\dots,t_{i-1})}\,. \tag{4}$$

The prediction of t_1 is excluded, as there is no condition. Because the vocabulary is discrete, each conditional is a categorical distribution. For particle physics, autoregressive probabilities have been introduced for phase space directions [55] and for (generated) particles [56, 57].

Next-token prediction can be considered self-supervised in the sense that no explicit labeling of text in the dataset is necessary. The objective is simply to complete partial data examples. This is a difficult task in the absence of a specialized context, and extremely large datasets are required. Modern LLMs are typically pretrained on 10^{11} to 10^{14} tokens. Given that datasets of this magnitude are collected in an unsupervised manner, the data quality has to be improved through filtering or other preprocessing steps [50]. Due to the immense computational cost of pretraining an LLM, hyperparameters must be carefully chosen ahead of time [47, 58].

2.3 Network architecture

Next-token prediction requires a network architecture that matches the conditional structure of Eq.(2),

$$f_\theta : V^n \to \text{Cat}(V)^n \qquad f_\theta(t_1,\dots,t_n) = \begin{pmatrix} p_\theta(t|t_1) \\ \vdots \\ p_\theta(t|t_1,\dots,t_n) \end{pmatrix} \qquad n \in \mathbb{N}\,. \tag{5}$$

First, the network f_θ has to process sequences of varying length n. Second, it must enforce the correct 'causal' conditioning, e.g. that $p_\theta(t|t_1)$ is independent of $t_{i>1}$ etc. Both requirements are satisfied by transformers [59]. We decompose f_θ into four parts, so a sequence of tokens $(t_1,\dots,t_n)$ is processed by

1. an embedding layer, which maps each discrete token to a high-dimensional latent vector,

$$E : V \to \mathbb{R}^d \qquad x_i = E(t_i) \qquad \text{with} \qquad d \sim 10^4 - 10^5\,; \tag{6}$$

2. a backbone transformer which maps between sets of latent vectors,

$$g : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d} \qquad (x_1,\dots,x_n) \mapsto (y_1,\dots,y_n)\,; \tag{7}$$

3. an un-embedding map which translates a latent vector into unnormalized log-probabilities,

$$E^T : \mathbb{R}^d \to \mathbb{R}^{|V|} \qquad y_i \mapsto z_i\,; \tag{8}$$

4. a normalization of the final categorical probabilities,

$$\text{Softmax} : \mathbb{R}^{|V|} \to \text{Cat}(V) \qquad \text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}\,. \tag{9}$$

We can then write the network f_θ as

$$f_\theta = \text{Softmax} \circ E^T \circ g \circ E \,, \tag{10}$$

where the softmax and (un)embedding layers act element-wise across the sequence. The embedding layers can be represented as matrices, $E \in \mathbb{R}^{|V|\times d}$, $E^T \in \mathbb{R}^{d\times|V|}$. In some LLMs, including Qwen2.5-0.5B [45–47], weights are shared between E and E^T. Since the embedding layers act element-wise, the backbone g is responsible for learning correlations among token representations. A prototypical LLM backbone architecture based on Qwen2.5 is depicted in Fig. 1. More information about Qwen2.5 and its training can be found in App. A; in the following we describe key features and concepts.

Self Attention [59]. This critical building block allows the backbone to handle variable-length sequences and satisfy the causal conditioning. It defines a vector representation inspired by an orthogonal basis [60], fitting the structure of Eq.(7).

We describe Grouped Query Attention [61], used in Qwen2.5. For each input vector $(x_1,\ldots,x_n) \in \mathbb{R}^{n\times d}$, h_Q query, h_{KV} key and h_{KV} value vectors are computed via trainable affine layers,

$$\begin{aligned}
q_i^{(j_Q)} &= W_Q^{(j_Q)} x_i + b_Q^{(j_Q)} \quad \in \mathbb{R}^{d_h} && (j_Q = 1 \ldots h_Q)\,, \\
k_i^{(j_{KV})} &= W_K^{(j_{KV})} x_i + b_K^{(j_{KV})} \in \mathbb{R}^{d_h} && (j_{KV} = 1 \ldots h_{KV})\,, \\
v_i^{(j_{KV})} &= W_V^{(j_{KV})} x_i + b_V^{(j_{KV})} \in \mathbb{R}^{d_h} && (d_h = d/h_Q)\,,
\end{aligned} \tag{11}$$

implying h_Q query heads and h_{KV} key-value heads. Here, h_Q has to be a multiple of h_{KV}, so the query vectors can be divided into groups of $G = h_Q/h_{KV}$ vectors. The attention matrix is

$$A_{ij}^{(j_Q)} = \frac{q_i^{(j_Q)} \cdot k_j^{(\lfloor j_Q/G \rfloor)}}{\sqrt{d_h}} \quad \in \mathbb{R}^{n\times n}\,. \tag{12}$$

The value vectors are summed for each token, weighted by attention score according to

$$a_i^{(j_Q)} = \sum_{j=1}^{n} \text{Softmax}\,(A_i)_j \, v_j^{(j_Q)}\,. \tag{13}$$

The resulting vectors are concatenated into

$$a_i = \left(a_i^{(1)},\ldots,a_i^{(h_Q)}\right) \in \mathbb{R}^d\,. \tag{14}$$

An attention mask controls the dependence of a_i on specific tokens. Causal conditioning corresponds to

$$A \to A + M_{\text{causal}} \qquad \text{with} \qquad (M_{\text{causal}})_{ij} = \begin{cases} -\infty & j > i \\ 0 & \text{otherwise} \end{cases}\,. \tag{15}$$

Finally, the attention output undergoes a linear map with trainable weight matrix W_O,

$$x_i' = W_O a_i \in \mathbb{R}^d\,. \tag{16}$$

For $h_Q = h_{KV}$ Grouped Query Attention turns into multi-head attention [59]. During inference, each token is sampled autoregressively, and the computed key-value pairs are cached for subsequent computations. Setting $h_Q > h_{KV}$ reduces the number of cached key-value pairs, speeding up inference.

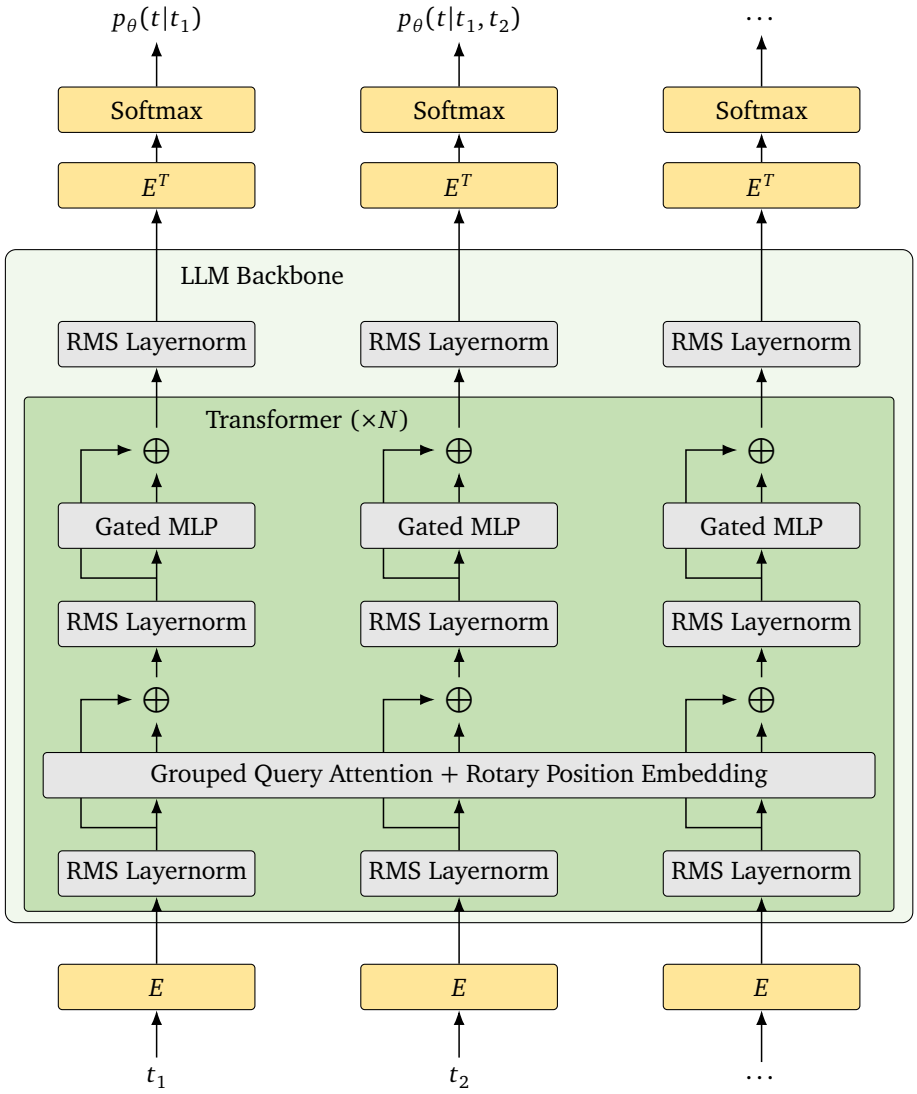

Figure 1: Qwen2.5 architecture, separating the embedding layers from the LLM backbone.

Rotary Position Embedding [62]. The updated token representation x_i' of Self Attention is manifestly invariant under permutations of the preceding token representations $(x_1, \ldots, x_{i-1})$. To add information about the relative positions between the token representations, Rotary Position Embedding is a common choice in LLMs. The scalar product in Eq.(12) is modified by inserting 2-dimensional rotations,

$$q_i \cdot_{\text{RoPE}} k_j \equiv \sum_{k=1}^{d_h/2} \begin{pmatrix} q_{i,2k} \\ q_{i,2k+1} \end{pmatrix} R((j-i)\theta_k) \begin{pmatrix} k_{j,2k} \\ k_{j,2k+1} \end{pmatrix}, \tag{17}$$

where $R((j-i)\theta_k)$ is a rotation by the angle of $(j-i)\theta_k$. The frequency θ_k depends on the dimension k, and is usually given by $\theta_k = \Theta^{-2k/d_h}$ with a base frequency Θ. These rotations tend to give more weight to the scalar product between query-key pairs when the corresponding tokens are closer to each other.

LLMs can only reliably generate tokens if the sequence length is at most as long as the maximal trained sequence length. Since the complexity of the self-attention operation scales quadratically with the sequence length, there is a maximal trainable sequence length in prac-

tice. By freezing a pretrained LLM and training interpolating frequencies [63, 64], the supported sequence length can be extended.

Attention dropout [65]. To reduce overfitting on dominant query-key pairs, attention dropout can be used. In this regularization technique, the entries of the softmax vector in Eq.(12) are randomly set to zero with probability p, which is a hyperparameter. The non-vanishing entries of the softmax vector are scaled by a factor $1/(1-p)$.

RMS Layernorm [66]. This operation normalizes a vector, $x \in \mathbb{R}^d$, with respect to its root mean square,

$$x_i' = \frac{\lambda_i \, x_i}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}} \in \mathbb{R}^d \,, \tag{18}$$

where $\lambda \in \mathbb{R}^d$ is a trainable scaling factor, which is initialized with $\lambda_i = 1$, and ϵ is a numerical cutoff. It stabilizes the training dynamics and accelerates the convergence for the deep LLMs.

Gated MLP [67, 68]. This operation realizes a non-linear map from $\mathbb{R}^d$ to itself through a larger latent space $\mathbb{R}^{d_{\text{ff}}}$, usually with $d_{\text{ff}} = 4d$. It is defined by three trainable weight matrices, $W_1 \in \mathbb{R}^{d\times d_{\text{ff}}}$ and $W_2, W_3 \in \mathbb{R}^{d_{\text{ff}}\times d}$, and a nonlinear activation function, act$(\cdot)$,

$$x' = W_1 \left(\text{act}\,(W_2 x) \odot (W_3 x)\right) \,, \quad x, x' \in \mathbb{R}^d \tag{19}$$

where $\odot$ is the element-wise multiplication. Empirically, it outperforms standard feedforward networks in LLMs.

Residual Connections [69]. To further stabilize the training dynamics for a deep network, residual connections are used for the Self-Attention and Gated MLP operations, indicated by $\oplus$ in Fig. 1. This structure reframes the learning objective for each block, encouraging it to learn a residual function with respect to its input rather than an entirely new representation.

2.4 Finetuning

After pretraining, the LLM has to be finetuned for a given task. A common approach is to create a dataset under supervision, which is much smaller than the one used for pretraining. The LLM is then trained further using the next-token objective of Eq.(4) on the curated dataset [70].

Reinforcement learning (RL) is another approach for finetuning an LLM [71] or to align the generated sequences with certain preferences [72]. A query sequence

$$\left(t_1^{(q)}, \ldots, t_n^{(q)}\right) \equiv q \tag{20}$$

is identified as a state, and the generated LLM-response

$$\left(t_1^{(r)}, \ldots, t_m^{(r)}\right) \equiv r \tag{21}$$

as the corresponding action. The conditional of the response on the query is the policy π,

$$\begin{aligned} p(r|q) &= p\left(t_1^{(r)}, \ldots, t_m^{(r)} \,|\, t_1^{(q)}, \ldots, t_n^{(q)}\right) \\ &\equiv \pi\left(t_1^{(r)}, \ldots, t_m^{(r)} \,|\, t_1^{(q)}, \ldots, t_n^{(q)}\right) = \pi(r|q) \,. \end{aligned} \tag{22}$$

During RL-based finetuning, a reward is assigned to each response,

$$\text{reward}(r|q) \in \mathbb{R} \,. \tag{23}$$

The policy is optimized to maximize the expected reward,

$$\pi_{\text{optimal}} = \arg\max_{\pi} \Big\langle \text{reward}(r|q) \Big\rangle_{\pi(r|q),\, p_{\text{data}}(q)} \,. \tag{24}$$

Prominent RL objectives are Proximal Policy Optimization [73], Direct Preference Optimization [74] and Group Relative Policy Optimization [75].

2.5 Efficient training

The computational cost of finetuning can be reduced by training only a fraction of the network weights. We describe two prominent examples, which we will use in our physics study.

Low Rank Adaptation (LoRa) [76]. Instead of training affine layers

$$x' = Wx + b \qquad \text{with} \qquad x', b \in \mathbb{R}^{d_1} \,,\; x \in \mathbb{R}^{d_2} \,,\; W \in \mathbb{R}^{d_1 \times d_2} \,, \tag{25}$$

with the large matrix W, we can introduce a matrix ΔW as

$$x' = (W + \alpha \Delta W)x + b \qquad \text{with} \qquad \Delta W = W_B W_A \,,\; W_B \in \mathbb{R}^{d_1 \times r} \,,\; W_A \in \mathbb{R}^{r \times d_2} \,, \tag{26}$$

where W_A and W_B are trainable, but W is frozen. The combination ΔW has at most rank r, which is a hyperparameter. For LoRa to be effective, it must satisfy

$$r \ll \frac{d_1 d_2}{d_1 + d_2} \,. \tag{27}$$

The matrix W_B is typically initialized with vanishing weights, such that the weight matrix ΔW does not initially modify the output of the affine layer. For the hyperparameter α we choose $\alpha = 2$ throughout.

Prompt tuning [77]. For this training technique, a new special token x_s is added to the vocabulary. Then, every sequence gets prepended by this special token,

$$(x_1, \ldots, x_n) \longrightarrow (x_s, x_1, \ldots, x_n) \,, \tag{28}$$

and only the embedding of this token, $E(x_s) \in \mathbb{R}^d$, is trained.

3 Lightcone Large Language Model (L3M)

3.1 Architecture

Our goal is to see if a pretrained LLM can be used for numerical fundamental physics data and if the out-of-domain pretraining leads to a performance gain. We review a few approaches and their (dis)advantages and motivate our method:

1. We can straightforwardly express numerical data as text and query the task, as has been done for arithmetics [78, 79], regression [80, 81] and extrapolation [82]. Although LLMs can, in principle, solve these problems with in-context learning, they perform poorly and require dedicated training [83, 84]. In general, it is hugely inefficient to express numerical data as text, especially because the resulting sequences are intractably long.

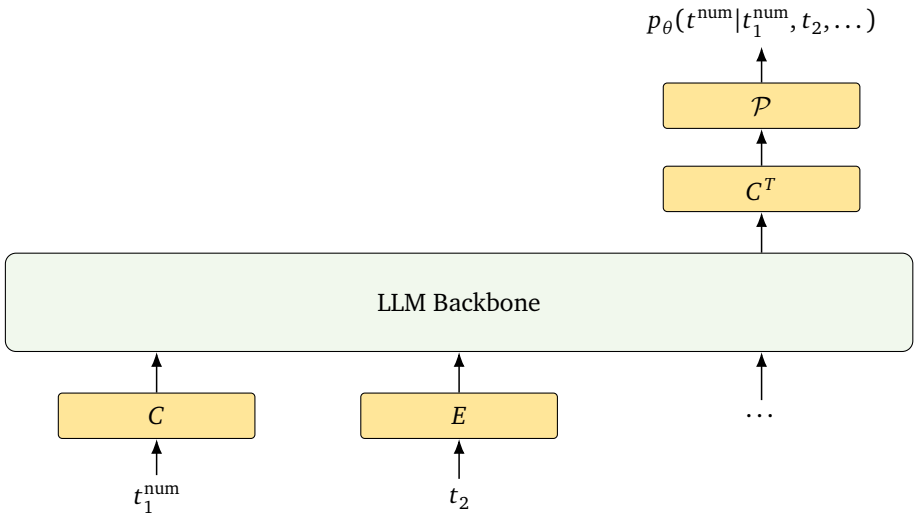

Figure 2: L3M setup connecting numerical tokens with the LLM backbone transformer.

2. Alternatively, we can work with multi-modal LLMs [85], which combine text and non-linguistic data. The latter is encoded with additional networks, e.g. vision transformers. The resulting embeddings are input to the LLM backbone. There are different training strategies to align the different modalities, linguistic-inspired next-token prediction is one of them. Since the generated output is text, this approach is not obviously suitable for physics.

Instead of these approaches we adapt the LLM architecture. We remember that the (un)embedding maps, E and E^T, connect the linguistic-coded tokens with corresponding representations, for which the backbone learns correlations. We re-purpose the LLM backbone for physics data in analogy to finetuning. However, we change the modality of the pretraining and the finetuning data, so our ansatz can be viewed as model reprogramming [86]. The non-local and long-range correlations of the linguistic modality make this approach very interesting, as learning them requires a lot of computing resources.

To utilize the transformer architecture, the physics data has to be represented as a sequence of numerical 'tokens' in analogy to Eq.(1),

$$\left(t_1^{\text{num}}, \dots, t_n^{\text{num}}\right), \qquad t_i^{\text{num}} \in \mathbb{R}^{d_{\text{num}}}. \tag{29}$$

In principle, the numerical tokens can be discrete, but they will not be in our architecture. To connect the numerical tokens to the backbone transformer we introduce input and output connectors, C and C^T, in analogy to the (un)embedding maps. The input connector simply maps the numerical tokens to the latent space of the backbone, while the output connector is combined with a predefined map $\mathcal{P}$ that yields a parametrization of the conditional probability $p(t_i^{\text{num}}|\cdot)$. For example, in the case of linguistic tokens we have $\mathcal{P} = \text{Softmax}$ and its normalized outputs define a categorical distribution. For several numerical modalities each of them gets an input and output connector network.

LLMs finetuned for time series forecasting [87–90] serve as a toy model for generative physics tasks or extrapolation. In particular, Ref. [87] re-programs the LLM backbone and achieves competitive results, supporting our L3M ansatz.

Our architecture is illustrated in Fig. 2. The input sequence starts with a numerical token, t_1^{num}, followed by a linguistic-coded token, t_2. The former is connected to the LLM backbone with an input connector, C, and the latter with the embedding map E. The output connector,

C^T, yields a parameterization of $p(t^{\text{num}}|t_1^{\text{num}}, t_2, \dots)$, which gets translated into a probability density by $\mathcal{P}$. For this paper, we use the small Qwen2.5-0.5B-Instruct LLM, because its limited size allows us to test different setups.

3.2 21cm lightcone data

We use complex data of 21cm background fluctuations as a testbed for an LLM performing standard cosmological tasks. The SKA, as the current state-of-the-art interferometer, enables the 3D mapping of neutral hydrogen, the most abundant baryonic element in the Universe, for over 50% of the observable Universe. The 3D lightcones of the 21cm signal, 2D spatial + 1D temporal, represent the brightness temperature offset $\delta T_{21}(x, \nu)$ measured against the Cosmic Microwave Background (CMB), with on-sky coordinates x and frequency ν (or equivalently, redshift z), as measured by a radio interferometer such as the SKA. For the regression and generative tasks in Secs. 4 and 5 we create a training dataset of several thousand lightcones.

21cm lightcones are created with the publicly available semi-numerical (approximate hydrodynamical) code `21cmFASTv3` [91, 92]. It generates initial density and velocity fields and evolves them in time, or redshift, at second-order Lagrangian perturbation theory using the Zel'dovich approximation [93]. Ionized regions are identified in an excursion set formalism by filtering the matter density field with a top-hat filter of decreasing size. A region at a certain filter scale is flagged as ionized, with a neutral fraction $x_{\text{HI}} = 0$, if the fraction of collapsed matter, f_{coll}, exceeds the inverse ionizing efficiency of star formation, ζ^{-1}. Partially ionized regions are accounted for with an ionized fraction $1 - x_{\text{HI}} = f_{\text{coll}}\zeta$.

The resulting 21cm brightness temperature field δT_{21} depends on ionized fraction x_{HI}, baryonic matter density as a tracer of the underlying dark matter field, and a flat background cosmology with a cosmological constant as

$$\delta T_{21}(x,z) \approx 27\, x_{\text{HI}}\,(1+\delta_{\text{b}})\left(\frac{H(z)}{\mathrm{d}v_{\parallel}/\mathrm{d}r_{\parallel} + H(z)}\right)\left(\frac{1+z}{10}\right)\left(\frac{0.15}{\Omega_{\text{m}}h^2}\right)^{1/2}\left(\frac{\Omega_{\text{b}}h^2}{0.023}\right)[\text{mK}] \quad (30)$$

with baryonic matter fluctuations $\delta_{\text{b}}(x,z)$, peculiar velocity field $\mathrm{d}v_{\parallel}/\mathrm{d}r_{\parallel}(x,z)$, Hubble function $H(z)$ for cosmological background expansion, and the matter density parameter Ω_{m}, Hubble parameter h, and baryonic matter density parameter Ω_{b} at present time. In this formula we assumed the so-called post-heating regime, where the spin temperature of neutral hydrogen is significantly larger than the CMB temperature, i.e., $T_{\text{S}} \gg T_{\gamma}$.

The resulting 21cm brightness offset fluctuation fields depend on several cosmological and astrophysical parameters. For our proof-of-concept study we combine parameters for cosmology and dark matter properties, with parameters describing astrophysics during cosmic dawn and the EoR (see also [10]):

Parameter	**Prior Range**
Matter density Ω_{m}	$\mathcal{U}[0.2, 0.4]$
Warm dark matter mass in keV m_{WDM}	$\mathcal{U}[0.3, 10]$
Minimum virial temperature in K T_{vir}	$\mathcal{U}[10^4, 10^{5.3}]$
Ionizing efficiency ζ	$\mathcal{U}[10, 250]$
X-ray energy threshold for self-absorption in eV E_0	$\mathcal{U}[100, 1500]$
Specific X-ray luminosity in erg/s log L_{X}	$\mathcal{U}[38, 42]$

Table 1: Summary of the cosmological (dark matter) and astrophysical parameters sampled to simulate the 21cm signal along with their prior ranges.

1. Matter density $\Omega_\mathrm{m} \in [0.2, 0.4]$
   It controls structure formation, where the chosen values encompass the Planck limits [94];
2. Warm dark matter mass $m_\mathrm{WDM} \in [0.3, 10]\,\mathrm{keV}$
   The prior range allows for a variety of phenomenological behavior; here the lower limit significantly deviates from a with Cold Dark Matter (CDM) scenario. Current astrophysical constraints favor mass values larger than a few keV [95, 96]. The larger m_WDM, the more structure formation and the distribution of DM halos look similar to CDM, as the free-streaming length is inversely proportional to the WDM mass;
3. Minimum virial temperature $T_\mathrm{vir} \in [10^4, 10^{5.3}]\,\mathrm{K}$
   This parameter defines the minimum virial temperature of dark matter halos required for cooling that is efficient enough for star formation to take place. It is defined by atomic cooling limits and observations of Lyman-break galaxies [97];
4. Ionization efficiency $\zeta \in [10, 250]$
   The ionization efficiency determines if a region is flagged as ionized. It is a composite parameter determined by both star formation parameters and recombinations in the IGM via

$$\zeta = 30 \frac{f_\mathrm{esc}}{0.3} \frac{f_\star}{0.05} \frac{N_{\gamma/b}}{4000} \frac{2}{1+n_\mathrm{rec}}\,, \tag{31}$$

   where f_esc is the escape fraction of ionizing UV photons into the IGM, $f_\star$ is the fraction of baryonic gas bound in stars, $N_{\gamma/b}$ is the number of ionizing photons emitted per baryon by stars, and n_rec is the number density of hydrogen recombinations in the IGM, calculated for example based on local gas densities;
5. Specific X-ray luminosity $L_\mathrm{X} \in [10^{38}, 10^{42}]\,\mathrm{erg\,s^{-1}\,M_\odot^{-1}\,yr}$
   Integrated luminosity at energies $< 2\,\mathrm{keV}$ per unit star formation rate in $\mathrm{M_\odot\,yr^{-1}}$ that escapes host galaxies;
6. X-ray energy threshold $E_0 \in [100, 1500]\,\mathrm{eV}$
   Energy threshold below which X-rays are absorbed by their respective host galaxies; X-rays with energies below E_0 do not escape the host galaxy and therefore do not contribute to heating and reionization.

Other cosmological parameters are fixed to the Planck ΛCDM values [98] and assume flatness. We take $\Omega_\mathrm{b} = 0.04897$, $\sigma_8 = 0.8102$, $h = 0.6766$, and $n_\mathrm{s} = 0.9665$.

To generate our training dataset of 21cm lightcones, we sample parameters from the uniform priors summarized in Tab. 1. For each parameter set we generate the corresponding lightcone in the redshift range $z = 5 - 35$. Each lightcone has a spatial box size of 200 Mpc at a resolution of 1.42 Mpc and consists of $(140, 140, 2350)$ voxels for 2350 temporal (redshift or frequency) bins. We note that the matter density Ω_m impacts the physical length in the temporal direction, as it changes the background time evolution of space-time. We therefore cut the highest-redshift voxels for a fixed number of 2350 temporal bins. Therefore, only for $\Omega_\mathrm{m} = 0.4$ the lightcones include $z = 35$, while smaller Ω_m values lead to lightcones slightly cropped at high redshift (lowest frequencies).

We use our dataset of around 5000 lightcones for training, validation, and testing. We filter extreme reionization histories that are strongly disfavored by current observational bounds, in terms of the optical depth [94] and the endpoint of reionization (small fraction of neutral hydrogen) being reached at $z \sim 5$ at the latest, as indicated by measurements of the Lyman-alpha forest [99, 100].

4 Parameter regression with frozen backbone

First, we examine the extent to which pretrained correlations in the LLM backbone can be utilized for physics tasks. As a benchmark task, we use the regression of simulation parameters from the 21cm lightcones, both astrophysical and related to dark matter (see Sec. 3.2 for a description of parameters and lightcone generation). To isolate the influence of pretraining, we completely freeze the backbone transformer, training only the connectors, and compare against a network where the weights of the backbone transformer are re-initialized. Any difference between the two networks can then be attributed to the pretrained LLM structure.

4.1 Data and connector architecture

For this regression task, we reduce the lightcones by spatially averaging the brightness temperature field, yielding the so-called global brightness temperature signal as a function of time, or redshift. In addition, we downsample the global signal by replacing 50 consecutive data points with their mean value, resulting in 47 brightness temperature values per lightcone, see Fig. 3. Each of these values is identified as a token. As preprocessing, we normalize the global signal to zero mean and unit variance and min-max normalize the 6 simulation parameters p_i from Sec. 3.2 as

$$p_i' \equiv \frac{p_i - p_{i,\min}}{p_{i,\max} - p_{i,\min}} \in [0,1]\,, \tag{32}$$

with $p_{i,\min}$ and $p_{i,\max}$ being the minimal and maximal values. The training, validation and test datasets consist of 3800, 960 and 250 lightcones, respectively.

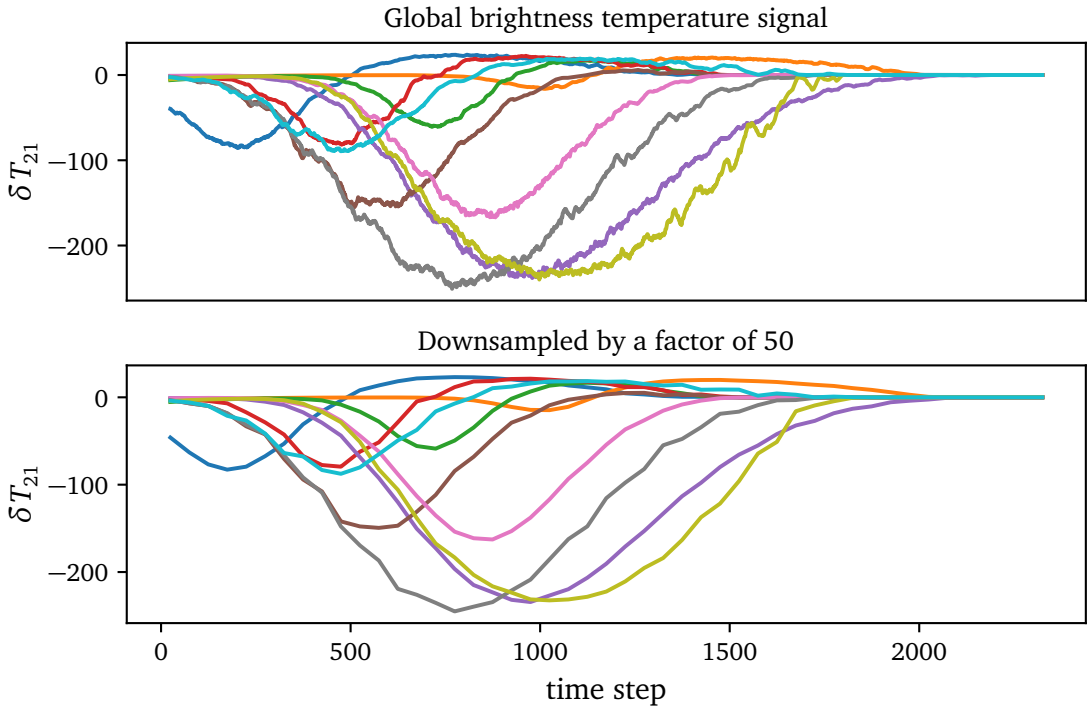

Figure 3: Global brightness temperature signal for 10 different lightcones and their corresponding downsampled distributions.

Architecture The networks follow the L3M architecture from Sec. 3.1. For regression, there are two numerical modalities: the global brightness temperature signal, $(t_1^{\mathrm{BT}}, \ldots, t_{47}^{\mathrm{BT}})$ as input and the target parameters $\vec{p}$ as output. For each of them we introduce a connector network. Large connectors improve the alignment of the numerical modalities with the linguistic token representations. On the other hand, they also reduce the importance of the backbone LLM — the connector networks may perform the regression while the backbone trivially transports the information. Since our focus is the backbone network, we use single affine layers for each connector.

We also introduce a learnable token, <|ska-param|>, which is appended to the input sequence after the brightness temperature tokens. The backbone embedding of this token,

$$z \equiv g\left(\texttt{<|ska-param|>} \,\middle|\, t_1^{\mathrm{BT}}, \ldots, t_{47}^{\mathrm{BT}}, \ldots\right), \tag{33}$$

can be interpreted as a summary embedding of the global signal, from which the simulation parameters are regressed. The final ellipsis in the above equation refers to additional tokens which we specify momentarily.

We model the systematic uncertainty of the regression as a Gaussian with a learned covariance matrix. The summary embedding z is inserted into the output connector, which predicts the mean values, $\vec{\mu}$, and the covariance matrix, Σ, of the Gaussian. Consequently, the network is trained with the heteroskedastic loss

$$\mathcal{L} = \frac{1}{2}\left\langle (\vec{p}-\vec{\mu})^T \Sigma^{-1} (\vec{p}-\vec{\mu}) - \log\det\Sigma^{-1} \right\rangle_{p_{\mathrm{data}}(\vec{p}\,|\,t^{\mathrm{BT}})}. \tag{34}$$

Due to the normalization of the parameter values from Eq. (32), the predicted mean values are activated with a sigmoid function, yielding

$$\vec{\mu} \in [0,1]^6. \tag{35}$$

The covariance matrix is parameterized by a lower triangular matrix with positive diagonal entries,

$$\Sigma^{-1} = LL^T \qquad \text{with} \qquad L \in \mathbb{R}^{15} \times \mathbb{R}_+^6. \tag{36}$$

A softplus activation function ensures that the diagonal elements are positive. Furthermore, we divide the values in the n-th row of L by $1/\sqrt{n}$ to unbias the initial covariance matrix. As an example, observe that

$$\begin{pmatrix} a & 0 \\ b & c \end{pmatrix}\begin{pmatrix} a & b \\ 0 & c \end{pmatrix} = \begin{pmatrix} a^2 & ab \\ ab & b^2+c^2 \end{pmatrix}. \tag{37}$$

We investigate 3 different prompting templates, containing the same information for the regression task but potentially additional (trainable) tokens:

1. **Minimal** contains only the necessary tokens,

   $t_1^{\mathrm{BT}} \ldots t_{47}^{\mathrm{BT}}$<|ska-param|>

2. **Chat-inspired** interleaves the numerical tokens with the chat template,

   <|im_start|>system
   <|im_end|>
   <|im_start|>user
   $t_1^{\mathrm{BT}} \ldots t_{47}^{\mathrm{BT}}$<|im_end|>
   <|im_start|>assistant
   <|ska-param|><|im_end|>

For the pretrained LLM, the tokens <|im_start|>, <|im_end|>, 'system', 'user' and 'assistant' have a pretrained embedding. For the randomly initialized network, we also randomly initialize the embeddings of these tokens and keep them frozen during training.

3. **Chat-inspired with trainable tokens** adds, in the spirit of prompt tuning [77], two tokens <|system-prompt-token|> and <|lightcone-token|> with trainable embeddings,

<|im_start|>system
<|system-prompt-token|><|im_end|>
<|im_start|>user
<|lightcone-token|>$t_1^{\mathrm{BT}} \dots t_{47}^{\mathrm{BT}}$<|im_end|>
<|im_start|>assistant
<|ska-param|><|im_end|>

The two connector network include 26.9k trainable parameters. The two trainable tokens of the last prompt template increase this number by 1.8k.

Training and reference networks The training hyperparameters are listed in the left panel of Tab. 2. For stable optimization, the input and output connector weights are updated separately. For each batch, the weights of the input connector are optimized first, after which the gradients are recomputed and the weights of the output connector are optimized. To reduce overfitting, we use attention dropout. In addition, we insert a copy of the test dataset into it. This way, a batch can contain two identical samples, which become effectively distinct due to attention dropout. Using the pretrained scaling factors of the final layer norm degrades the performance in early stages of training, so we always re-initialize those factors to ones.

To provide a reference for the regression results, we introduce two networks with the same structure as Qwen2.5, but trained from scratch. They do not have a causal attention mask, which means every token attends to every other token. We use the minimal prompt template since networks trained from scratch do not benefit from the chat template:

1. a small network illustrating the performance for a comparable number of trainable parameters as the L3M connectors.
2. a large network illustrating the ultimate performance of a dedicated network;

The network hyperparameters of the small network are determined via a hyperparameter search, while ones of the large network are reasonably chosen as we do not care about its best possible performance. The choices for each network are listed in the right panel of Tab. 2. Both reference networks are trained with the setup described above, but with two adjustments: (i) they are trained for 40,000 epochs without changing the warm-up and decay periods; (ii)

Batch size	1024
Epochs	1500
Learning rate	$5 \cdot 10^{-5}$
Learning rate schedule	100 epochs linear warm-up, 1300 epochs stable, 100 epochs cosine decay
Max. gradient norm	30
Attention dropout	10^{-3}
Optimizer	Adam

Hidden dim	128	32
Transformer blocks	6	3
Query heads	4	2
Key-Value heads	4	2
MLP hidden dim	256	64
Number of parameters	990k	32K

Table 2: Training (left) and reference network (right) hyperparameters. The number of trainable parameters of the small reference network matches the L3M connectors.

since all network parameters are trained, we do not use the interleaved training and update all parameters simultaneously.

4.2 Results

First, we look at the training efficiency of the L3M compared to the two reference networks. We show the validation loss during training in Fig. 4. The top row is grouped by prompt template: minimalist, chat-inspired, or chat-inspired with trainable tokens; the bottom row by backbone initialization: pretrained or random. In each panel, we provide the best validation losses of the reference networks. Throughout, the final validation losses of the pretrained and the randomly initialized L3Ms lie between the two reference losses, and the pretrained backbone always outperforms the random backbone. This indicates that the L3M performance is sensible.

A nontrivial common feature is that the chat template significantly boosts the performance of the pretrained network, despite adding no information to the regression task. It increases the speed of convergence and also improves the loss at each epoch. Since the embeddings for the chat template tokens are manifestly aligned with the LLM latent space through pretraining, we rationalize their benefit as providing structural information for the to-be-trained embeddings of the numerical tokens. For the randomly initialized network, the chat template has little effect, improving the network performance only at the end of training. Adding trainable tokens does not significantly affect the loss. Figure 4 shows that the pretrained LLM backbone is more computationally efficient than the random backbone.

A second observation is that even randomly initialized L3M backbone weights outperform

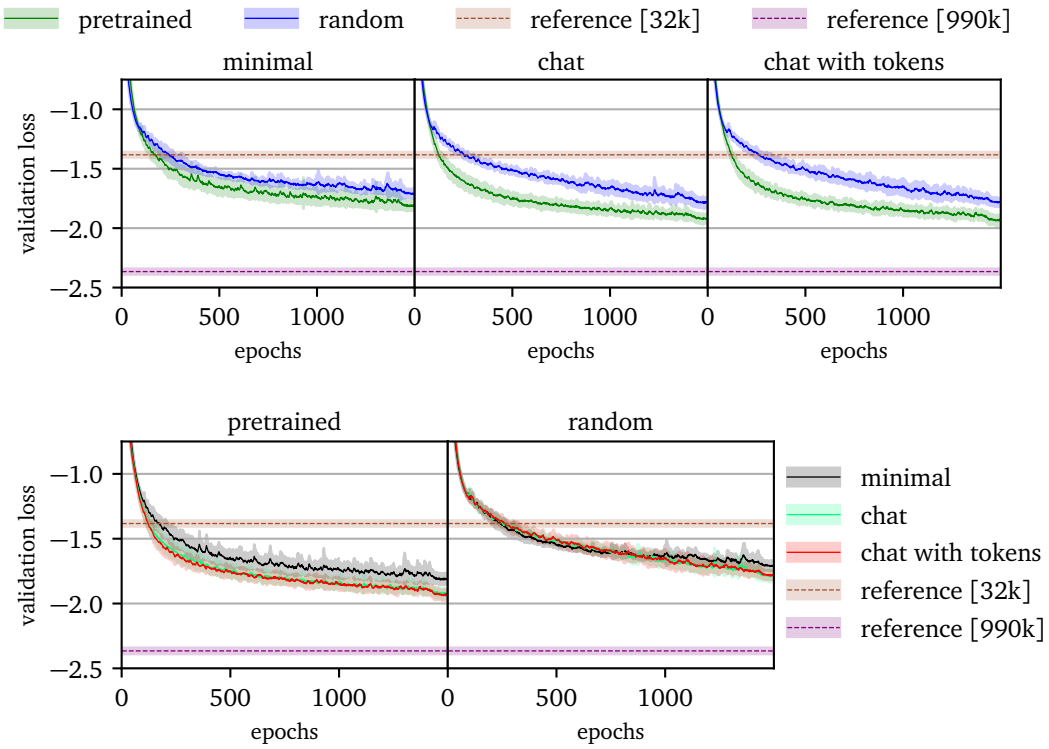

Figure 4: Mean validation loss with a 1σ-band determined from 8 runs, grouped by prompting template (upper) and backbone initialization (lower). For the reference networks, only the best validation loss is shown.

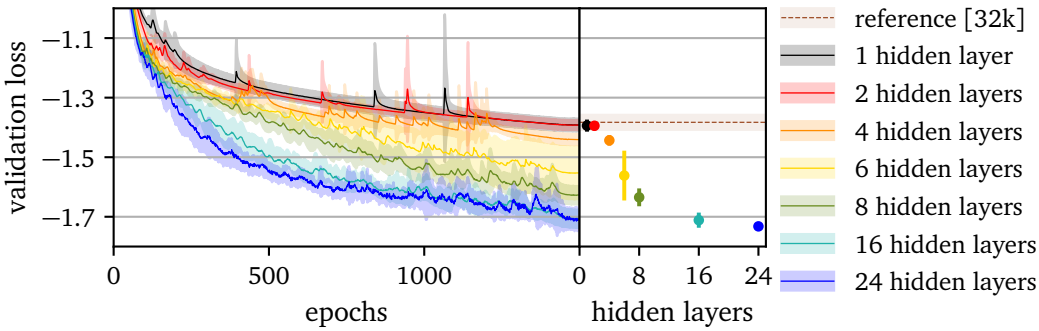

Figure 5: Mean validation loss with a 1σ-band for randomly initialized backbone weights and the minimalist chat template. The number of hidden layers of the backbone network is varied. The right panel shows the best validation losses, including the small reference network.

the reference network with a similar number of trainable parameters. To understand this, we train the L3M with randomly initialized backbone and the minimal chat template for varying numbers of transformer blocks in the backbone. The resulting validation losses are shown in Fig. 5. Although the backbone is not trained, increasing the depth improves the network performance.

Finally, in Fig. 6, we show the best regression results, measured by validation loss, for the astrophysical and cosmological parameters introduced in Sec. 3.2. We show results for the pretrained and randomly initialized L3Ms, as well as both reference networks. For each lightcone of the test set, we sample 50 parameter sets from the regressed Gaussian distribution. Since the parameter values are bounded from above and below, we clamp the sampled parameters into this interval. After that, we bin the resulting parameters and compute the standard deviation for each bin. First, we observe that the shapes of the regressed parameters agree for all setups, including the known limitations discussed in detail in Refs. [22, 29]. Especially in the lower panels we also see that the reference network with the small number network parameters, comparable to the L3M connectors, performs much worse than both L3M setups. The performance of the pretrained L3M tends to be slightly better than the random-initialized L3M, almost matching the performance of the large reference network.

5 Generation with finetuned backbone

Now, we turn to the more sophisticated task of generating slices of lightcones. For this task we will also finetune the LLM, rather than just training the connector networks on a pretrained LLM backbone.

5.1 Data and connector architecture

We interpret a lightcone as a time series of spatial slices, $\delta T_{BT}(\vec{x}; t)$, represented as a sequence of continuous tokens. Every spatial slice is divided into patches of 14×14 pixels and each patch is identified as a token. Then, the resulting 2d grid of patches is flattened into a sequence of patches, inserting a new-line token, `<|nl_1|>`, after every line break. Finally, the sequences of spatial slices are concatenated by interleaving another new-line token, `<|nl_2|>`, as illustrated in Fig. 7. This representation partially breaks the inherent local 3D structure of the

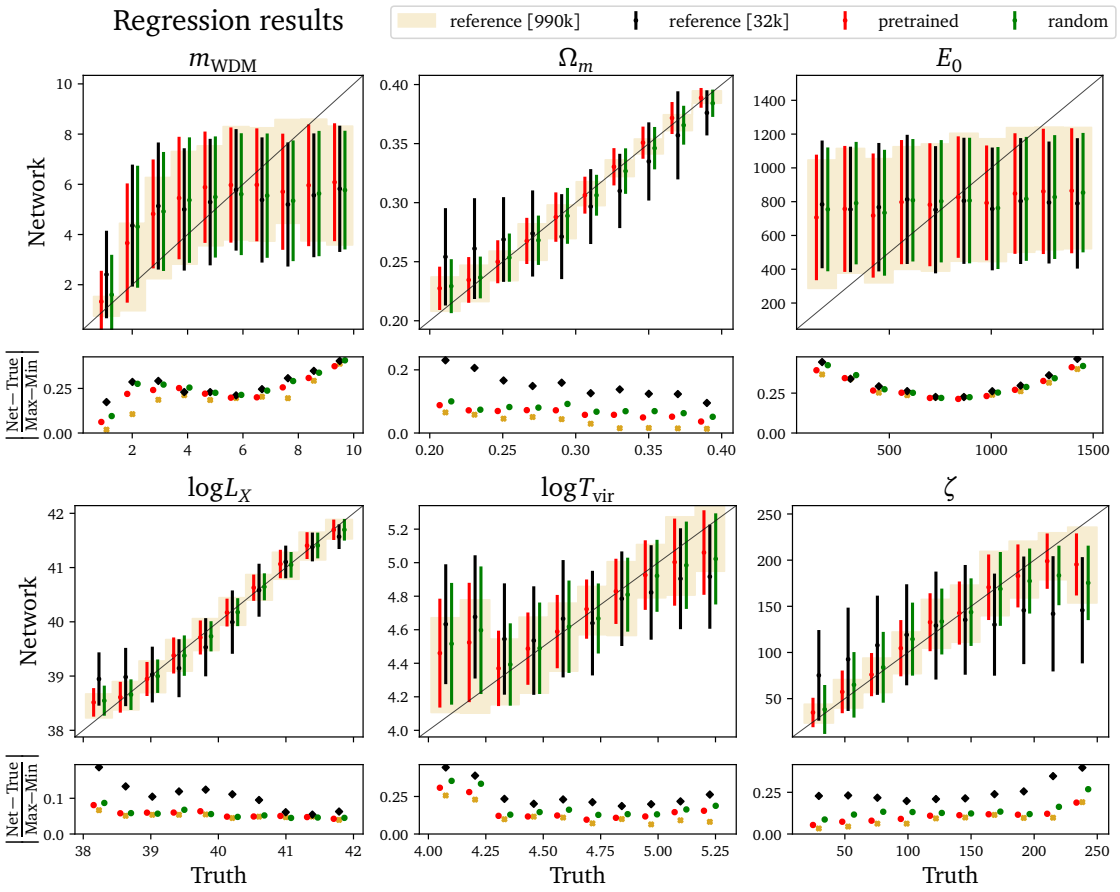

Figure 6: Regressed parameters after sampling 50 times from each regressed Gaussian distribution.

lightcone, forcing the L3M network to recognize long-range correlations, for which the pretrained LLM backbones should be advantageous.

The generative task of the network is next-patch prediction, or next-token prediction, as discussed in Sec. 2.2. To render the sequence length feasible, we restrict to 12 consecutive spatial slices, which we call a sublightcone. These contain a manageable 12,000 patches in total. The first two slices are excluded from the next-patch prediction, ensuring that each forecast slice depends on at least two preceding slices. This guarantees that a velocity field can be extracted from the context. Since the simulation parameters and the redshift influence the evolution of the brightness temperature distribution, we condition the task on those parameters and the time step of the first spatial slice.

At late times, the brightness temperature distribution remains zero for most of the lightcones. We trim the lightcones by keeping the first 100 slices of this period and removing the remaining ones. Additionally, the brightness temperature distribution contains outliers with large absolute values. To regularize that distribution, we effectively clamp the distribution by preprocessing the brightness temperature via

$$\delta T'_{\mathrm{BT}} = \mathrm{sgn}(\delta T_{\mathrm{BT}}) \cdot \ln |\delta T_{\mathrm{BT}}| \ . \tag{38}$$

The resulting values are normalized to zero mean and unit variance. The parameter values and the time step of the first slice are min-max normalized as specified in Eq.(32).

We divide the lightcones into a training, validation and test set, dataset consisting of 3800, 540 and 750 lightcones, respectively. During training, we augment every sublightcone with

spatial rotations, reflection and translations. The latter is possible because the lightcones have periodic boundary conditions. More details about the augmentation and processing of the sublightcones can be found in the App. B.

Architecture For generation, we introduce 8 input connectors, one for each of the 6 simulation parameters, $\vec{p}$, one for the time step of the first sublightcone slice, t_{init}, and one for the brightness temperature patches, t^{BT}. For the latter, we also add an output connector. Altogether, we want to train a network that encodes the conditional probability, where the condition is represented internally as z,

$$p_\theta\left(t_i^{\text{BT}}|z\right) \approx p\left(t_i^{\text{BT}}|t_{i-1}^{\text{BT}},\dots,t_1^{\text{BT}},\vec{p},t_{\text{init}}\right) . \tag{39}$$

To focus again on the backbone, all connectors consist of one affine layer.

To translate the parameterization z into the probability density of Eq.(39) we need a flexible family of distributions which can encode the high inter-pixel correlations of a patch. We use Conditional Flow Matching (CFM) [55,60,101], which is known for this ability. It requires another network to learn a velocity field v_θ that transports samples of a Gaussian distribution to samples of another distribution along an internal time direction, $\tau \in [0,1]$. By making this vector field dependent on the parameterization z, the CFM setup is able to yield the desired mapping

$$\frac{dx(\tau)}{d\tau} = v(x,\tau,z) \qquad \text{with} \quad x(\tau) \sim \begin{cases} \mathcal{N}(\mu=0,\sigma=1) & \tau=0 \\ p(t_i^{\text{BT}}|t_{i-1}^{\text{BT}},\dots,t_1^{\text{BT}},\vec{p},t_{\text{init}}) & \tau=1 \end{cases} . \tag{40}$$

The network is then trained to regress the vector field interpolating linearly between sampled endpoints,

$$\begin{aligned} \mathcal{L} &= \left\langle (v - v_\theta(x_\tau,\tau,z))^2 \right\rangle_{p(t_i^{\text{BT}}|t_{i-1}^{\text{BT}},\dots,t_1^{\text{BT}},\vec{p},t_{\text{init}}),\mathcal{N}(x_0;0,1),U(\tau;0,1)} \\ &\text{with} \quad v \equiv t_i^{\text{BT}} - x_0 \quad \text{and} \quad x_\tau \equiv (1-\tau)x_0 + \tau t_i^{\text{BT}} . \end{aligned} \tag{41}$$

The network encoding the velocity is a convolutional neural network, which inputs 14×14 patches with 4 channels. The first two channels contain the transported sample x_τ and the internal time τ, the latter being expanded to match the patch shape. The output connector of the brightness temperature patches yields a parameterization $z \in \mathbb{R}^{d_z}$ with dimension

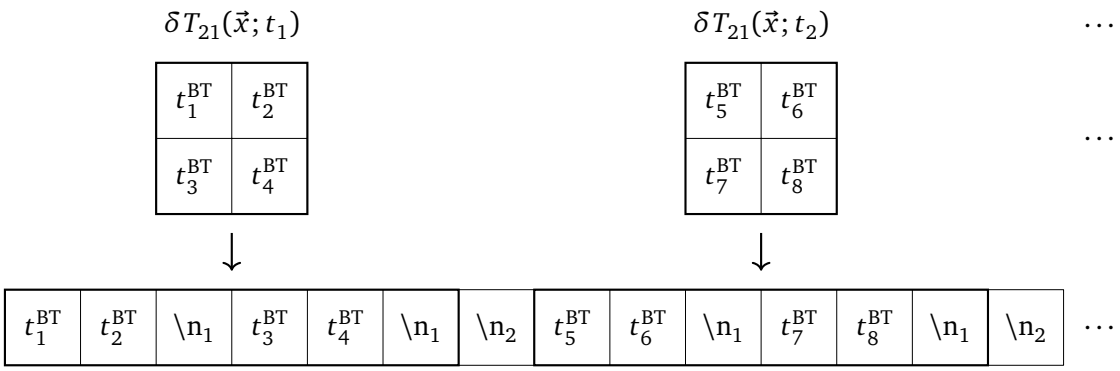

Figure 7: Illustration of the 3D lightcone representation as a sequence of continuous tokens. Every spatial slice, $\delta T_{BT}(\vec{x};t_i)$, is patched and the resulting grid is flattened. The resulting sequences are concatenated into a single one. Two newline characters, $\backslash n_1$ and $\backslash n_2$, are inserted to keep track of the 3D structure.

$d_z = 2{\cdot}14{\cdot}14$, which gets reshaped to fill the remaining 2 channels. The convolutional network consists of a convolutional input layer mapping the 4 channels to 64, 6 residual convolutional layers and an output convolutional layer mapping the 64 channels to 1. A residual layer consists of one convolutional layer mapping the 64 channels to 128 and another convolutional layer with a kernel size of 1 mapping the 128 channels to 64, followed by the residual connection. Every convolutional layer has a kernel size of 3 if not stated differently. The convolutional layers are interleaved with silu activations, and the padding is chosen to keep the patch shape. In total, the convolutional velocity network contains 490k parameters.

Training Starting from either the Qwen2.5 or random backbone, we finetune L3M

1. completely (360M parameters);
2. with LoRA of rank $r = 8$ (5.5M parameters); or
3. with LoRA of rank $r = 2$ (2.2M paramerers).

All layer norm weights are trained in all cases. As for the regression task, the scaling factors of the final layer norm are reinitialized to ones for the pretrained LLM setup. The number of trainable parameters includes the convolutional velocity network. For the L3M setups, we use the prompt

<|im_start|>system
<|system-prompt-token|><|im_end|>
<|im_start|>user
<|parameter_0|> ...<|parameter_5|> <|time_step|> <|im_end|>
<|im_start|>assistant
<|lightcone-token|>$t^{\mathrm{BT}}_1 \dots t^{\mathrm{BT}}_{196}$<|nl_1|><|nl_2|>$t^{\mathrm{BT}}_{197} \dots$<|im_end|>

In addition, we also train reference networks from scratch which have a similar number of parameters as the previous setups. For these networks, we use the prompt

<|system-prompt-token|> <|parameter_0|> ...<|parameter_5|>
<|time_step|> <|lightcone-token|> $t^{\mathrm{BT}}_1 \dots t^{\mathrm{BT}}_{196}$ <|nl_1|> <|nl_2|>
$t^{\mathrm{BT}}_{197} \dots$

Here, <|parameter_0|>, ..., <|parameter_5|> and <|t_index|> are placeholders for the parameter- and time-step-modalities. Moreover, <|system-prompt-token|> and <|lightcone-token|> are additional trainable tokens.

Batch size	64
Epochs	20
Learning rate	$5 \cdot 10^{-5}$
Learning rate schedule	1 epochs linear warmup, 18 epochs stable, 2 epochs cosine decay
Weight decay	10^{-3}
Max. gradient norm	1
Attention dropout	10^{-2}
Optimizer	AdamW

Hidden dim	128	256
Transformer blocks	7	7
Query heads	4	4
Key-Value heads	4	4
MLP hidden dim	424	560
Number of parameters	2.18M	5.48M

Table 3: Training (left) and reference network (right) hyperparameters. The number of parameters match the number of trainable of the LoRA finetuned L3M networks.

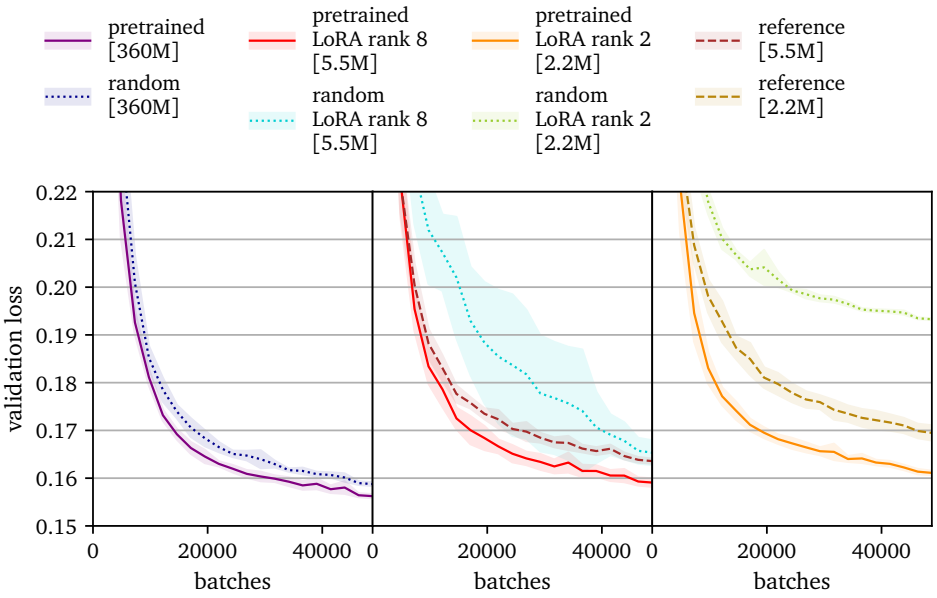

Figure 8: Mean validation loss with a 1σ-band determined from 5 runs. Due to the large variations of the randomly initialized L3M backbone finetuned with LoRA of rank 2, this setup has been finetuned 7 times.

We do not extensively optimize the hyperparameters of the reference networks since we focus on the comparison between pretrained and random backbone weights. The selected hyperparameters for the L3M and reference networks are listed in Tab. 3. Since training the randomly initialized L3M backbone completely matches our definition of a reference network, we do not train an additional reference network for this case. The only conceptual difference to the other two reference networks is the chosen prompt template. However, we do not expect any significant difference from replacing it.

5.2 Results

We again start by showing the validation loss of different L3M setups in Fig. 8. Each curve is the mean value from five runs and the band covers one standard deviation. The L3M setups differ both by the initialization of the backbone and by the number of finetuned parameters, and we group the networks by the latter in Fig. 8 in descending order from left to right. Due to the data augmentation, every instance of a batch can be viewed as a new sample. We therefore show the loss as a function of the batch number.

Similarly to the regression task, we find that the best performing setup at each number of trainable parameters is the pretrained LLM backbone. This holds for the case with maximal number of trainable parameters, where the random backbone is essentially a dedicated network. When using LoRA finetuning, only the L3M with pretrained backbone networks improve on the dedicated network. In particular at LoRA rank 2, the L3M with random backbone essentially fails as we demonstrate later on. This is expected to some degree, since it is difficult to introduce meaningful structure to the backbone through rank 2 modifications. Meanwhile, the LLM backbone can be effectively finetuned beyond the performance of the dedicated network, even at LoRA rank 2. These results therefore highlight the advantage offered by pretrained backbones, where the existing structure can be repurposed for the task at hand even through finetuning only a small fraction of the weights.

L3M setup	trainable parameters	next-patch MSE
pretrained	360M	0.08393 ± 0.00021
random	360M	0.0871 ± 0.0003
pretrained LoRA rank 8	5.5M	0.0875 ± 0.0012
reference	5.5M	0.0951 ± 0.0003
random LoRA rank 8	5.5M	0.098 ± 0.004
pretrained LoRA rank 2	2.2M	0.0930 ± 0.0014
reference	2.2M	0.1039 ± 0.0028
random LoRA rank 2	2.2M	0.2232 ± 0.0008

Table 4: MSE for next patch prediction, as described in the text. The different L3M setups are grouped in the same way as in Fig. 8. The uncertainties are the standard deviations of the different runs.

The loss in Fig. 8 measures how well the network is able to regress the CFM target vector field specified in Eq.(41). To determine how well the network approximates the conditional distribution from Eq.(39), we compute the MSE for individual predicted patches in the test set. First, we load the checkpoint with the best validation loss for each run. Then, we sample the next patch conditioned on the preceding ground truth patches, and finally compute the MSE relative to the ground truth. The resulting MSE together with the variation between different runs are stated in Tab. 4. The order of network performance remains the same. However, this measure amplifies the difference between the pretrained and randomly initialized backbone finetuned with LoRA of rank 2.

As a validation that the networks generate coherent lightcone slices, we present samples in Figure 9. These samples belong to the same representative lightcone where for different time steps two consecutive slices are autoregressively generated with a context of 10 preceding slices. While the pretrained L3M network finetuned with LoRA of rank 2 generates marginally worse slices than the completely finetuned L3M network with pretrained backbone, the randomly initialized network finetuned with LoRA of rank 2 is only able to generate typical patches for the corresponding brightness temperature value range. It is not able to forecast the evolution of the large scale structure and sometimes, it even fails to generate coherent slices.

6 Conclusion

The impressive success of pretrained transformers in fundamental physics bears the question, if the largest and most general pretrained networks, LLMs, can be used for physics data. In this paper, we have shown for the first time that LLM backbones can be successfully finetuned for SKA data. We introduced a scheme for adapting an LLM to new modalities via connector networks and applied it to the 0.5B Qwen2.5 language model. To judge the specific advantage offered by the pretrained weights, we compared to a baseline with the same LLM architecture but with completely re-initialized weights.

As benchmark tasks for our Large Lightcone Language Model (L3M), we studied (i) regression of cosmological and astrophysical parameters from the spatially-averaged 21cm signal and (ii) forecasting full 21cm lightcone slices. In both tasks, we found that pretrained LLM weights improve the performance over the re-initialized baseline for a given number of training iterations, indicating improved convergence and data-efficiency. Interestingly, the advantage can be amplified by wrapping the input data with a chat-style template.

We also compared the L3M to dedicated reference networks with a matching number of

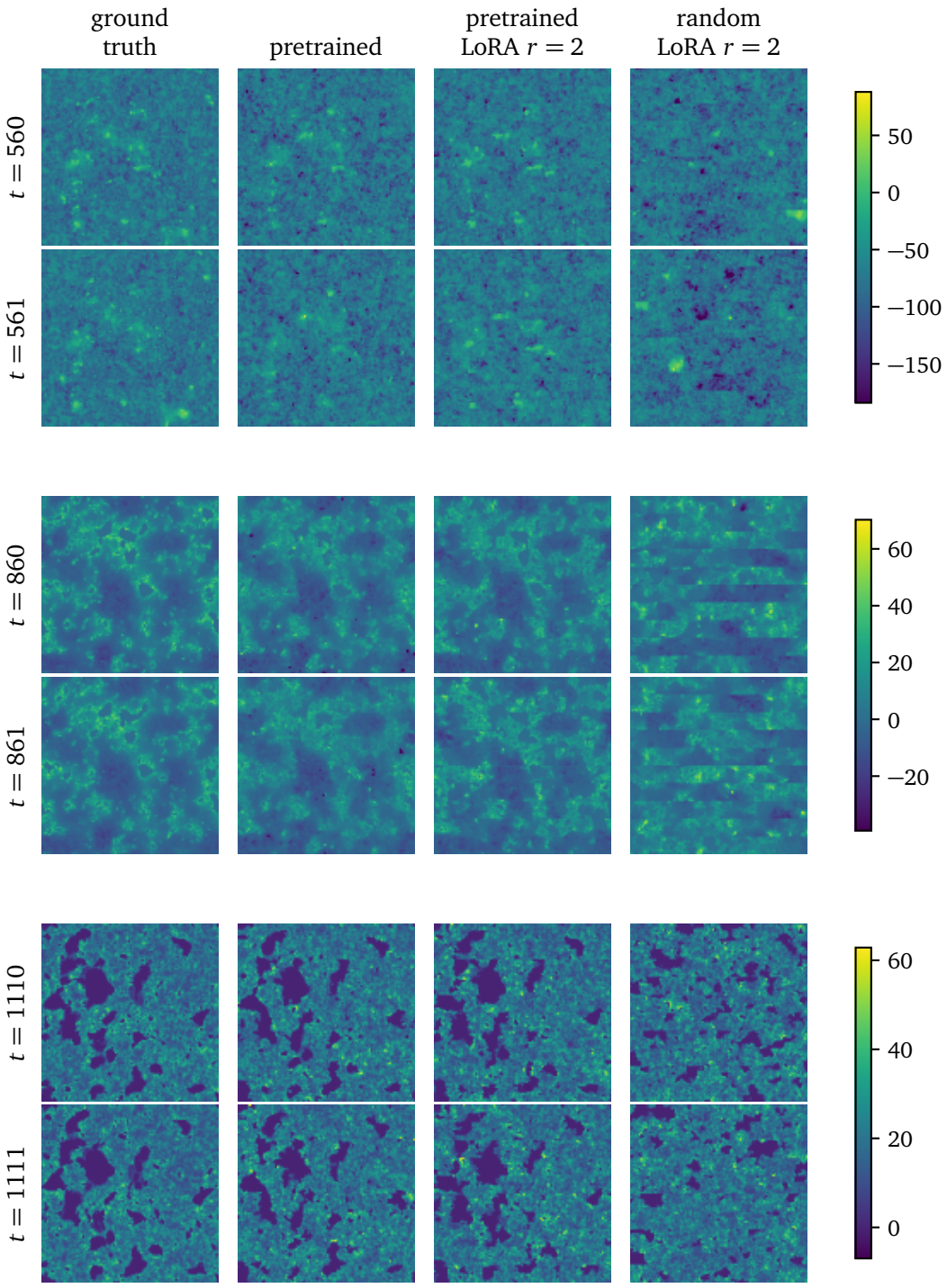

Figure 9: Forecast slices as described in the text for different time steps t by a pretrained L3M setup which is either completely finetuned or partially finetuned with LoRA of rank 2 and a randomly initialized L3M setup finetuned with the same LoRA configuration. The pixel values is the post-processed brightness temperature distribution.

trainable parameters. Again, the pretrained LLM backbones outperform these references. For the regression, we also observe this behavior in the randomly-initialized network and identified a scaling with the number of transformer blocks in the network. For the more difficult generative task, the random backbones do not improve on a dedicated network. However, pretrained and then finetuned LLM backbones retain their advantage. Our results suggest that pretrained weights from LLMs offer a strong initialization, even for truly out-of-domain fundamental physics tasks. Whether our observed efficiency and performance gains justify the use of LLMs in fundamental physics needs to be evaluated for a given specific application.

Code availability

The code for this project is published at https://github.com/heidelberg-hepml/L3M.

Acknowledgements

We thank Nina Elmer, Henning Bahl and Ramon Winterhalder for showing us how to unbias the heteroskedastic covariance matrix.

CH's work is funded by the Volkswagen Foundation. This work is supported through the KISS consortium (05D2022) funded by the German Federal Ministry of Education and Research BMBF in the ErUM-Data action plan, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant 396021762 – TRR 257: *Particle Physics Phenomenology after the Higgs Discovery*, and through Germany's Excellence Strategy EXC 2181/1 – 390900948 (the *Heidelberg STRUCTURES Excellence Cluster*). We acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG.

Hidden dim	896
Transformer blocks	24
Query heads	14
Key-Value heads	2
MLP hidden dim	4,864
MLP activation	silu
RMS norm ϵ	10^{-6}
RoPE base frequency	10^6
Shared embedding weights	✓
Parameters	0.49B
Non-embedding parameters	0.36B
Max sequence length	32,768
Max generation length	8,192
Vocabulary size	151,936

Table 5: Network hyperparameters of Qwen2.5-0.5B.

A Specifics about Qwen2.5

In this section, we review the hyperparameters of the Qwen2.5-0.5B-Instruct LLM and the general Qwen2.5 training setup [47]. The network hyperparameters are stated in Tab. 5 and the training hyperparameters in Tab. 6.

The pretraining has been split into two stages: First, the networks are trained with sequences consisting of 4,096 tokens and a RoPE base frequency of 10^5. Then, this base frequency is enlarged to 10^6 and the network is trained on sequences with $32,768$ tokens.

After that, the Qwen2.5 networks undergo a supervised finetuning stage, with focus on generating long sequences with $8,192$ tokens, following instructions, understanding structured data, and reasoning.

Finally, the networks are trained with Reinforcement Learning in two stages. In the first stage, the networks are trained with Direct Preference Optimization, which only requires one positive and one example for every query. This labeling has been created semi-automatically, without the use of any reward model. In the second stage, a reward model is trained. Another dataset is joined with the previous Reinforcement Learning dataset. In this stage, Group Relative Policy Optimization is used, sampling 8 responses per query.

B Generation dataset

In advance of training, the lightcones are split into sublightcones consisting of 2 initial slices and 48 remaining ones. To split a lightcone, we iterate along the time direction with a step size of 48. This guarantees that the initial 2 slices (except the very first ones) are also part of another sublightcone for which the next-patch loss can be computed. During the sampling of a batch, a random interval of 12 slices is chosen. In this way, the network is able to see different initial slice configurations.

To reduce the RAM footprint, each lightcone slice is loaded in FP16. After sampling a batch, the lightcones are converted to FP32 and afterwards, the pre-processing defined in Eq.38 is applied.

Pretraining:	
Dataset size	17T tokens
Sequence lengths	4,096 - 32,768
Supervised finetuning:	
Dataset size	$\sim$ 1M examples
Sequence length	32,768
Epochs	2
Learning rate	Decay from $7 \cdot 10^{-6}$ to $7 \cdot 10^{-7}$
Weight decay	0.1
Gradient clipping	1.0
Reinforcement Learning I:	
Dataset size	$\approx$ 150,000 examples
Epochs	1
Learning rate	10^{-7}
Reinforcement Learning II:	
Batch size	2048

Table 6: Training hyperparameters for Qwen2.5 models.

References

[1] K. Cranmer, J. Brehmer, and G. Louppe, *The frontier of simulation-based inference*, Proc. Nat. Acad. Sci. **117** (2020) 48, 30055, arXiv:1911.01429 [stat.ML].

[2] S. Badger *et al.*, *Machine learning and LHC event generation*, SciPost Phys. **14** (2023) 4, 079, arXiv:2203.07460 [hep-ph].

[3] C. Dvorkin *et al.*, *Machine Learning and Cosmology*, in *Snowmass 2021*. 3, 2022. arXiv:2203.08056 [hep-ph].

[4] E. Cuoco *et al.*, *Enhancing Gravitational-Wave Science with Machine Learning*, Mach. Learn. Sci. Tech. **2** (2021) 1, 011002, arXiv:2005.03745 [astro-ph.HE].

[5] A. Butter, G. Kasieczka, T. Plehn, and M. Russell, *Deep-learned Top Tagging with a Lorentz Layer*, SciPost Phys. **5** (2018) 3, 028, arXiv:1707.08966 [hep-ph].

[6] A. Bogatskiy, B. Anderson, J. T. Offermann, M. Roussi, D. W. Miller, and R. Kondor, *Lorentz Group Equivariant Neural Network for Particle Physics*, arXiv:2006.04780 [hep-ph].

[7] M. Favoni, A. Ipp, D. I. Müller, and D. Schuh, *Lattice Gauge Equivariant Convolutional Neural Networks*, Phys. Rev. Lett. **128** (2022) 3, 032003, arXiv:2012.12901 [hep-lat].

[8] S. Bulusu, M. Favoni, A. Ipp, D. I. Müller, and D. Schuh, *Equivariance and generalization in neural networks*, EPJ Web Conf. **258** (2022) 09001, arXiv:2112.12493 [hep-lat].

[9] S. Gong, Q. Meng, J. Zhang, H. Qu, C. Li, S. Qian, W. Du, Z.-M. Ma, and T.-Y. Liu, *An efficient Lorentz equivariant graph neural network for jet tagging*, JHEP **07** (2022) 030, arXiv:2201.08187 [hep-ph].

[10] S. Neutsch, C. Heneka, and M. Brüggen, *Inferring astrophysics and dark matter properties from 21 cm tomography using deep learning*, Mon. Not. Roy. Astron. Soc. **511** (2022) 3, 3446, arXiv:2201.07587 [astro-ph.CO].

[11] X. Zhao, Y. Mao, C. Cheng, and B. D. Wandelt, *Simulation-based Inference of Reionization Parameters from 3D Tomographic 21 cm Light-cone Images*, Astrophys. J. **926** (2022) 2, 151, arXiv:2105.03344 [astro-ph.CO].

[12] M. Favoni, A. Ipp, and D. I. Müller, *Applications of Lattice Gauge Equivariant Neural Networks*, EPJ Web Conf. **274** (2022) 09001, arXiv:2212.00832 [hep-lat].

[13] A. Bogatskiy, T. Hoffman, D. W. Miller, J. T. Offermann, and X. Liu, *Explainable equivariant neural networks for particle physics: PELICAN*, JHEP **03** (2024) 113, arXiv:2307.16506 [hep-ph].

[14] D. Murnane, S. Thais, and A. Thete, *Equivariant Graph Neural Networks for Charged Particle Tracking*, in *21th International Workshop on Advanced Computing and Analysis Techniques in Physics Research: AI meets Reality*. 4, 2023. arXiv:2304.05293 [physics.ins-det].

[15] D. Prelogović and A. Mesinger, *Exploring the likelihood of the 21-cm power spectrum with simulation-based inference*, MNRAS**524** (Sept., 2023) 4239, arXiv:2305.03074 [astro-ph.CO].

[16] A. Bhardwaj, C. Englert, W. Naskar, V. S. Ngairangbam, and M. Spannowsky, *Equivariant, safe and sensitive — graph networks for new physics*, JHEP **07** (2024) 245, arXiv:2402.12449 [hep-ph].

[17] J. Spinner, V. Bresó, P. de Haan, T. Plehn, J. Thaler, and J. Brehmer, *Lorentz-Equivariant Geometric Algebra Transformers for High-Energy Physics*, arXiv:2405.14806 [physics.data-an].

[18] J. Brehmer, V. Bresó, P. de Haan, T. Plehn, H. Qu, J. Spinner, and J. Thaler, *A Lorentz-Equivariant Transformer for All of the LHC*, arXiv:2411.00446 [hep-ph].

[19] D. Maître, V. S. Ngairangbam, and M. Spannowsky, *Optimal equivariant architectures from the symmetries of matrix-element likelihoods*, Mach. Learn. Sci. Tech. **6** (2025) 1, 015059, arXiv:2410.18553 [hep-ph].

[20] S. Nabat, A. Ghosh, E. Witkowski, G. Kasieczka, and D. Whiteson, *Learning broken symmetries with approximate invariance*, Phys. Rev. D **111** (2025) 7, 072002, arXiv:2412.18773 [hep-ph].

[21] D. Breitman, A. Mesinger, S. G. Murray, D. Prelogovic, Y. Qin, and R. Trotta, *21cmemu: an emulator of 21cmfast summary observables*, Mon. Not. Roy. Astron. Soc. **527** (2023) 4, 9833, arXiv:2309.05697 [astro-ph.CO]. [Erratum: Mon.Not.Roy.Astron.Soc. 533, 1045–1047 (2024)].

[22] B. Schosser, C. Heneka, and T. Plehn, *Optimal, fast, and robust inference of reionization-era cosmology with the 21cmPIE-INN*, SciPost Phys. Core **8** (2025) 037, arXiv:2401.04174 [astro-ph.CO].

[23] J. Spinner, L. Favaro, P. Lippmann, S. Pitz, G. Gerhartz, T. Plehn, and F. A. Hamprecht, *Lorentz Local Canonicalization: How to Make Any Network Lorentz-Equivariant*, arXiv:2505.20280 [stat.ML].

[24] I. V. Slijepcevic, A. M. M. Scaife, M. Walmsley, M. Bowles, O. I. Wong, S. S. Shabala, and S. V. White, *Radio galaxy zoo: towards building the first multipurpose foundation model for radio astronomy with self-supervised learning*, RAS Techniques and Instruments **3** (12, 2023) 19, https://academic.oup.com/rasti/article-pdf/3/1/19/61224539/rzad055.pdf.

[25] L. Parker, F. Lanusse, S. Golkar, L. Sarra, M. Cranmer, A. Bietti, M. Eickenberg, G. Krawezik, M. McCabe, R. Morel, R. Ohana, M. Pettee, B. Régaldo-Saint Blancard, K. Cho, S. Ho, and T. P. A. Collaboration, *Astroclip: a cross-modal foundation model for galaxies*, Monthly Notices of the Royal Astronomical Society **531** (06, 2024) 4990, https://academic.oup.com/mnras/article-pdf/531/4/4990/58325481/stae1450.pdf.

[26] T. Różański, Y.-S. Ting, and M. Jabłońska, *Toward a spectral foundation model: An attention-based approach with domain-inspired fine-tuning and wavelength parameterization*, arXiv:2306.15703 [astro-ph.IM].

[27] G. Zhang, T. Helfer, A. T. Gagliano, S. Mishra-Sharma, and V. A. Villar, *Maven: a multimodal foundation model for supernova science*, Mach. Learn. Sci. Tech. **5** (2024) 4, 045069, arXiv:2408.16829 [astro-ph.HE].

[28] M. J. Smith, R. J. Roberts, E. Angeloudi, and M. Huertas-Company, *Astropt: Scaling large observation models for astronomy*, arXiv:2405.14930 [astro-ph.IM].

[29] A. Ore, C. Heneka, and T. Plehn, *SKATR: A Self-Supervised Summary Transformer for SKA*, SciPost Phys. **18** (2025) 155, arXiv:2410.18899 [astro-ph.IM].

[30] B. M. Dillon, G. Kasieczka, H. Olischlager, T. Plehn, P. Sorrenson, and L. Vogel, *Symmetries, safety, and self-supervision*, SciPost Phys. **12** (2022) 6, 188, arXiv:2108.04253 [hep-ph].

[31] M. Vigl, N. Hartman, and L. Heinrich, *Finetuning foundation models for joint analysis optimization in High Energy Physics*, Mach. Learn. Sci. Tech. **5** (2024) 2, 025075, arXiv:2401.13536 [hep-ex].

[32] T. Golling, L. Heinrich, M. Kagan, S. Klein, M. Leigh, M. Osadchy, and J. A. Raine, *Masked particle modeling on sets: towards self-supervised high energy physics foundation models*, Mach. Learn. Sci. Tech. **5** (2024) 3, 035074, arXiv:2401.13537 [hep-ph].

[33] J. Birk, A. Hallin, and G. Kasieczka, *OmniJet-α: the first cross-task foundation model for particle physics*, Mach. Learn. Sci. Tech. **5** (2024) 3, 035031, arXiv:2403.05618 [hep-ph].

[34] P. Harris, J. Krupa, M. Kagan, B. Maier, and N. Woodward, *Resimulation-based self-supervised learning for pretraining physics foundation models*, Phys. Rev. D **111** (2025) 3, 032010, arXiv:2403.07066 [hep-ph].

[35] V. Mikuni and B. Nachman, *Solving key challenges in collider physics with foundation models*, Phys. Rev. D **111** (2025) 5, L051504, arXiv:2404.16091 [hep-ph].

[36] M. Leigh, S. Klein, F. Charton, T. Golling, L. Heinrich, M. Kagan, I. Ochoa, and M. Osadchy, *Is Tokenization Needed for Masked Particle Modelling?*, Mach. Learn. Sci. Tech. **6** (2025) 2, 025075, arXiv:2409.12589 [hep-ph].

[37] A. J. Wildridge, J. P. Rodgers, E. M. Colbert, Y. yao, A. W. Jung, and M. Liu, *Bumblebee: Foundation Model for Particle Physics Discovery*, in *38th conference on Neural Information Processing Systems*. 12, 2024. arXiv:2412.07867 [hep-ex].

[38] J. Bardhan, R. Agrawal, A. Tilak, C. Neeraj, and S. Mitra, *HEP-JEPA: A foundation model for collider physics using joint embedding predictive architecture*, arXiv:2502.03933 [cs.LG].

[39] B. M. Dillon, L. Favaro, F. Feiden, T. Modak, and T. Plehn, *Anomalies, representations, and self-supervision*, SciPost Phys. Core **7** (2024) 056, arXiv:2301.04660 [hep-ph].

[40] H. Qu, C. Li, and S. Qian, *Particle Transformer for Jet Tagging*, arXiv:2202.03772 [hep-ph].

[41] O. Amram, L. Anzalone, J. Birk, D. A. Faroughy, A. Hallin, G. Kasieczka, M. Krämer, I. Pang, H. Reyes-Gonzalez, and D. Shih, *Aspen Open Jets: unlocking LHC data for foundation models in particle physics*, Mach. Learn. Sci. Tech. **6** (2025) 3, 030601, arXiv:2412.10504 [hep-ph].

[42] CAMELS, F. Villaescusa-Navarro *et al.*, *The CAMELS project: Cosmology and Astrophysics with MachinE Learning Simulations*, Astrophys. J. **915** (2021) 71, arXiv:2010.00619 [astro-ph.CO].

[43] R. Meriot and B. Semelin, *The LORELI database: 21 cm signal inference with 3D radiative hydrodynamics simulations*, Astron. Astrophys. **683** (2024) A24, arXiv:2310.02684 [astro-ph.CO].

[44] C. Fanelli, J. Giroux, P. Moran, H. Nayak, K. Suresh, and E. Walter, *Physics event classification using Large Language Models*, JINST **19** (2024) 07, C07011, arXiv:2404.05752 [physics.data-an].

[45] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, *et al.*, *Qwen technical report*, arXiv:2309.16609 [cs.CL].

[46] A. Yang *et al.*, *Qwen2 technical report*, arXiv:2407.10671 [cs.CL].

[47] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, *et al.*, *Qwen2. 5 technical report*, arXiv:2412.15115 [cs.CL].

[48] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.*, *A survey of large language models*, arXiv:2303.18223 [cs.CL].

[49] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, *A comprehensive overview of large language models*, arXiv:2307.06435 [cs.CL].

[50] S. Minaee, T. Mikolov, N. Nikzad, M. A. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, *Large language models: A survey*, ArXiv **abs/2402.06196** (2024) .

[51] R. Sennrich, B. Haddow, and A. Birch, *Neural machine translation of rare words with subword units*, arXiv:1508.07909 [cs.CL].

[52] M. Schuster and K. Nakajima, *Japanese and korean voice search*, in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE. 2012.

[53] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, *Google's neural machine translation system: Bridging the gap between human and machine translation*, arXiv:1609.08144 [cs.CL].

[54] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, *Improving language understanding by generative pre-training*, .

[55] A. Butter, N. Huetsch, S. Palacios Schweitzer, T. Plehn, P. Sorrenson, and J. Spinner, *Jet diffusion versus JetGPT – Modern networks for the LHC*, SciPost Phys. Core **8** (2025) 026, arXiv:2305.10475 [hep-ph].

[56] T. Finke, M. Krämer, A. Mück, and J. Tönshoff, *Learning the language of QCD jets with transformers*, JHEP **06** (2023) 184, arXiv:2303.07364 [hep-ph].

[57] A. Butter, F. Charton, J. M. n. Villadamigo, A. Ore, T. Plehn, and J. Spinner, *Extrapolating Jet Radiation with Autoregressive Transformers*, arXiv:2412.12074 [hep-ph].

[58] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, *Training compute-optimal large language models*, arXiv:2203.15556 [cs.CL].

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, arXiv:1706.03762 [cs.CL].

[60] T. Plehn, A. Butter, B. Dillon, T. Heimel, C. Krause, and R. Winterhalder, *Modern Machine Learning for LHC Physicists*, arXiv:2211.01421 [hep-ph].

[61] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, *Gqa: Training generalized multi-query transformer models from multi-head checkpoints*, arXiv:2305.13245 [cs.CL].

[62] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, *Roformer: Enhanced transformer with rotary position embedding*, arXiv:2104.09864 [cs.CL].

[63] S. Chen, S. Wong, L. Chen, and Y. Tian, *Extending context window of large language models via positional interpolation*, arXiv:2306.15595 [cs.CL].

[64] B. Peng, J. Quesnelle, H. Fan, and E. Shippole, *Yarn: Efficient context window extension of large language models*, arXiv:2309.00071 [cs.CL].

[65] L. Zehui, P. Liu, L. Huang, J. Chen, X. Qiu, and X. Huang, *Dropattention: A regularization method for fully-connected self-attention networks*, arXiv:1907.11065 [cs.CL].

[66] B. Zhang and R. Sennrich, *Root mean square layer normalization*, arXiv:1910.07467 [cs.LG].

[67] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, *Language modeling with gated convolutional networks*, arXiv:1612.08083 [cs.CL].

[68] N. Shazeer, *Glu variants improve transformer*, arXiv:2002.05202 [cs.LG].

[69] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, arXiv:1512.03385 [cs.CV].

[70] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, and O. Levy, *Lima: Less is more for alignment*, arXiv:2305.11206 [cs.CL].

[71] DeepSeek-AI *et al.*, *Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning*, arXiv:2501.12948 [cs.CL].

[72] L. Ouyang *et al.*, *Training language models to follow instructions with human feedback*, arXiv:2203.02155 [cs.CL].

[73] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, arXiv:1707.06347 [cs.LG].

[74] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, *Direct preference optimization: Your language model is secretly a reward model*, arXiv:2305.18290 [cs.LG].

[75] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo, *Deepseekmath: Pushing the limits of mathematical reasoning in open language models*, arXiv:2402.03300 [cs.CL].

[76] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, *Lora: Low-rank adaptation of large language models*, arXiv:2106.09685 [cs.CL].

[77] B. Lester, R. Al-Rfou, and N. Constant, *The power of scale for parameter-efficient prompt tuning*, arXiv:2104.08691 [cs.CL].

[78] R. Nogueira, Z. Jiang, and J. Lin, *Investigating the limitations of transformers with simple arithmetic tasks*, arXiv:2102.13019 [cs.CL].

[79] Z. Yuan, H. Yuan, C. Tan, W. Wang, and S. Huang, *How well do large language models perform in arithmetic tasks?*, arXiv:2304.02015 [cs.CL].

[80] E. Tang, B. Yang, and X. Song, *Understanding llm embeddings for regression*, arXiv:2411.14708 [cs.LG].

[81] D. Maltoni and M. Ferrara, *Arithmetic with language models: From memorization to computation*, Neural Networks **179** (2024) 106550.

[82] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, *Large language models are zero-shot time series forecasters*, arXiv:2310.07820 [cs.LG].

[83] T. Liu and B. K. H. Low, *Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks*, arXiv:2305.14201 [cs.LG].

[84] N. Lee, K. Sreenivasan, J. D. Lee, K. Lee, and D. Papailiopoulos, *Teaching arithmetic to small transformers*, arXiv:2307.03381 [cs.LG].

[85] S. Yin, C. Fu, S. Zhao, K. Li, X. Sun, T. Xu, and E. Chen, *A survey on multimodal large language models*, National Science Review **11** (Nov., 2024) .

[86] P.-Y. Chen, *Model reprogramming: Resource-efficient cross-domain machine learning*, 2023. arXiv:2202.10629 [cs.LG].

[87] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen, *Time-llm: Time series forecasting by reprogramming large language models*, arXiv:2310.01728 [cs.LG].

[88] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, *One fits all:power general time series analysis by pretrained lm*, arXiv:2302.11939 [cs.LG].

[89] C. Chang, W. Peng, and T.-F. Chen, *Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters*, ACM Transactions on Intelligent Systems and Technology (2023) .

[90] J. Su, C. Jiang, X. Jin, Y. Qiao, T. Xiao, H. Ma, R. Wei, Z. Jing, J. Xu, and J. Lin, *Large language models for forecasting and anomaly detection: A systematic literature review*, arXiv:2402.10350 [cs.LG].

[91] A. Mesinger, S. Furlanetto, and R. Cen, *21cmfast: a fast, seminumerical simulation of the high-redshift 21-cm signal: 21cmfast*, Monthly Notices of the Royal Astronomical Society **411** (Nov., 2010) 955–972.

[92] S. G. Murray, B. Greig, A. Mesinger, J. B. Muñoz, Y. Qin, J. Park, and C. A. Watkinson, *21cmfast v3: A python-integrated c code for generating 3d realizations of the cosmic 21cm signal.*, Journal of Open Source Software **5** (2020) 54, 2582.

[93] Y. B. Zel'dovich, *Gravitational instability: An approximate theory for large density perturbations.*, A&A**5** (Mar., 1970) 84.

[94] Planck, N. Aghanim *et al.*, *Planck 2018 results. VI. Cosmological parameters*, Astron. Astrophys. **641** (2020) A6, arXiv:1807.06209 [astro-ph.CO]. [Erratum: Astron.Astrophys. 652, C4 (2021)].

[95] B. Villasenor, B. Robertson, P. Madau, and E. Schneider, *New constraints on warm dark matter from the Lyman-α forest power spectrum*, Phys. Rev. D**108** (July, 2023) 023502, arXiv:2209.14220 [astro-ph.CO].

[96] V. Iršič, M. Viel, M. G. Haehnelt, J. S. Bolton, M. Molaro, E. Puchwein, E. Boera, G. D. Becker, P. Gaikwad, L. C. Keating, and G. Kulkarni, *Unveiling Dark Matter free-streaming at the smallest scales with high redshift Lyman-alpha forest*, arXiv e-prints (Sept., 2023) arXiv:2309.04533, arXiv:2309.04533 [astro-ph.CO].

[97] B. Greig and A. Mesinger, *21CMMC with a 3D light-cone: the impact of the co-evolution approximation on the astrophysics of reionization and cosmic dawn*, Monthly Notices of the Royal Astronomical Society **477** (03, 2018) 3217, https://academic.oup.com/mnras/article-pdf/477/3/3217/24802887/sty796.pdf.

[98] N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. J. Banday, R. B. Barreiro, N. Bartolo, and et al., *Planck 2018 results*, Astronomy & Astrophysics **641** (Sep, 2020) A6.

[99] S. E. I. Bosman, F. B. Davies, G. D. Becker, L. C. Keating, R. L. Davies, Y. Zhu, A.-C. Eilers, V. D'Odorico, F. Bian, M. Bischetti, S. V. Cristiani, X. Fan, E. P. Farina, M. G. Haehnelt, J. F. Hennawi, G. Kulkarni, A. Mesinger, R. A. Meyer, M. Onoue, A. Pallottini, Y. Qin, E. Ryan-Weber, J.-T. Schindler, F. Walter, F. Wang, and J. Yang, *Hydrogen reionization ends by $z = 5.3$: Lyman-alpha optical depth measured by the xqr-30 sample*, Monthly Notices of the Royal Astronomical Society **514** (June, 2022) 55–76.

[100] B. Spina, S. E. I. Bosman, F. B. Davies, P. Gaikwad, and Y. Zhu, *Damping wings in the lyman-α forest: A model-independent measurement of the neutral fraction at* $5.4 < z < 6.1$, Astronomy & Astrophysics **688** (Aug., 2024) L26.

[101] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, *Flow matching for generative modeling*, arXiv:2210.02747 [cs.LG].