Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Aaron Johannes Nordmann

born in Darmstadt (Germany)

2024

SKATR: A Foundational Transformer Model for SKA Lightcones

This Bachelor Thesis has been carried out by Aaron Johannes Nordmann at the
Institut for Theoretical Physics in Heidelberg
under the supervision of
Prof. Dr. Tilman Plehn

Abstract

The upcoming Square Kilometre Array is set to deliver a three-dimensional tomographic view of hydrogen gas distribution mapping structure formation in the early universe. The complex nature of the underlying 21cm signal requires new analysis methods more sophisticated than traditional statistical methods. Lightcones (LCs) intended to model SKA data are simulated using 21cmFAST. Six key cosmological and astrophysical simulation parameters are varied in the LC dataset. Machine learning methods are employed to perform parameter regression on the LCs. A vision transformer (ViT) architecture is shown to outperform a previously top-performing convolutional neural network. The foundation model SKATR is developed for the LCs. It employs a ViT encoder architecture in an unsupervised pretraining schedule. In pretraining the encoder learns an embedding by contrasting a masked embedded view with the full LC embedding. SKATR is shown to retain a 144-dimensional informative summary of the LCs. In transferring to LCs with distinct qualities not seen in pretraining, the summary is able to learn realistic noise modeled after SKA detector influences by 21cmSense. Furthermore it is able to learn a simulation parameter not observed during training.

Zusammenfassung

Das kommende Square Kilometre Array soll eine dreidimensionale tomographische Ansicht der Wasserstoffgasverteilung liefern, die die Strukturbildung im frühen Universum abbildet. Die komplexe Natur des zugrundeliegenden 21cm-Signals erfordert neue Analysemethoden, die anspruchsvoller sind als traditionelle statistische Methoden. Lightcones (LCs), die SKA-Daten modellieren sollen, werden mit 21cmFAST simuliert. Sechs wichtige kosmologische und astrophysikalische Simulationsparameter werden in dem LC-Datensatz variiert. Methoden des maschinellen Lernens werden eingesetzt, um eine Parameterregression für die LCs durchzuführen. Es wird gezeigt, dass eine Vision Transformer (ViT) Architektur ein zuvor leistungsfähiges Faltungsneuronales Netzwerk übertrifft. Das Foundation Model SKATR wird für die LCs entwickelt. Es verwendet eine ViT-Encoder-Architektur und unbeaufsichtigtes Vortraining. Beim Vortraining lernt der Kodierer eine Einbettung, indem er die Gleichheit zwischen einer maskierten Einbettung mit der vollständigen LC-Einbettung maximiert. Es wird gezeigt, dass SKATR eine 144-dimensionale informative Zusammenfassung der LCs liefert. Bei der Übertragung auf LCs mit neuen Eigenschaften die im Vortraining nicht enthalten waren, ist die Zusammenfassung in der Lage, realistisches Rauschen zu erlernen, das nach realistischen Bedingungen des SKA von 21cmSense modelliert wurde. Außerdem ist sie in der Lage, einen Simulationsparameter zu erlernen, der im Vortraining nicht enthalten war.

Contents

1 Introduction

The Square Kilometre Array (SKA)* is an upcoming radio interferometer project designed to improve the sensitivity and survey speed of current radio telescopes by an order of magnitude [19]. This will give an unprecedented view of the early Universe and is anticipated to give new insights ranging from cosmological structure formation and galaxy evolution to modifications of gravity and dark energy, as well as dark matter [12].
The SKA will allow the study of the period Cosmic Dawn (CD), where the first galaxies and stars form, as well as the Epoch of Reionization (EoR), which marks the ionization of the intergalactic medium (IGM). Large-scale structural information during these eras is captured by the intensity of the redshifted 21cm line emitted in the forbidden spin-flip of hydrogen. While current telescopes such as LOFAR [9] and HERA [6] are limited to a statistical detection of the 21cm signal, the SKA will have a three-dimensional tomographic view of the 21cm signal and thus track hot gas distribution over the early evolution of the universe [12]. Both the complex non-Gaussian structure of the data and its size of many hundreds of petabytes [19] require the development of new signal analysis methods. Simulation-based inference is performed on low-dimensional statistics such as power spectra. While these statistics have the advantage of capturing physically interpretable structures, much of the information contained in the original data is lost. Modern machine learning methods offer considerable potential for improvement. Employing simulation-based inference based on SKA-inspired data simulations has been shown to bring significant improvements over traditional methods. In Ref. [14, 15] the use of Convolutional Neural Network (CNN) architectures and conditional invertible neural network (cINNs) on simulated data shows a robust inference of astrophysical and cosmological parameters.
Expanding on these works, we explore the deployment of a foundation model on SKA data. First, simulated SKA-like data in the form of light cones (LCs) is generated using 21cmFAST. The six simulation parameters varied in the simulation of the LCs pose the regression task for this work. To find a robust encoder architecture, a vision transformer (ViT) is explored and compared to the previously established CNN architecture [14]. We develop a foundation model termed SKA Transformer (SKATR) that retains an informative low-dimensional summary of the high-dimensional simulated LCs. This summary is probed on different datasets and varying training conditions.

*https://www.skatelescope.org

2 SKA Physics

Cosmic Dawn (CD), marking the formation of the universe's first luminous sources, and the Epoch of Reionization (EoR), when the earliest stars and galaxies ionized the surrounding intergalactic medium (IGM), represent crucial periods in cosmic history. By mapping the spin-flip transition of neutral hydrogen known as the 21cm line, we can generate 3D tomographic light cones of these formative billion years [14]. In addition to exploring the formation and evolution of early stars and galaxies, 21cm intensity mapping offers a promising method for investigating our cosmic concordance model at redshifts $z >> 6$. This technique can provide insights into the nature of dark energy, potential modifications to gravity, and the characteristics of dark matter (DM). Recently, radio interferometers have focused on detecting the redshifted 21cm radiation to explore the EoR and CD. The upcoming Square Kilometre Array (SKA) aims to deliver a 3D tomography through 21cm line emission.

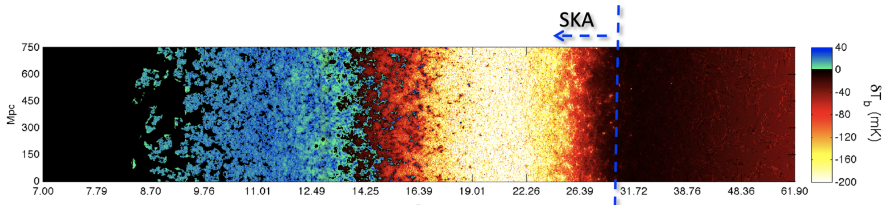

Figure 1: Scheme of the redshift eras observable by the SKA. Illustration taken from Ref. [12].

The 21cm signal serves as a cosmological probe by mapping the spatial distribution of neutral hydrogen and the ionization state of the intergalactic medium (IGM). This signal is typically expressed as the difference between the 21cm brightness temperature and the cosmic microwave background (CMB) temperature, T_γ, measured along a line of sight at a given observed frequency, ν [12], as shown in

$$\delta T(\nu) = \frac{T_S - T_\gamma}{1+z}(1 - e^{-\tau_{\nu_0}}). \tag{1}$$

3 Machine Learning

In recent times, large amounts of data and computational power have brought about the advent of Machine Learning (ML) in the natural sciences. By repeating what in practical terms is nothing more than matrix multiplication and on large amounts of data, the advent of ML has opened new possibilities and given way to insights previously difficult to attain.
In ML a model is a task and is iteratively improved depending on its performance. The performance of a model is measured by a metric referred to as a loss function. Minimizing this loss is equivalent to optimizing the model, which can be considered as the model "learning" the objective. The minimization of the loss is achieved through gradient descent. Gradient descent refers to an algorithm where the impact of model parameters on the loss is calculated by considering the output of the model or just taking the derivative. This gradient dictates to which extent a given weight is adjusted.

3.1 Learning Tasks

3.1.1 Supervised Learning

A supervised learning task is one where a model is asked to predict pre-defined labels on a dataset. The loss is defined based on the prediction of the model and the label of a given data point. Perhaps the simplest loss function to consider is the mean absolute error defined as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \,. \tag{2}$$

As it grows linearly with the difference between prediction and label, it corrects as much for small deviations as it does for large ones. This is often undesirable, as the statistical nature of data means that fitting it perfectly corresponds to fitting statistical variations. This is known as overfitting, which is a common and recurring issue in Machine Learning.
The Mean squared error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3}$$

counterfeits this behaviour by weighing large deviations stronger, allowing for statistical variation to persist.

3.1.2 Unsupervised Learning

In unsupervised learning, there is no assumed truth to predict. An example contrastive learning a set of augmentations are applied to the data. This results in different views of each instance in the data. The dissimilarity between different augmentations is minimized in order to learn the augmentations.
The set of augmentations may include transformations which the system is invariant under (i.e. rotations, reflections, translations). In training, this invariance is learned by the network. Furthermore, augmentations may reduce the informational content. Ideally, the network will learn to predict the missing information.

In unsupervised learning, there is no assumed truth to predict. Instead, the goal is to discover the underlying structure or patterns within the data without explicit labels. This is done

by exploring the relationships, similarities, and differences between the data points. An example for unsupervised learning is contrastive learning, where a set of augmentations is applied to the data. This results in different views of each instance in the data. The model is trained to minimize the dissimilarity between these different views of the same instance, effectively learning useful representations that capture the essential characteristics of the data. By doing so, the model learns to distinguish between different instances based on the underlying features that are consistent across various augmentations. This approach helps in learning robust feature representations without the need for labelled data, making it a powerful method in the realm of unsupervised learning.

3.2 Neural network

A network is employed to process some input of data into a desired output. To achieve this, the network must learn on a training set. It has a set of free parameters that are updated repeatedly in training until the network's output is optimal. The available data is generally divided into three splits, see Tab. 1.

split	purpose	typical size
training	find network parameters	60-80%
validation	track training performance	10-20%
test	test final performance	10-20%

Table 1: Data splits used in the different stages of a ML network employment.

In training, one aims to find a global minimum in the loss function with respect to the parameters. Training is performed over a set of epochs. In one epoch, the network sees the whole training set once. Throughout each epoch, the entire training set is passed through the network in batches, with the loss being aggregated across these batches. At the end of an epoch, the parameters are updated through gradient descent. The parameter j at epoch $i+1$ is updated to

$$\theta_j^{i+1} = \theta_j^i - \eta \frac{\partial L}{\partial \theta_j^i} \tag{4}$$

in terms of the parameter of the previous epoch θ_j^i and the gradient in respect to the loss L. The learning rate η dictates the degree to which the parameters are updated and is a hyperparameter of training. A learning rate that is too large may overshoot an optimum set of parameters, while one that is too small may get stuck in local minima. The gradients are calculated using backpropagation, which uses the chain rule to propagate the effect of a given parameter on the loss backward through the network.

3.2.1 Multi Layer Perceptron

A Multi Layer Perceptron (MLP) is perhaps the most basic architecture in ML. It consists of multiple layers, which each contain a set of neurons. Each neuron of a layer is connected to all neurons of the previous layer. A scheme of the data flow is shown in Fig. 2. The first layer is given by a data instance and the last layer is the network output. The layers in between contain the learned structure and are referred to as 'hidden layers'. The neurons encode information as scalars. The value of a neuron j in a layer i is given by a weighted aggregation of the previous neurons

$$x_j^i = \sum_k f^i(w_{kj}^i x_k^{i-1}), \tag{5}$$

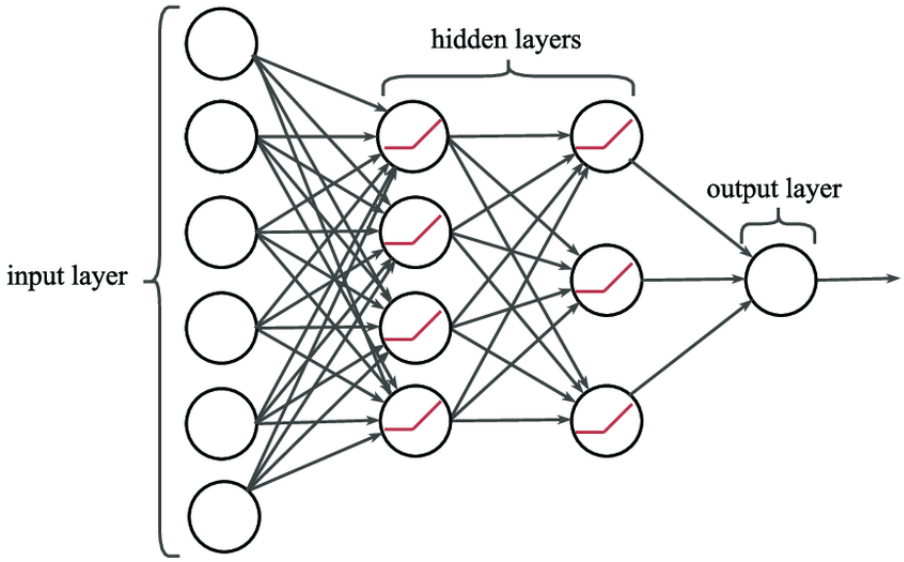

Figure 2: A dedication of the structure of an MLP.
The MLP consists of an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the subsequent layer, forming a fully connected network. The neurons apply a weighted sum of their inputs followed by an activation function, as indicated by the red lines within the neurons. This introduces non-linearity into the model. The hidden layers capture the learned features and representations, while the output layer produces the final prediction. Illustration taken from Ref. [16].

where the weights w^i_{kj} dictate the influence of the k-th neuron of the previous layer. These weights are the model's free parameters and are learned during training. Between layers, an activation function f_i may act. Activation functions introduce non-linearities into the model, allowing it to learn more complex representations. The most typical activation function is the rectified linear unit (ReLU) given by

$$f(x) = \max(0, x) \tag{6}$$

The ReLu sets all negative input values to zero while leaving positive values unchanged. This has the effect of creating a sparse representation, as only some neurons are activated (i.e., have non-zero outputs) for any given input. Other commonly used activation functions include the sigmoid function, given by

$$\text{Sigmoid}(z) = \frac{1}{1 + e^{-z}}, \tag{7}$$

which squashes the input into a range between 0 and 1. The softmax function converts a vector into probabilities that sum to one as

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{8}$$

where: z_i is the i-th element of the input vector $\mathbf{z}$, K is the number of elements in the vector $\mathbf{z}$. Depending on the desired structure, different activation functions are employed. has different properties and can be chosen based on the specific requirements of the neural network model.

3.2.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a type of neural network architecture designed to process and analyse data with a grid-like topology, such as images. Unlike the fully connected layers in an MLP, CNNs use convolutional layers to extract local patterns in the data. A scheme of the architecture is seen in Fig. 3.

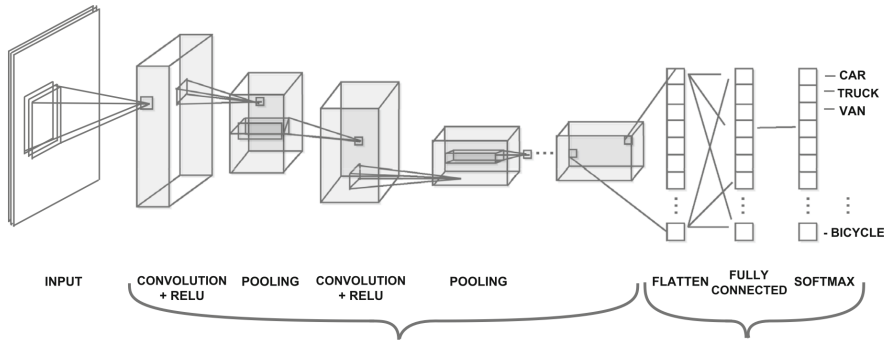

Figure 3: This figure illustrates the general architecture of a Convolutional Neural Network (CNN). The input is typically an image, represented as a multichannel matrix. The network consists of a sequence of layers, beginning with convolutional layers, which apply a set of filters to extract feature maps from the input. Illustration taken from Ref. [16].

The fundamental building blocks of a CNN are convolutional layers, pooling layers, and fully connected layers. In a convolutional layer, neurons are arranged in feature maps, and each neuron is connected only to a local region of the previous layer. This local connection allows CNNs to capture spatial structure in the data, where lower layers detect basic features like edges, and higher layers detect more complex patterns like shapes or objects. The operation performed in a convolutional layer is defined as

$$x_j^i = f^i\left(\sum_k w_{kj}^i * x_k^{i-1}\right), \tag{9}$$

where $*$ denotes the convolution operation, w_{kj}^i are the filter weights, and f^i is the activation function. The filters, also known as kernels, slide over the input feature maps to produce output feature maps, capturing essential input features. Pooling layers are typically inserted between convolutional layers to progressively reduce the spatial dimensions of the data, which helps to decrease the computational load and reduce the risk of overfitting the data. The most common type of pooling is max-pooling, which selects the maximum value from a patch of the feature map, as defined by

$$x_j^i = \max(x_{m,n}^{i-1}), \tag{10}$$

where m and n index the local patch in the feature map. This operation retains the most prominent features while discarding less significant information. After several convolutional and pooling layers, the final output is often passed through one or more fully connected layers, similar to an MLP. In a 3D CNN, the convolutional layers apply 3D kernels to the input data, enabling the network to capture spatial and temporal features simultaneously. The pooling operations in 3D CNNs similarly reduce the dimensionality across all three axes.

3.2.3 Vision Transformer

A Vision Transformer (ViT) is a transformer architecture applied to two or higher-dimensional data, such as images. Originally introduced in the context of natural language processing (NLP) through the seminal work "Attention is All You Need" [17], transformers have become a dominant architecture in Natural Language Processing (NLP) due to their ability to model long-range dependencies effectively and their computational efficiency. The success of transformers in NLP led to their adaptation for visual data processing, giving rise to the ViT. A scheme of the ViT is shown in Fig. 4.

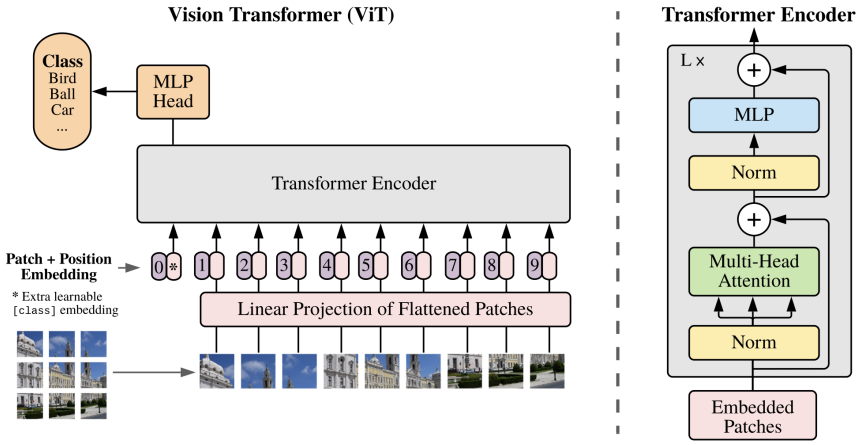

Figure 4: Scheme of the Vision Transformer for image classification. The input image is split into patches, flattened, and projected into embeddings, combined with position encodings. These embeddings pass through a series of transformer encoders, where self-attention captures global dependencies. The output from the token is processed by an MLP head. Illustration taken from Ref. [7].

In a ViT, the input image is first divided into a grid of patches, which are then flattened into vectors. Each vector is treated as a token, similar to words in a sentence for NLP tasks. These tokens are embedded and augmented with positional encodings to retain spatial information. The resulting sequence of token embeddings is fed into a standard transformer model, where the key innovation lies in the self-attention mechanism. This mechanism enables the model to weigh the importance of different tokens relative to each other, effectively capturing global context and dependencies within the image. The self-attention mechanism in a ViT operates by computing a weighted sum of values, where the weights are determined by the similarity between queries and keys. Mathematically, for a given token i, the attention score for token j is calculated as the dot product of the query Q_i and key K_j, scaled by the square root of the dimensionality of the key vectors, and then passed through a softmax function

$$\text{Attention}(Q_i, K_j, V_j) = \text{softmax}\left(\frac{Q_i K_j^\top}{\sqrt{d_k}}\right) V_j$$

where the query $Q_i = W_Q x_i$, key $K_j = W_K x_j$, and value $V_j = W_V x_j$ vectors are computed from the input token embeddings x_i and x_j using the learned weight matrices W_Q, W_K, and W_V, respectively. The self-attention mechanism in a ViT operates in parallel across all tokens. This allows for parallelization and thus computational efficiency. Unlike CNNs, which typically focus on local features by convolutions, ViTs can model long-range interactions. Another

key advantage of ViTs is their scalability. As transformers are known to perform better with increasing data sizes[17], ViTs make use of large datasets to learn richer representations. The generalization to three dimensions is straightforward. In a 3D ViT, the input data is divided into 3D patches or cubes instead of 2D patches, allowing the model to process. These 3D patches are flattened and embedded like before.

3.3 Foundation Model

A foundation model uses unsupervised training to learn the general structure expressed by a dataset. It is typically trained over many epochs to attain many views of the data. This training is called pretraining. The foundation model is then used to train on some "downstream" task called. In pretraining, an unsupervised training is employed to attain an informative representation. Contrastive learning is used to maximise the similarity between different equivalent views of the data. Augmentations are applied that reflect symmetries the data is intended to reflect. In the process of maximising the similarity between these different views, the augmentations are learned and lead to an encoding informative of these. Another point would be shareability which is related to data efficiency. If I want to solve a language task, it is easier for one to download a pretrained model than to train on 1 trillion words myself.

3.3.1 Use cases

Foundation models can serve several purposes. In compressing the data, the model reduces the dimensionality, aiming to retain all information present. During pretraining, the model learns to capture the underlying structure this allows for efficient data handling, especially in the context of interpretability. Foundation models need fewer examples of the data in downstream training, as it has already seen many views in pretraining. By pretraining on a large dataset, the model develops a generalized understanding of the data, which can be adapted to new, related tasks with much smaller datasets. This also enables transfer learning, where new aspects may be picked up in downstream training that weren't present in a distinct pretraining dataset. As fewer views of the data are necessary, the computational efficiency is expected to be higher than training from scratch. The representations yielded by a foundation model are expected to be more regularized. This results in a greater training stability and generally makes downstream training less sensitive to hyperparameter tuning. In general, foundation models leverage large data volumes to achieve higher performance in downstream tasks. The pretraining process allows the model to learn a broad range of features and nuances from the data, which can then be fine-tuned to excel in specific applications. Foundation models promote shareability and reuse across different applications. Since pretraining requires substantial resources, sharing pretrained models enables others to build upon them without the need to start from scratch.

3.3.2 Operation Modes

Foundation models can operate in several modes. The relevant modes will be the “Free” and the “Summary” mode.

In the free mode, the model weights obtained during pretraining are left trainable during the fine-tuning phase on the downstream task. This approach allows for maximum flexibility, enabling the model to adapt extensively to the new data and task-specific requirements. By allowing all the weights to be adjusted, the model can potentially achieve superior performance, as it can leverage the full capacity of its learned representations to optimize for the new task. Fine-tuning in Free mode is computationally expensive and may require significant

resources. Additionally, this is less informative in terms of understanding the specific contributions of different components of the pretrained model, as all parameters are subject to change and representations are high-dimensional. The Summary mode focuses on generating a compressed, informative summary of the pretrained model's weights. This mode involves computing a summary vector of minimal dimensionality, designed to be maximally informative about the pretraining dataset. Typically, this summary vector is derived from the final state of the model and may involve techniques such as averaging over model weights or attention layers to reduce dimensionality while retaining critical information. The resulting summary vector serves as a condensed representation of the pretrained knowledge, which can then be used for downstream tasks with minimal additional training.

4 Data

4.1 Simulated lightcones

Dataset	# LCs	Sim. res. [Mpc]	DS	Shape [voxels]	LC size [MB]	Total size [GB]
HR_{x1}	5038	1.42	None	[140, 140, 2350]	176	885
HR_{x2}	5038	1.42	x2	[70, 70, 1175]	22.0	110
HR_{x5}	5038	1.42	x5	[28, 28, 470]	1.41	7.10
HR^{noise}_{x1}	5012	1.42	None	[140, 140, 2350]	351	1,761
HR^{noise}_{x2}	5012	1.42	x2	[70, 70, 1175]	43.9	220
HR^{noise}_{x5}	5012	1.42	x5	[28, 28, 470]	2.81	14
LR_{x2}	34741	2.84	x2	[70, 70, 1175]	22.0	763
LR_{x5}	34741	2.84	x5	[28, 28, 470]	1.41	49

Table 2: Overview of the available datasets.
HR $\hat{=}$ high resolution, LR $\hat{=}$ low resolution, DS $\hat{=}$ downsampling.
Dataset subscripts indicate the DS factor.

The database used in this work is given by a set of lightcones (LCs). An LC is a 3D object composed of voxels that represent the $21cm$ brightness temperature fluctuations $\delta T_b(x, y, z)$ with on-sky coordinates x and y, and redshift z as introduced in Sect. 2.

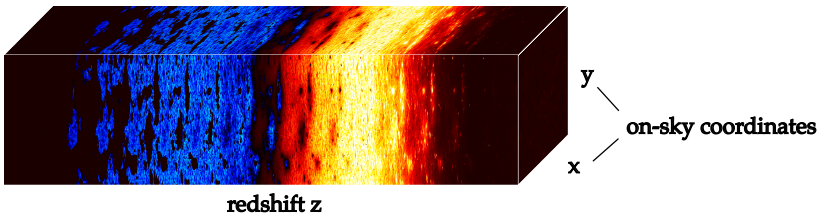

Figure 5: Example of a LC as generated with 21cmFAST. Illustration adapted from Ref. [14].

An illustration is given in Fig. 5. The LCs are simulated using version 3 [13] of 21cmFAST [11] code[†]. This seminumerical code is comparable to full hydrodynamic simulations. It models the scales of low-frequency radio telescopes quite well, while being significantly more efficient to run at the same time.
21cmFAST generates LCs by evolving coeval cubes with redshift as follows. Initial perturbations are sampled using Gaussian random fields that correspond to the matter distribution in the early universe. Thus, obtained density and velocity fields are evolved using first- and second-order perturbation theory based on the Zel'dovich approximation [20], which is sufficient for large-scale structure formation. Collapsed regions of dark matter halos within the evolved density field are identified. The ionization state of hydrogen is found by tracking the ionizing photons produced by galaxies formed within these dark matter halos. To find ionized regions, the number of ionizing photons produced by galaxies within these halos is calculated and propagated through the inter galactic medium (IGM). Given the density and ionization

[†]https://github.com/21cmFAST/21cmFAST

state fields along with the spin temperature of hydrogen dependent on gas temperature, the differential brightness temperature of the 21 cm signal is computed[19].
Two sets of LCs generated on two different simulation resolutions are available, as summarized in Tab. 2. The high resolution (HR) dataset HR was created prior to this work [14] and the low resolution (LR) dataset was simulated as part of this work. The subtext for each dataset indicates the resolution downsampling (DS) factor r. In DS a set of LC voxels is summarized into a single voxel. The light cone dimensions are thus reduced by r in all three dimensions. Here r is relative to the maximum resolution dataset HR ($\mathrm{LR}_{\mathrm{x2}}$ is not downsampled as it was already simulated at half the resolution of HR, while $\mathrm{LR}_{\mathrm{x5}}$ is its downsampled variant with a DS factor of 2.5). The dataset HR and its noised variant $\mathrm{HR}^{\mathrm{noise}}$ were taken from Ref. [14] while the $\mathrm{LR}_{\mathrm{x2}}$ dataset was simulated using a wrapper for 21cmFASTv3[‡]. The wrapper utilizes CPU parallelization along with other features that increase computational efficiency and speed. As introduced in Sect. 3.2.3 and Sect. 3.3 both ViTs and foundation models are expected to scale well with data size. Thus, the low-resolution dataset LR was created to expand upon the existing data volume. As later shown in Sect. 6.1.1 the lower simulation resolution leads to LR being uninformative of the warm darm matter (WDM) mass m_{WDM} which structurally sets $\mathrm{LR}_{\mathrm{x2}}/\mathrm{LR}_{\mathrm{x5}}$ apart from $\mathrm{HR}_{\mathrm{x2}}/\mathrm{HR}_{\mathrm{x5}}$.
In 21cmFAST, a plethora of simulation parameters are used. A subset of six free simulation parameters is sampled evenly across a range, ensuring a flat prior distribution. Parameter sampling ranges as well as the fixed simulation parameters are set in agreement with the simulation configuration of HR [14]. This configuration assumes a cosmological constant and flatness, setting the fixed simulation parameters to Planck measurements [1].
The six free simulation parameters include four astrophysical and two dark matter parameters and are sampled as follows:

- Warm dark matter (WDM) mass $m_{\mathrm{WDM}} \in [0.3, 10]\,\mathrm{keV}$
  Current constraints point towards values $> 3.1keV$ [18]. The relatively wide limits allow for a large variability in LC behaviour. Towards the upper limit, the WDM free-streaming scale approaches that of cold dark matter (CDM) and structure formation exhibits behaviour akin to CDM;
- Dark matter density parameter $\Omega_{\mathrm{m}} \in [0.2, 0.4]$
  Strongly controls structure formation. A relatively wide range is chosen, most of which pertains to Planck limits [1];
- Minimum virial temperature $T_{\mathrm{vir}} \in [10^4, 10^{5.3}]\,\mathrm{K}$
  Sets the minimum virial temperature of halos contributing to star formation. The lower limit is needed for sufficiently efficient atomic cooling;
- Ionization efficiency $\zeta \in [10, 250]$
  A composite parameter proportional to the
  - fraction of ionizing photons escaping a galaxy;
  - fraction of galactic gas contained in stars;
  - number of ionizing photons per baryon in stars;

  and dependent on the typical number density of recombinations for hydrogen in the IGM. The range allows for a diverse range of recombination scenarios;
- Specific X-ray luminosity $L_{\mathrm{X}} \in [10^{38}, 10^{42}]\,\mathrm{erg\,s^{-1}\,M_{\odot}^{-1}\,yr}$
  Integrated luminosity with energy $< 2\,\mathrm{keV}$ per unit star formation rate escaping host galaxies. Range contains observational and simulation limits [8];
- X-ray energy threshold for self-absorption by host galaxies $E_0 \in [100, 1500]\,\mathrm{eV}$
  X-rays of energies below E_0 do not escape their host galaxy. The range is motivated by

[‡]https://github.com/astro-ML/21cm-wrapper

simulations of high-redshift galaxies [10];

The LC size is $200Mpc$ and the redshift range is $z \in [5, 35]$. The number of voxels in the redshift dimension is dependent on Ω_m. Voxels in excess of the redshift limit given by the voxel dimensions seen in Tab. 2 are cut off.
As performed in Ref. [14], SKA mock observational LCs HR^{noise} are attained by adding noise to HR. Depending on the redshift, different thermal noise levels are calculated using the optimistic noise scenario given by 21cmSense[§] and applied. Thermal noise aims to reflect noise observed in the integrated SKA-Low stage 1 observations [4].

4.2 Preprocessing

Voxel values and simulation parameter labels of each LC are shifted and normalized to lie within the unit range $[0, 1]$. At each training epoch, random transformations are applied, that reflect the LC symmetries. These include:

- 90° rotations in the spatial coordinates;
- Flip in the spatial coordinates;

ensuring that they do not compose the identity, so we can keep the identity in the preprocessing augmentations.

[§]https://github.com/jpober/21cmSense

5 Models

5.1 SKATR

We propose SKATR, a foundational transformer model designed for use on SKA data. Employing an unsupervised training framework [2], SKATR uses a ViT encoder architecture to attain an informative compressed representation in the form of a 144 dimensional summary vector.

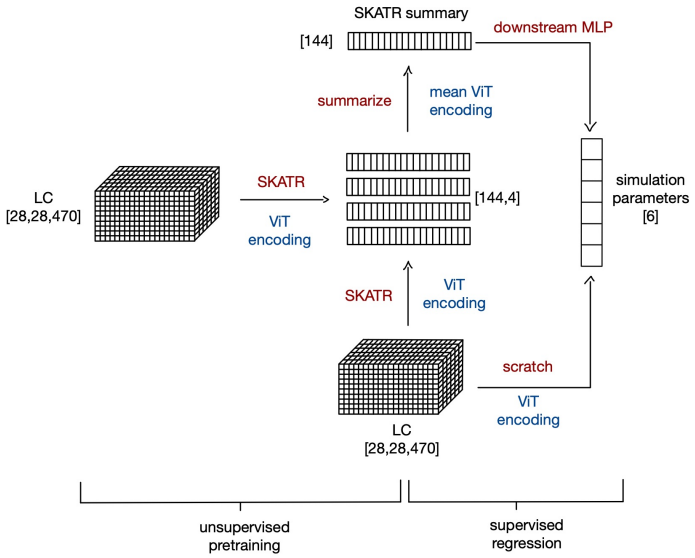

Figure 6: SKATR dataflow scheme.
In pretraining, SKATR attains an encoding through its pretraining schedule. In downstream regression, a frozen weight forward pass is performed on the regression training LC, and encoder ViT embedding is calculated. An average of the attention heads yields the summary vector. Upon this summary, an MLP is trained to perform regression. In the case of scratch regression, an independent ViT is trained using the same MLP structure to regress the six simulation parameters.

An overview of SKATR dataflow is shown in Fig. 6.

5.1.1 Model architecture

The goal of pretraining is to learn an encoding f_θ that produces informative representations of the data without using any labels. The LCs pertain to a high-dimensional voxel space that is expected to be highly compressible. We explore encoders that map to a low-dimensional embedding space.

Different pretraining schedules are tried. The most rudimentary schedule we investigated is SimSiam **??**. It combines ideas from previous work in computer vision **??** into a uniquely simple architecture. The largest problem to overcome is representation collapse in contrastive learning. SimSiam solves this by employing two encoder networks on different views of the data, where only one is trained. From this encoder, a smaller predictor maximizes the similarity

between the two encodings. This preatraining schedule was ultimately abandoned as we did not see successful downstream parameter regression.

To train the encoder, we adopt JEPA, a self-supervised learning framework developed in unsupervised image [2] and more recently video [3] learning applications.

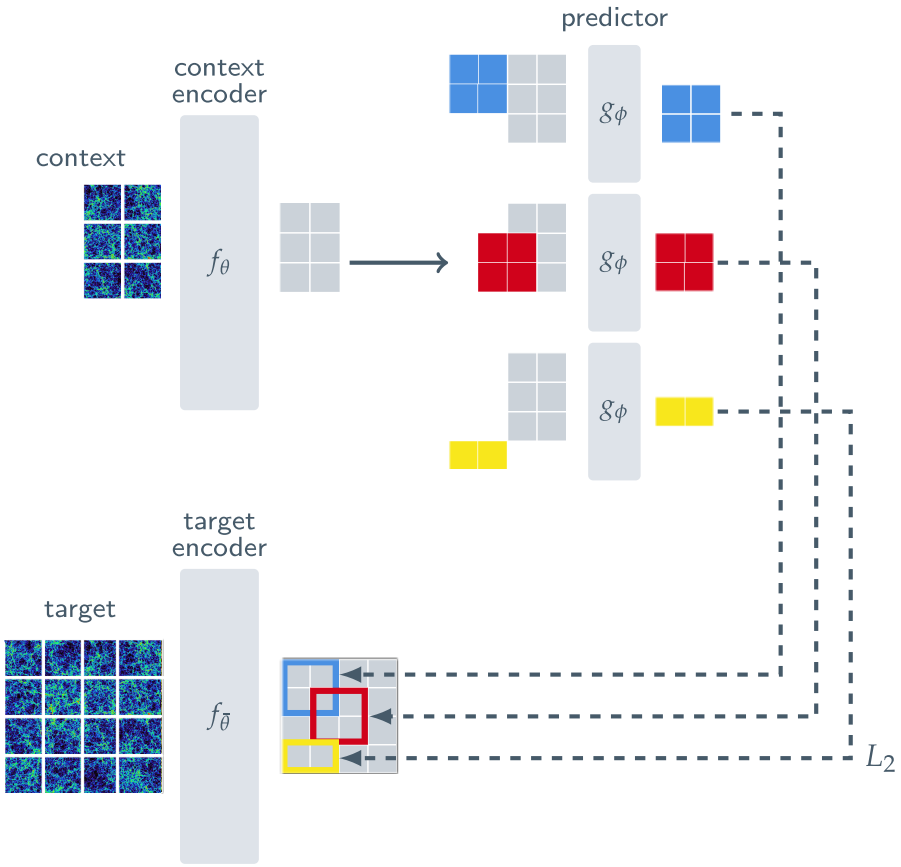

Figure 7: Scheme of SKATR pretraining and downstream simulation parameter regression.

A scheme of SKATR pretraining is shown in Fig. 7. The setup involves two ViTs with identical architectures: a context encoder f_θ and a target encoder g_ϕ. In training, a LC (for simplicity the batch is omitted) is divided into a set of N patches $x = \{x_i\}_{i=1}^N$ and a masked view $\tilde{x}$ is generated by sampling a set of patch locations M to be dropped yielding

$$\tilde{x} = \{x_i \in x \,|\, i \notin M\}\,. \tag{11}$$

The masked and original LCs are embedded as context $\tilde{z}$ and target z representations by the context encoder f_θ and target encoder g_ϕ respectively.

$$\tilde{z} = f_\theta(\tilde{x}), \qquad z = g_\phi(x)\,. \tag{12}$$

Given the context representation $\tilde{z}$, another transformer h_ψ predicts the target representation

$$p = h_\psi(\tilde{z})\,. \tag{13}$$

The above prediction is repeated four times for four different samples of masked views $\tilde{x}$. The loss function is the mean absolute error between p and z at the locations of the masked

patches. In full, this reads

$$\mathcal{L}_{\text{pretrain}} = \left\langle \frac{1}{|M|} \sum_{i \in M} \left| g^i_\phi(x) - h^i_\psi(f_\theta(\tilde{x})) \right| \right\rangle_{p_{\text{data}}(x),\, p_{\text{transform}}(\tilde{x}\,|\,x)} \tag{14}$$

where $p_{\text{transform}}(\tilde{x}\,|\,x)$ is a distribution over possible masks of x. Given that the above loss is not contrastive, there is a risk of representation collapse if $\mathcal{L}_{\text{pretrain}}$ is optimized with respect to all sets of parameters θ, ϕ, ψ [5]. To avoid this, only the context encoder and predictor parameters are updated via gradient descent. The target encoder parameters ϕ instead follow the exponential moving average of the context encoder parameters θ,

$$\phi_{i+1} = \tau \phi_i + (1-\tau)\, \theta_i\,, \tag{15}$$

where τ is a momentum hyperparameter controlling the rate at which the target encoder follows the context encoder.

To get the masked views $\tilde{x}$ from the full LC patches x, multi-block patch masking is applied. Out of the four predictions made per LC, two of the masked views are sampled from long-range and two from short-range masks, following the general structure introduced in JEPA [3]. The patches that compose a view $x = \{x_i\}_{i=1}^N$ correspond to the three-dimensional structure of the LC. The masks are a union of N_{blocks} three-dimensional patch blocks in embedding space. One set of block dimensions is sampled for a number of blocks according to the sampling mode: The number of blocks and

- short-range: 8 blocks with spatial fraction $\in [0.1, 0.2]$
- long-range: 2 blocks with spatial fraction $\in [0.6, 0.8]$

where the spatial fraction gives the fraction of patches to mask in the spatial dimensions. The redshift dimension is masked entirely for both modes. The blocks pertain to a spatial aspect ratio $\in [0.75, 1.5]$.
The masked patches are all set to the same value, which is initialized randomly and is a free parameter during training. This allows the network to learn to distinguish masked patches from informative ones by separating them in the embedding space.

5.1.2 Model training

Three foundation models were trained on 33250, 26000 and 18750 LCs of LR_{x5}. In downstream regression the performance of these was nearly the same. As the model trained on 18750 retained the best regression results by a small margin, it was employed through. This model was trained over 1000 epochs on the h100 GPU.The training took almost 5 days.

5.2 Architectures

5.2.1 ViT architectures

We use a ViT architecture for the encoders f_θ, g_ϕ and h_ψ using the x5 adaptation. The hyperparameters settled on are as follows. For the x2 variant patch dimensions of $[7, 7, 25]$ are used and for the x5 variant $[4, 4, 10]$. Regression performance is observed to be sensitive to changes in the patch size. While the x2 variant uses a hidden dimension of 96, the x5 variant uses a larger space of 144. The lower choice for the x2 variant is significant, as for higher dimensions regression becomes unstable. The architecture depth is 2 for x2 and 4 for x5. The depth did not have a significant influence on performance. The number of attention heads if 4 for both variants.

5.2.2 CNN architectures

There are two architectures x2 and x5. The layer structure is seen in Fig. 8

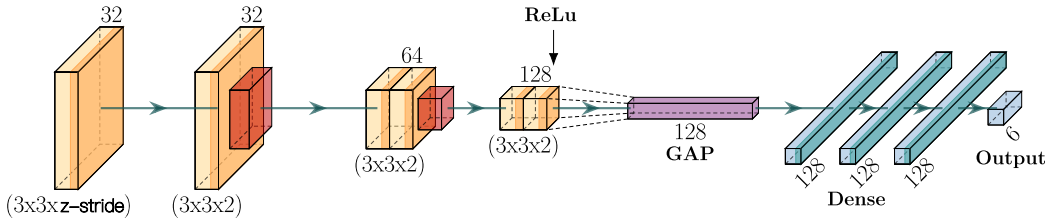

Figure 8: Schematic representation of the CNN architecture (adapted from Ref. [14]). Layers of the network are shown in orange and red. The z-stride is given by the x2 and x5 architectures. The fourth dimension corresponding to the filter is not shown. Instead, the number of filters is indicated above each layer. The Global Average Pooling (GAP) is shown in purple.

The x2 variant summarized in Fig. 3 was robustly tested and optimized.

Layer	Shape
Input Layer	(70, 70, 1175, 1)
3x3x51 Conv3D	(68, 68, 23, 32)
3x3x2 Conv3D	(66, 66, 22, 32)
2x2x1 Max Pooling	(33, 33, 22, 32)
3x3x2 Conv3D	(31, 31, 21, 64)
1x1x0 Zero Padding	(33, 33, 21, 64)
3x3x2 Conv3D	(31, 31, 20, 64)
2x2x1 Max Pooling	(15, 15, 20, 64)
3x3x2 Conv3D	(13, 13, 19, 128)
1x1x0 Zero Padding	(15, 15, 19, 128)
3x3x2 Conv3D	(13, 13, 18, 128)
Global Average Pooling	(128)
3x Dense with ReLu activations	(128)
Dense	(6)
Trainable Parameters: 636,838	

Table 3: x2 CNN Architecture; see Fig. 8 for a schematic overview.

The x5 variant summarized in Fig. **??** was adapted only for data scaling studies. Hyperparameters were not explored. Only the z-stride was matched to match the dimension reduction of the x2 variant in the first layer.

5.3 Simulation parameter regression

The simulation parameter regression is performed on an MSE loss. The data splits for the for HR datasets are 75% training 10% validation and 15% testing. In the case of LR a dedicated test set of 1000 LCs was used. The remaining LCs are split according to 75% training 10% validation. In applying the foundation model to regression, the different operation modes intro-

Layer	Shape
Input Layer	(28, 28, 470, 1)
3x3x20 Conv3D	(26, 26, 23, 32)
3x3x2 Conv3D	(24, 24, 22, 32)
2x2x1 Max Pooling	(12, 12, 22, 32)
3x3x2 Conv3D	(10, 10, 21, 64)
1x1x0 Zero Padding	(12, 12, 21, 64)
3x3x2 Conv3D	(10, 10, 21, 64)
2x2x1 Max Pooling	(5, 5, 21, 64)
3x3x2 Conv3D	(3, 3, 20, 128)
1x1x0 Zero Padding	(5, 5, 20, 128)
3x3x2 Conv3D	(3, 3, 18, 128)
Global Average Pooling	(128)
3x Dense with ReLu activations	(128)
Dense	(6)
Trainable Parameters: 627,910	

Table 4: x5 CNN Architecture; see Fig. 8 for a schematic overview.

duced in 3.3.2 are applicable. On early tests, we find that the free mode yields no performance gain over the summary mode on regression. As the summary mode is both computationally cheaper and more informative, it is used throughout. The summary vector is computed as a mean over the context encoder f_θ attention heads to be of dimension of 144. In downstream regression, this summary vector is calculated once for each LC with no training on the backbone. A fully connected MLP of shape $[144, 64, 64, 6]$ is trained upon this summary to regress the simulation parameters. Training is performed until the validation loss has not improved for 40 epochs. Typical training times on an a30 GPU are 3 hours.

6 Results

6.1 Data studies

To get a general overview of the data, the two datasets HR and LR_{x2} 4.1 and their relevant downsamples are studied. As a measure of the information content about the six simulation parameters contained in the datasets, the ViT architectures 5.2.1 perform parameter regression. Training is performed as described in Sect. 5.3.

6.1.1 Dataset comparison

To investigate the effect of reducing the simulation resolution for the newly generated dataset LR_{x2}, it is compared against HR_{x2}.

The six panels in Fig. 9 show the network predictions as a function of the true simulation parameters. For better readability and to gain a measure for network uncertainty, the parameter values are binned along the x-axis. The mean and standard deviation of the network predictions in these binned intervals are displayed. The mean and deviation of the two networks in each bin are offset slightly from one another to be able to distinguish them visually. The black diagonals indicate the ideal prediction curve. The subpanels show the mean absolute relative error

$$MARE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\text{truth}_i - \text{pred}_i}{\text{truth}_i}\right| \tag{16}$$

of predictions to the true parameters in each bin containing n samples. This error is asymmetrical, as observed in the $m_W DM$ panel. The horizontal dashed lines indicate the average of this error and quantify the performance in the regression of each parameter.
On most parameters, the regression on LR_{x2} yields a more accurate prediction. This is reflected in the lower mean MARE as well as a lower prediction variance. This may be explained by the larger data size of LR_{x2} which improves network training. A behaviour present in the regression of all parameters is the 'prior window' effect. A limited prior window given by the sampling ranges of simulation parameters 4.1 leads to a biased prediction at the borders of these windows. This results in the characteristic s-shape of the network predictions, where parameter values at the low ends are overpredicted and values at the high end are underpredicted. Depending on the parameter, different network behaviours are apparent, ranging from an almost perfect regression to the presence of a significant fraction of outliers along with degeneracies.

- Warm dark matter (WDM) mass m_{WDM}
  In HR_{x2}, the network learns to regress m_{WDM} to some degree for values $\lesssim 3$. In LR_{x2}, the network shows that it cannot learn this relationship because it provides a constant prediction, regardless of the true simulation m_{WDM}.Since it is learnt on dataHRx2 but not on LR_{x2}, the lower simulation resolution of LR means that it becomes uninformative for m_{WDM}. This may be due to m_{WDM} being sensitive to small frequency scales [15] that aren't resolved in the lower resolution simulations. Removing m_{WDM} along with Ω_m from the regression loss has been shown to improve the regression of the remaining parameters [14]. As m_{WDM} contributes as a constant offset to the regression loss, the network effectively has one less parameter to learn and is not forced to adjust its weights to compromise on other parameters. This is also expected to contribute to the overall better regression on LRx2.
- Dark matter density parameter Ω_{m}; Specific X-ray luminosity L_{X}; Ionization efficiency ζ
  Both datasets yield an accurate prediction in all parameters. The truth parameter values mostly lie within network deviations. On LR_{x2}, predictions are both more accurate and

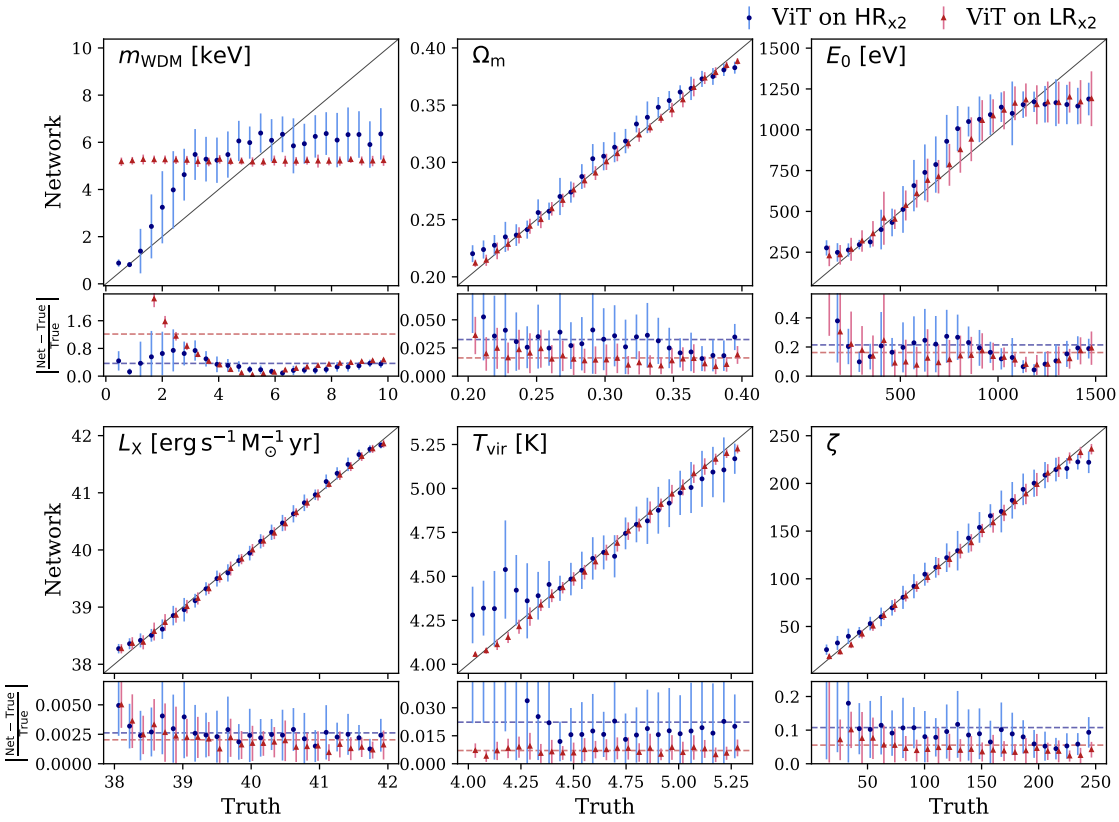

Figure 9: Comparing HR$_{x2}$ (blue circles) to the lower simulation resolution dataset LR$_{x2}$ (red triangles) by scratch ViT simulation parameter regression.
The six panels show the network predictions as a function of the true simulation parameters. Parameter values are binned along the x-axis. The mean and standard deviation in each bin are displayed. The black diagonals indicate the ideal prediction curve. The subpanels show the mean absolute relative error in each bin. The coloured dashed lines indicate the mean in this error.
The warm dark matter (WDM) mass m_{WDM} is only predicted in HR$_{x2}$. The dataset LR$_{x2}$ is thus regarded to be uninformative of m_{WDM}. The predictions on the other parameters are more accurate on LR$_{x2}$. The information lost due to the lower simulation resolution of LR$_{x2}$ seems to be outweighed by the greater data size, as well as not needing to regress m_{WDM}. The loss of small-scale information may help the network to more easily capture large-scale information relevant to the remaining five parameters.

have less variance. The dominating mismatch seems to be attributable to the prior window effect.

- X-ray energy threshold for self-absorption by host galaxies E_0
  Both datasets yield the same regression behaviour, where predictions worsen for larger values. Scenarios of large E_0 become indistinguishable [15] as radiation that does not escape its host galaxy for a value of E_0 will continue to be confined for greater values of E_0.
- Minimum virial temperature T_{vir}
  While in $\mathrm{HR}_{\mathrm{x2}}$ T_{vir} is only robustly predicted for values $\gtrsim 4.25$, $\mathrm{LR}_{\mathrm{x2}}$ yields an accurate prediction all throughout with smaller deviations. Though the poor accuracy for $\lesssim 4.25$ was previously explained by a degeneracy with m_{WDM} [15], it was robustly predicted on $\mathrm{HR}_{\mathrm{x2}}$.

Overall, the information lost due to the lower simulation resolution of $\mathrm{LR}_{\mathrm{x2}}$ seems to be outweighed by the greater data size as well as not needing to regress m_{WDM}. The loss of small-scale information can make it easier for the network to capture large-scale information that is relevant for the other five parameters. In the subsequent analysis, $\mathrm{LR}_{\mathrm{x2}}$ will be regarded as uninformative of m_{WDM}.

6.1.2 Downsampling comparison

To understand the effect of downsampling the datasets, parameter regression on $\mathrm{LR}_{\mathrm{x2}}$ and $\mathrm{LR}_{\mathrm{x5}}$ is compared.

The subpanels in Fig. 10[¶] show that all predictable parameters are regressed more accurately and confidently in the downsampled variant $\mathrm{LR}_{\mathrm{x5}}$. While the higher resolution variant $\mathrm{LR}_{\mathrm{x2}}$ may contain information that is lost in downsampling, the information retained that is relevant to regression is expected to be more densely distributed in the smaller voxel space. This may make the loss landscape smoother, and thus training on it easier and more stable. As training is on the order of four times faster on the x5-downsampled variations and predictions are generally better, the downsamples will be used where possible in the following sections.

[¶] In this type of plot comparing model regression the scales incorrect and should be switched for E_0 and L_x

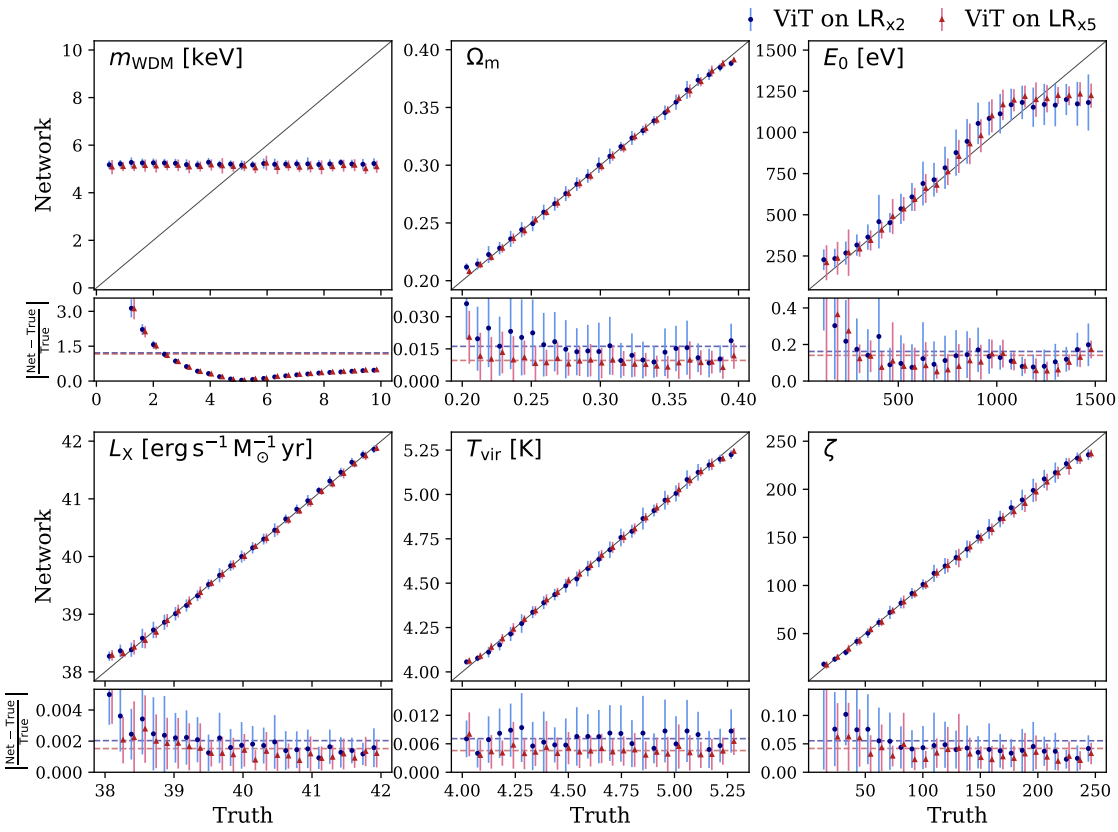

Figure 10: Comparing LR_{x2} (blue circles) to its downsampled variant LR_{x5} (red triangles) by scratch ViT simulation parameter regression.
The six panels show the network predictions as a function of the true simulation parameters. Parameter values are binned along the x-axis. The mean and standard deviation in each bin are displayed. The black diagonals indicate the ideal prediction curve. The subpanels show the mean absolute relative error in each bin. The coloured dashed lines indicate the mean in this error.
Regression is more accurate and of lower variability on the downsampled variant LR_{x5}. By downsampling, the smaller voxel space is expected to be more densely populated with information relevant to regression. This may make the loss landscape smoother, and thus training on it easier and more stable.

6.2 Architecture comparison

The adaptation 5.2.2 of the CNN architecture established in previous work on SKA simulation-based inference [14] is compared against the ViT architecture 5.2.1. The benchmark task is a regression of the simulation parameters. Training is performed as described in Sect. 5.3.

6.2.1 Regression performance

To most fairly compare the ViT architecture to Ref. [14] both networks are compared on the x2 variants. This is because our ViT architecture is not optimized for the full resolution and our adaptation of the CNN is only optimized for full resolution and x2-downsampled variants.

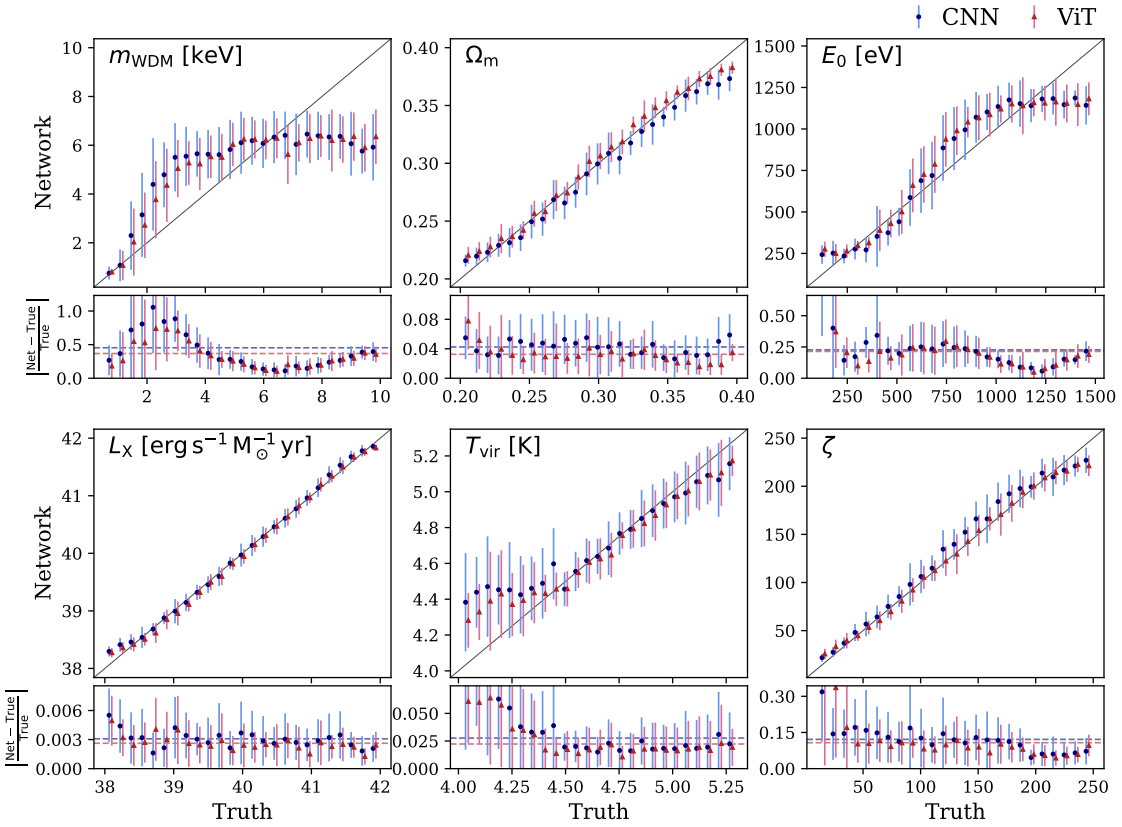

Figure 11: Comparing our adaptation of the original CNN architecture [14] (blue circles) and ViT (red triangles) architectures by simulation parameter regression on the downsampled variant $\mathrm{HR}_{\mathrm{x2}}$.
The six panels show the network predictions as a function of the true simulation parameters. Parameter values are binned along the x-axis. The mean and standard deviation in each bin are displayed. The black diagonals indicate the ideal prediction curve. The subpanels show the mean absolute relative error in each bin. The coloured dashed lines indicate the mean in this error.
The ViT consistently makes more accurate predictions with lower variability than the CNN. The networks show the same predictive behaviours across the different parameters.

The regression behaviour of the CNN and ViT network seen in Fig. 11 largely show the same success and failure modes discussed previously 6.1.1. The ViT consistently makes more

accurate predictions with lower variability than the CNN. The mean MARE in five of the six parameters is lower, and variations turn out smaller.

6.2.2 Data scaling

To study the scaling behaviour of the two architectures with growing data size, networks are trained on regression on the x5-downsampled variant LR_{x5} with varying training set sizes. Due to the high computational cost of training a batch of networks, the study is performed on the x5-downsampled variant LR_{x5}. The high-resolution dataset HR_{x5} was not selected as it contains approximately 7x fewer LCs and therefore does not allow a sufficiently large range of data sizes to be analysed. As mentioned in the previous section 6.2.1, this comparison is not entirely fair, since the x5 CNN architecture adaptation was not optimized fully 5.2.2. As the optimization of the architecture is expected to affect the networks independent of training set size, the scaling behaviour should still be observable but can not be compared to the ViT in absolute terms.

In Fig. 12 the data scaling behaviour of the CNN and ViT architecture is shown. The mean of the normalized absolute error

$$NAE = \frac{\sum_{i=1}^{n} \left|\text{truth}_i - \text{pred}_i\right|}{\sum_{i=1}^{n} |\text{truth}_i|} \tag{17}$$

in parameter, prediction is shown as a function of the number of training set LCs. The data points indicate the mean and standard deviation of a minimal ensemble of three independently trained networks. The dotted lines indicate a polynomial fit intended to approximate the scaling behaviour. The difference in the NAEs between the two architectures can be considered constant across the training set sizes and points to the same scaling behaviour. Increasing the data size improves performance most when only a few training LCs are available. This improvement plateaus but remains observable until the maximum number of roughly 33k training LCs is reached.

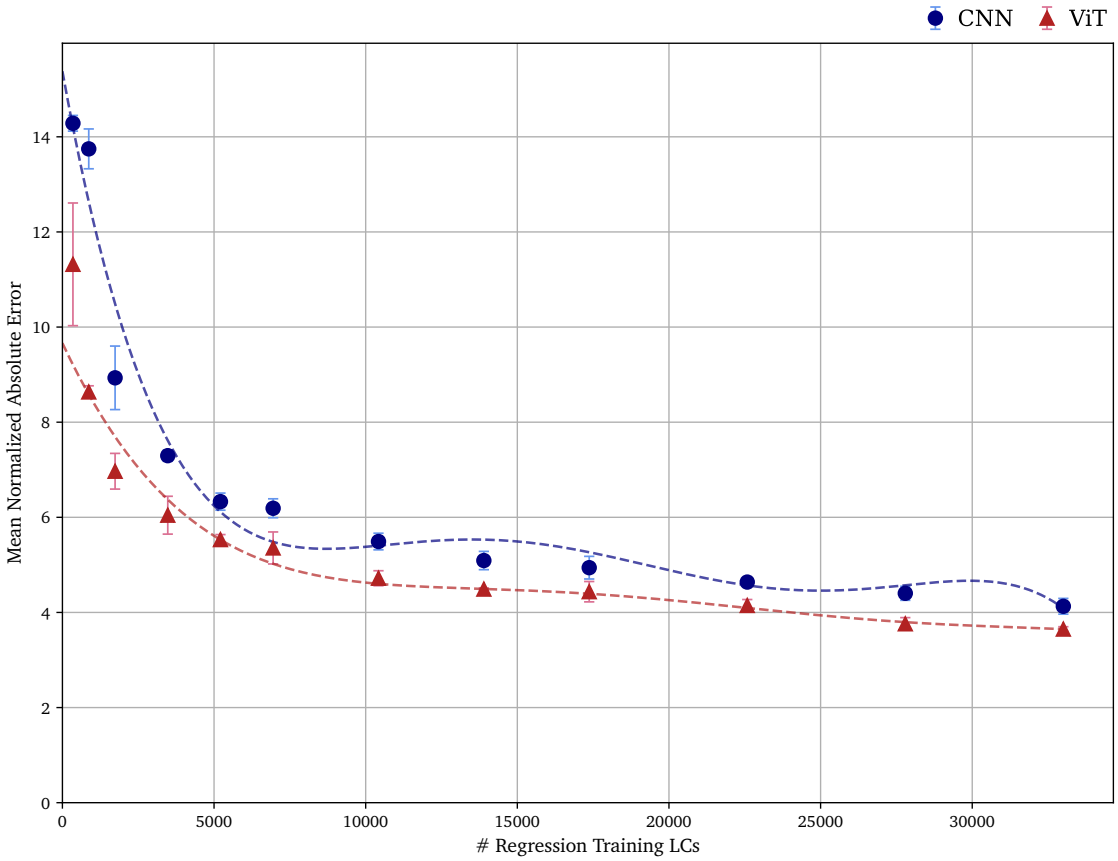

Figure 12: Comparing data scaling behaviour of CNN (blue circles) and ViT (red triangles) networks on simulation parameter regression on the downsampled variant of the large dataset LR_{x5}.
Each mean normalized absolute error (NAE) in parameter prediction is shown as a function of the number of training set LCs. The data points indicate the mean and standard deviation of three networks. The dotted lines indicate a polynomial fit intended to approximate the scaling behaviour.
The difference in the NAEs between the two architectures can be considered constant across the training set sizes and points to the same scaling behaviour. Increasing the amount of data improves performance the most when only a few training LCs are available. This improvement plateaus but remains observable until the maximum number of training LCs available is reached. The absolute NAEs are not considered representative, as the CNN adaptation is not fully optimised for x5 downsample variants.

6.3 SKATR summary

The behaviour of the SKATR 5.1 summary is investigated in various contexts. Comparisons are made on the regression of simulation parameters in reference to the ViT architecture previously shown to outperform the CNN architectures 6.2.1. Downstream SKATR regression is performed as introduced in Sect. 5.3, where a fully connected MLP is trained on the SKATR summary vector. The ViT 5.2.1 trained from scratch gives the baseline performance and shares the same architecture as the SKATR encoder networks. As SKATR foundation models are only available for employment on the x5 dataset variants, these will be used throughout. The test split is independent of both downstream and pretraining splits.

6.3.1 Regression performance

SKATR is first employed on $\mathrm{LR}_{\mathrm{x5}}$, the largest dataset and the same it was pretrained on.

In Fig. 13, the SKATR summary is compared against the scratch ViT. It is observed that the Downstream SKATR summary regression matches the scratch ViT network in all parameters up to marginal differences except for the specific X-ray luminosity L_{X}. In the Specific X-ray luminosity L_{X} the SKATR summary is slightly less predictive of truth parameters. The networks' regression prediction behaviours are similar on all parameters. The near match in performance suggests that the SKATR summary is as informative of the regression parameters as the high-dimensional freely trained scratch ViT embedding. Since the backbone weights are frozen in downstream training, it can be concluded that SKATR has learned to summarize LCs into a nearly fully informative summary.

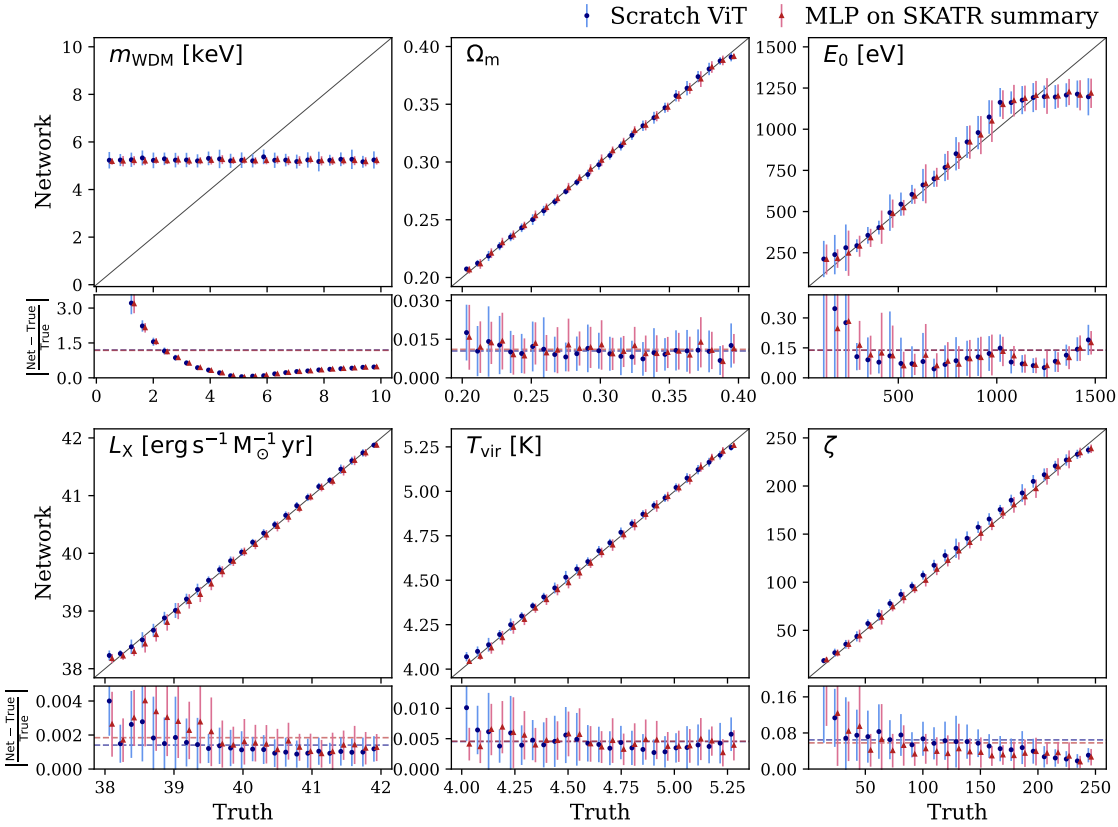

Figure 13: Comparing downstream on the SKATR summary (red triangles) regression performance on $\mathrm{LR}_{\mathrm{x5}}$ parameter regression against the scratch ViT (blue circles). SKATR encoders and the ViT share the same architectures. Regression training is performed on the same dataset as SKATR pretaining 5.1.
The six panels show the network predictions as a function of the true simulation parameters. Parameter values are binned along the x-axis. The mean and standard deviation in each bin are displayed. The black diagonals indicate the ideal prediction curve. The subpanels show the mean absolute relative error in each bin. The coloured dashed lines indicate the mean in this error.
Downstream SKATR summary regression matches the scratch ViT network up to marginal differences in all parameters except for the specific X-ray luminosity L_{X}. Network behaviours are similar on all parameters. The SKATR summary can be considered nearly as informative of the regression parameters as the scratch ViT embedding.

6.3.2 Data efficiency

In this section, the data efficiency of the SKATR summary is investigated. The number of training LCs available for downstream/scratch training is varied to study the impact of restricted data availability. A minimal ensemble of three networks is trained at each data size. On these, the mean model performance and its standard deviation are calculated.

6.3.2.1 Pretraining dataset

Networks are trained on LR_{x5}, the same dataset as used during SKATR pretraining.

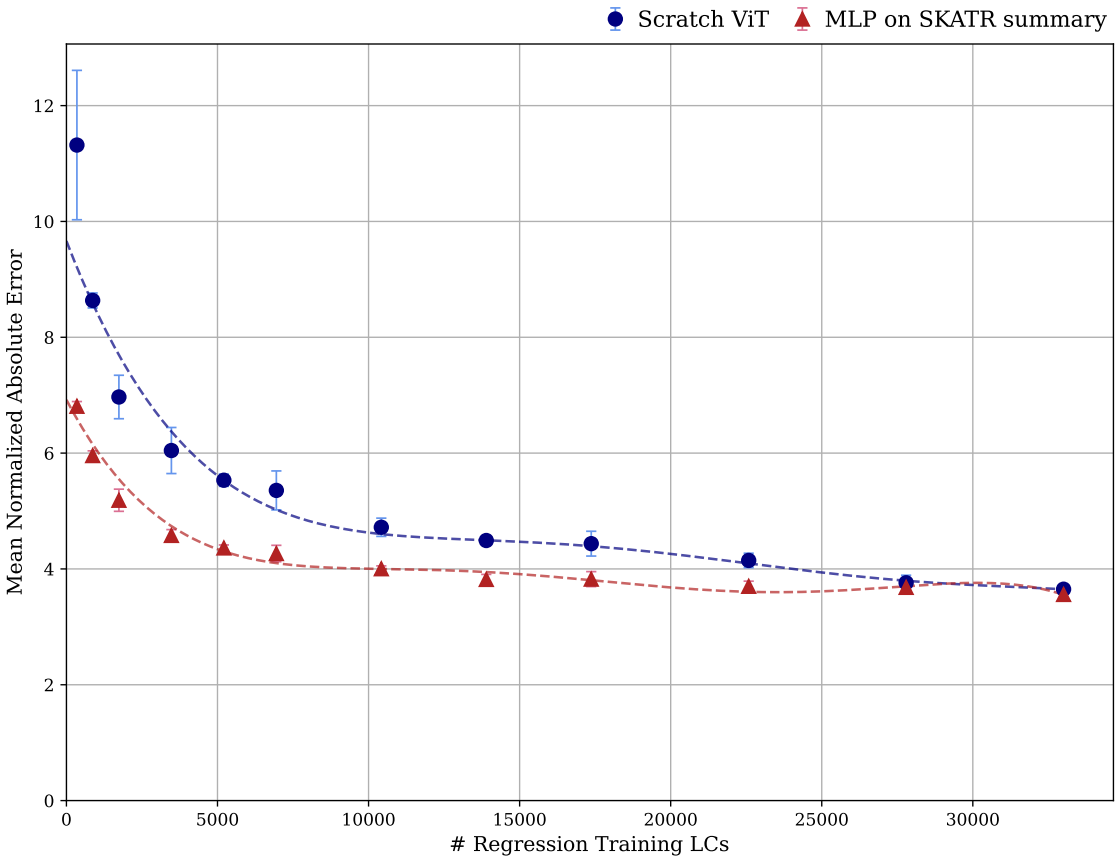

Figure 14: Comparing data efficiency of the SKATR summary downstream (red triangles) on the basis of regression performance on LR_{x5} parameter regression against the scratch ViT (blue circles). SKATR encoders and the ViT share the same architectures. Regression training is performed on the same dataset as SKATR pretaining 5.1.
The mean normalized absolute error (NAE) in parameter prediction is shown as a function of the number of training set LCs. The data points indicate the mean and standard deviation of three networks. The dotted lines indicate a polynomial fit intended to approximate the scaling behaviour.
The downstream SKATR summary performs significantly better when only a few hundred LCs are available. This gap shrinks with increasing data size. The SKATR summary requires fewer training LC examples.

As seen in Fig. 14 the summary network yields better performance for low data availability and is matched by the scratch model as the number of regression training LCs increases. The

difference is most pronounced with a number of training LCs in the order of a few hundred, but remains up to $\lesssim 25000$ training LCs. In this regime, the pretraining schedule has effectively given the SKATR summary many more views of the data. The SKATR summary thus requires fewer LC examples to attain the same predictive power as the scratch ViT.

As seen in the parameter-wise data efficiency of the SKATR summary 15, the previously observed 6.3.1 deficiency in the prediction of L_X is observed. It is apparent that this predictive deficiency is small but is shown by the network variance to be statistically significant. Thus, the SKATR summary is not as informative of L_X as the scratch ViT embedding.

6.3.2.2 Transfer learning

To study the robustness of the SKATR summary, the comparison made in the previous section is repeated on the high simulation resolution dataset $\mathrm{HR}_{\mathrm{x5}}$. Contrary to the SKATR pretraining dataset $\mathrm{LR}_{\mathrm{x2}}$, the HR dataset is informative of the WDM mass m_{WDM} 6.1.1 and is thus structurally distinct from the pretraining dataset. Furthermore, SKATR pretraining did not include the mock noise present in $\mathrm{HR}^{\mathrm{noise}}$. This enables the study of the effect of realistic detector influences on the SKATR summary. These two structural differences in the data set are used to investigate how the summary performs in transfer learning when confronted with a structure that has not been shown before.

The parameter wise SKATR summary data efficiency on $\mathrm{HR}_{\mathrm{x5}}$ shown in Fig. 16 reflects the previously observed structure in all parameters up to the WDM mass m_{WDM}. Downstream SKATR summary regression matches the scratch ViT network up to marginal differences in these parameters, including the previously observed deficiency in the prediction of L_X 6.3.2.1. Notably, the downstream SKATR summary learns m_{WDM}, which the pretraining dataset $\mathrm{LR}_{\mathrm{x5}}$ was not informative of. It can be concluded that the SKATR summary can transfer what it has learned in pretraining to the newly apparent parameter m_{WDM}.

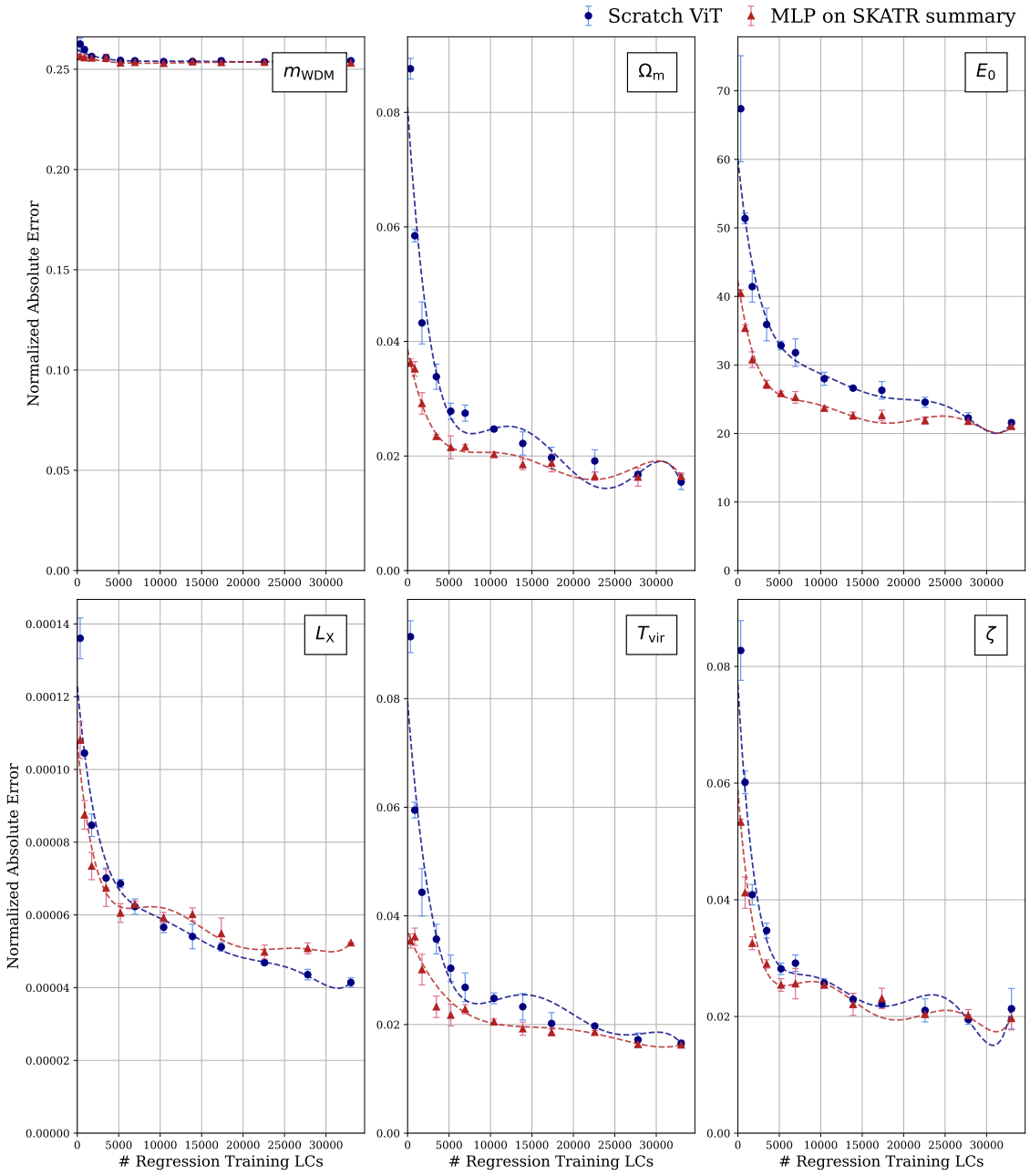

Figure 15: Comparing parameter-wise data efficiency of the SKATR summary downstream (red triangles) on the basis of regression performance on $\mathrm{LR}_{\mathrm{x5}}$ parameter regression against the scratch ViT (blue circles). SKATR encoders and the ViT share the same architectures. Regression training is performed on the same dataset as SKATR pretraining 5.1.
The six panels show the normalized absolute error in parameter prediction as a function of the number of training LCs. The data points indicate the mean and standard deviation of three networks. The dotted lines indicate a polynomial fit intended to approximate the scaling behaviour.
The previously observed 6.3.1 deficiency in the prediction of L_{X} is observed and shown to be statistically significant. In this parameter, the SKATR summary is not as informative as the scratch ViT embedding.

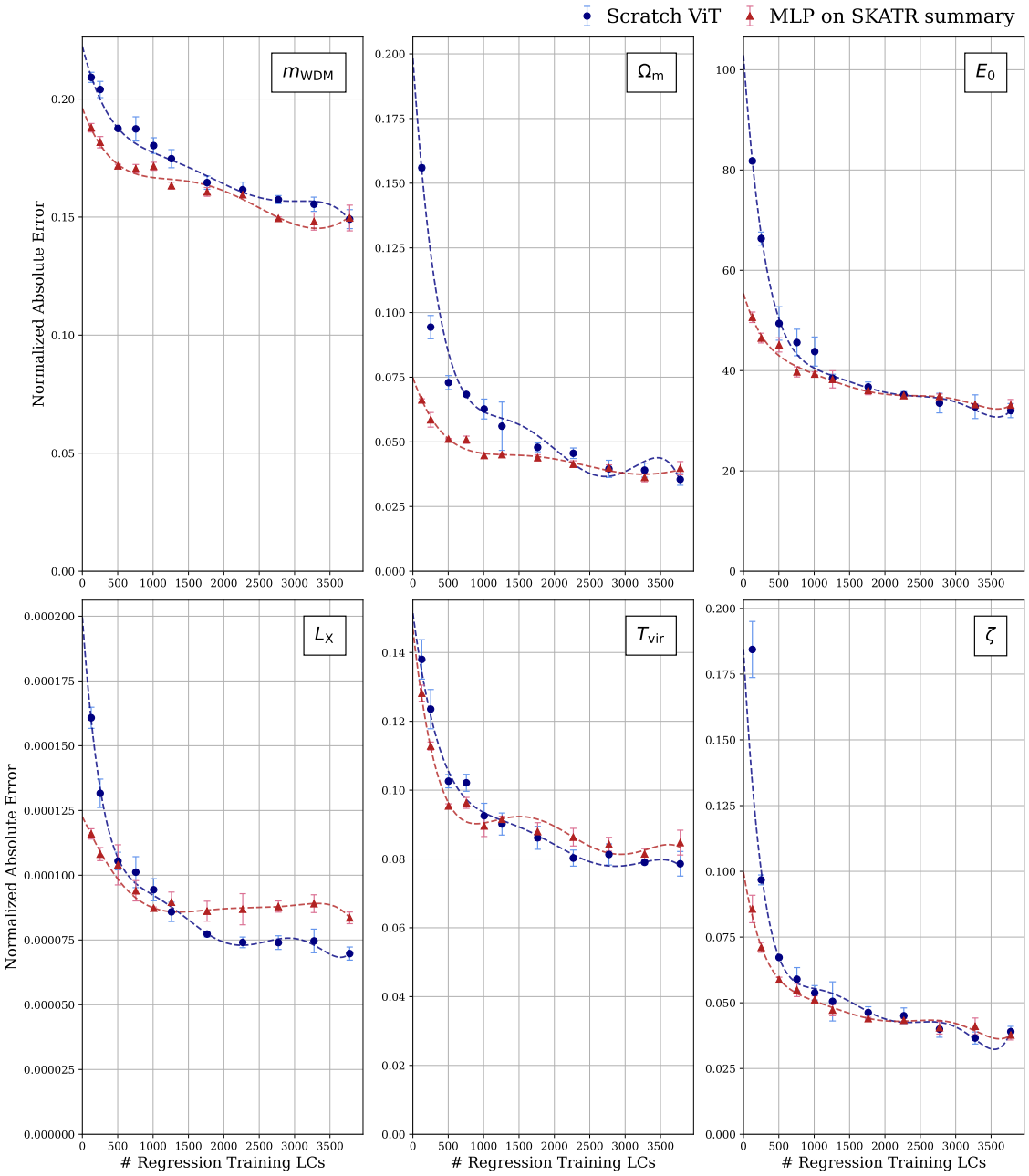

Figure 16: Comparing parameter-wise data efficiency of the SKATR summary downstream (red triangles) on the basis of regression performance on $\mathrm{HR}_{\mathrm{x5}}$ parameter regression against the scratch ViT (blue circles). SKATR encoders and the ViT share the same architectures. The regression training dataset $\mathrm{HR}_{\mathrm{x5}}$ is distinct from the dataset $\mathrm{LR}_{\mathrm{x5}}$ used in SKATR pretraining 5.1. In particular, $\mathrm{HR}_{\mathrm{x5}}$ is informative of the m_{WDM} while $\mathrm{LR}_{\mathrm{x5}}$ is not 6.1.1.
The six panels show the normalized absolute error in parameter prediction as a function of the number of training LCs. The data points indicate the mean and standard deviation of three networks. The dotted lines indicate a polynomial fit intended to approximate the scaling behaviour.
Downstream SKATR summary regression matches the scratch ViT network up to marginal differences in all parameters, except for the previously observed deficiency in the prediction of L_{X} 6.3.2.1. Importantly, this includes m_{WDM}, which the SKATR pretraining dataset was not informative of. It can be concluded that the SKATR summary can transfer what it has learned in pretraining to the newly apparent parameter m_{WDM}.

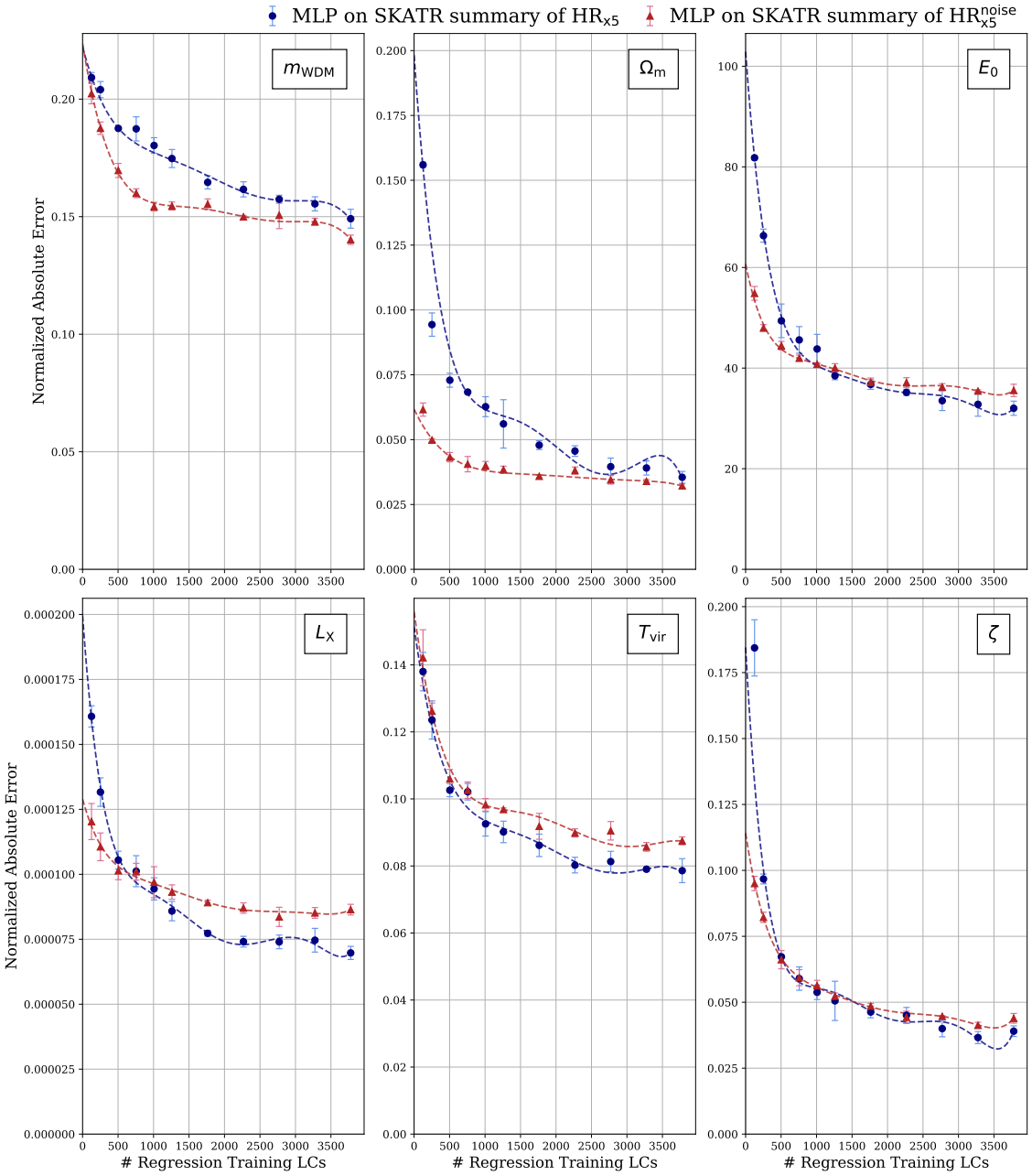

Figure 17: Comparing parameter-wise data efficiency of the SKATR summary downstream on HR_{x5} (red triangles) and HR_{x5}^{noise} (blue circles) on the basis of regression performance. The regression training dataset HR_{x5}^{noise} is distinct from the dataset LR_{x5} used in SKATR pretaining 5.1. In particular, HR_{x5}^{noise} is both informative of the m_{WDM} 6.1.1 and contains mock noise 4.1 where LR_{x5} does not satisfy either aspect. The six panels show the normalized absolute error in parameter prediction as a function of the number of training LCs. The data points indicate the mean and standard deviation of three networks. The dotted lines indicate a polynomial fit intended to approximate the scaling behaviour.
The Downstream SKATR summary predictions show different behaviours on the noised dataset HR_{x5}^{noise}. On T_{vir} the predictions are worse or equal for all data sizes. On 'E_0', L_X and ζ the predictions are better for few LCs and worse for many. In the case of m_{WDM} and Ω_m the predictions are improved all throughout. On average, the predictions are of similar quality as the unnoised dataset HR_{x5}. It can be concluded that the SKATR summary exhibits a structure that is robust to learning noise.

In Fig. 17 the SKATR summary parameter regression is compared between the unnoised dataset $\mathrm{HR}_{\mathrm{x5}}$ and its noised variant $\mathrm{HR}_{\mathrm{x5}}^{\mathrm{noise}}$. The summary exhibits a different behaviour on the noised dataset. For the minimum virial temperature T_{vir} the predictions are worse or equal for all data sizes. For the X-ray energy threshold of self-absorption by host galaxies E_0', the predictions for the specific X-ray luminosity L_{X} and the ionisation efficiency ζ are better for a few LCs and worse for many. In the case of WDM mass m_{WDM} and the dark matter density parameter Ω_{m} predictions are improved all throughout. A tentative explanation for these discrepancies is that adding noise reduces the information about small-scale structures. As certain parameters are more sensitive to small- and others to large-scales, the accuracy on parameters more sensitive to small-scales would decrease. Yet despite the WDM mass m_{WDM} being more sensitive to small-scale information [14] it is predicted more accurately in the noised dataset $\mathrm{HR}_{\mathrm{x5}}^{\mathrm{noise}}$. A more refined explanation poses that the loss of information is redshift-dependent as the strength in added noise increases with redshift. Thus, more small-scale information is lost with growing redshift. This results in a disproportionate increase in the predictive power of large-scales for high redshifts (and a disproportionate decrease in the predictive power of small-scales for high redshifts). On average, the predictions are of similar quality as the unnoised dataset $\mathrm{HR}_{\mathrm{x5}}$. It can be concluded that the SKATR summary exhibits a structure that is robust to learning the noise modelled after realistic detector influences present in $\mathrm{HR}_{\mathrm{x5}}^{\mathrm{noise}}$.

7 Conclusion

7.1 Summary

The foundation model SKATR is successfully developed and deployed on 21cmFast noised LCs. The database of simulated LCs is expanded and investigated using simulation parameter regression. The ViT architecture used in SKATR is shown to outperform the CNN used in previous work. The SKATR summary is shown to yield an informative representation of the LCs that is robust to learning LC effects not seen during pretraining.

The data studies showed that dataLRx2 is uninformative of m_{WDM}. The information lost due to the lower simulation resolution of dataLR seems to be outweighed by the greater data size as well as not needing to regress m_{WDM}. The loss of small-scale information may help the network to more easily capture large-scale information relevant to the remaining five parameters. It is shown that parameter regression on downsampled LCs is more accurate and of lower variability. In downsampling, the smaller voxel space is expected to be more densely populated with information relevant to regression. This may make the loss landscape smoother, and thus training on it easier and more stable.
Comparing CNN to ViT architectures shows the same predictive behaviour. The ViT consistently makes more accurate predictions with lower variability than the CNN. This justifies the adoption of ViT architectures as the encoders used in SKATR. Increasing the number of training LCs improves performance most when few training LCs are available. This improvement plateaus but remains observable until the maximum number of roughly 33k training LCs is reached. Thus, a further expansion of the database is expected to increase accuracy in simulation parameter regression.
The SKATR summary is examined on the available data in various regression regimes. For unnoised LCs, the summary retains a LC representation fully informative of the simulation parameters, except for a slight decrease in predictive power on the specific X-ray luminosity L_{X}. The summary data efficiency for LCs taken from the same low-resolution dataset as used for pretraining is found to be statistically significant for $\lesssim 25000$ downstream training LCs. The summary is shown to learn new structures not present in pretraining LCs. By taking advantage of the low-resolution pretraining LCs not being informative of m_{WDM} the summary is shown to fully learn the m_{WDM} parameter despite it being absent in pretraining. The summary exhibits a qualitatively different behaviour on noised LCs. Compared to the unnoised LCs, some parameters are predicted more and others less robustly, depending on the number of regression training LCs. Explaining this discrepancy by certain parameters being more sensitive to small- and others to large-scale structures is inconsistent. In the case of m_{WDM} small-scale information is more relevant, yet it is predicted more accurately in the noised LCs. Overall, noised LC predictions are of comparable accuracy as unnoised LCs. Consequently, SKATR is shown to retain a 144-dimensional informative summary of 21cmFAST light cones (LCs), accounting for realistic noise modeled after SKA detector influences.

7.2 Outlook

There are many possible avenues for foundation models on SKA data. Firstly, there are several ways of improving the current model performance.

Though in early tests, the SKATR free model showed no improvement over the SKATR summary in parameter regression, no firm restriction was attained on the validity of the free model. Reasons as to why the free model currently does not show at least equal performance

to the summary should be investigated in more detail. While scratch training continues to yield better regression past 33k light cones, pre-training seems to plateau after 15k LCs. This might be due to the summary's low dimensionality. For long pretraining epochs, a higher-dimensional summary given by the ViT encoder hidden dimension may yield an improvement as it gives the model a larger encoding space to retrain information. Currently, the summary is a simple mean of the attention heads. More sophisticated summary schedules could be explored to retain more information contained in the heads. As pretraining is expected to profit more from data size than the downstream task, this indicates that pretraining could be optimized further. Pre-training hyperparameters, such as different masking schedules, could be investigated more quantitatively. Furthermore, additional augmentations could be explored. Noise has already been shown to improve performance on some simulation parameters. This effect could be used to attain an improved representation during pretraining. Preprocessing that employs a clamping of large and negative LC values using the function

$$T(x) = \mathrm{sgn}(x) \cdot \log\left(|x| + 1\right) \ , \tag{18}$$

was observed to yield better regression performance, but was not adopted for the final results. The interpretability of the informative SKATR summary attained is an area of particular potential. Summary studies could investigate the structure retained in the summary vector. For example, a visualization of the summary could identify clusters or correlation. Certain parts of the summary vector could be identified to correlate with regressing a certain simulation parameter.

References

[1] N. Aghanim et al. “Planck 2018 Results - VI. Cosmological Parameters”. In: *Astronomy & Astrophysics* 641 (Sept. 2020), A6. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201833910. (Visited on 08/05/2024).

[2] Mahmoud Assran et al. “Self-Supervised Learning From Images With a Joint-Embedding Predictive Architecture”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 15619–15629.

[3] Adrien Bardes et al. “Revisiting feature prediction for learning visual representations from video”. In: *arXiv preprint arXiv:2404.08471* (2024).

[4] Robert Braun et al. *Anticipated Performance of the Square Kilometre Array – Phase 1 (SKA1)*. Dec. 2019. DOI: 10.48550/arXiv.1912.12699. arXiv: 1912.12699 [astro-ph]. (Visited on 08/05/2024).

[5] Xinlei Chen and Kaiming He. *Exploring Simple Siamese Representation Learning*. Nov. 2020. DOI: 10.48550/arXiv.2011.10566. arXiv: 2011.10566 [cs]. (Visited on 08/04/2024).

[6] David R. DeBoer et al. “Hydrogen Epoch of Reionization Array (HERA)”. In: *Publications of the Astronomical Society of the Pacific* 129.974 (Mar. 2017), p. 045001. ISSN: 1538-3873. DOI: 10.1088/1538-3873/129/974/045001. URL: http://dx.doi.org/10.1088/1538-3873/129/974/045001.

[7] Alexey Dosovitskiy et al. *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. June 2021. DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929 [cs]. (Visited on 08/10/2024).

[8] Bradley Greig and Andrei Mesinger. “Simultaneously Constraining the Astrophysics of Reionisation and the Epoch of Heating with 21CMMC”. In: *Proceedings of the International Astronomical Union* 12.S333 (Oct. 2017), pp. 18–21. ISSN: 1743-9213, 1743-9221. DOI: 10.1017/S1743921317011103. arXiv: 1705.03471 [astro-ph]. (Visited on 08/05/2024).

[9] M. P. van Haarlem et al. “LOFAR: The LOw-Frequency ARray”. In: *Astronomy & Astrophysics* 556 (July 2013), A2. ISSN: 1432-0746. DOI: 10.1051/0004-6361/201220873. URL: http://dx.doi.org/10.1051/0004-6361/201220873.

[10] *High-Mass X-ray Binaries and the Cosmic 21-Cm Signal: Impact of Host Galaxy Absorption | Monthly Notices of the Royal Astronomical Society | Oxford Academic*. https://academic.oup.com/mnras/ar (Visited on 08/05/2024).

[11] Andrei Mesinger, Steven Furlanetto, and Renyue Cen. “21cmFAST: A Fast, Semi-Numerical Simulation of the High-Redshift 21-Cm Signal”. In: *Monthly Notices of the Royal Astronomical Society* 411.2 (Feb. 2011), pp. 955–972. ISSN: 00358711. DOI: 10.1111/j.1365-2966.2010.17731.x. arXiv: 1003.3878 [astro-ph]. (Visited on 08/09/2024).

[12] Andrei Mesinger et al. *Constraining the Astrophysics of the Cosmic Dawn and the Epoch of Reionization with the SKA*. Jan. 2015. DOI: 10.48550/arXiv.1501.04106. arXiv: 1501.04106 [astro-ph]. (Visited on 08/09/2024).

[13] Steven G. Murray et al. “21cmFAST v3: A Python-integrated C Code for Generating 3D Realizations of the Cosmic 21cm Signal.” In: *Journal of Open Source Software* (Oct. 2020). ISSN: 2475-9066. DOI: 10.21105/joss.02582. (Visited on 08/04/2024).

[14] S. Neutsch, C. Heneka, and M. Brüggen. “Inferring Astrophysics and Dark Matter Properties from 21cm Tomography Using Deep Learning”. In: *Monthly Notices of the Royal Astronomical Society* 511.3 (Feb. 2022), pp. 3446–3462. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stac218. arXiv: 2201.07587 [astro-ph]. (Visited on 08/04/2024).

[15] Benedikt Schosser, Caroline Heneka, and Tilman Plehn. *Optimal, Fast, and Robust Inference of Reionization-Era Cosmology with the 21cmPIE-INN*. Jan. 2024. DOI: 10.48550/arXiv.2401.04174. arXiv: 2401.04174 [astro-ph, physics:hep-ph]. (Visited on 08/02/2024).

[16] B. P. Sowmya and M. C. Supriya. "Convolutional Neural Network (CNN) Fundamental Operational Survey". In: *Intelligent Computing Paradigm and Cutting-edge Technologies*. Ed. by Margarita N. Favorskaya et al. Cham: Springer International Publishing, 2021, pp. 245–258. ISBN: 978-3-030-65407-8. DOI: 10.1007/978-3-030-65407-8_21.

[17] Ashish Vaswani et al. *Attention Is All You Need*. Aug. 2023. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762 [cs]. (Visited on 08/05/2024).

[18] Bruno Villasenor et al. "New Constraints on Warm Dark Matter from the Lyman-α Forest Power Spectrum". In: *Physical Review D* 108.2 (July 2023), p. 023502. ISSN: 2470-0010, 2470-0029. DOI: 10.1103/PhysRevD.108.023502. arXiv: 2209.14220 [astro-ph]. (Visited on 08/05/2024).

[19] A. Weltman et al. "Fundamental Physics with the Square Kilometre Array". In: *Publications of the Astronomical Society of Australia* 37 (2020), e002. ISSN: 1323-3580, 1448-6083. DOI: 10.1017/pasa.2019.42. arXiv: 1810.02680 [astro-ph, physics:hep-ph]. (Visited on 08/04/2024).

[20] Ya. B. Zel'dovich. "Gravitational Instability: An Approximate Theory for Large Density Perturbations." In: *Astronomy and Astrophysics* 5 (Mar. 1970), pp. 84–89. ISSN: 0004-6361. (Visited on 08/05/2024).

Acknowledgements

I would like to thank my supervising professor Tilman Plehn for suggesting this area of study. During the course of the project, he was always clear on the direction of the work. With his large overview, he provided feedback that was uniquely helpful. Caroline Heneka as the SKA expert effectively became my co-supervising professor. In discussions, she gave suggestions and background knowledge that helped tremendously.

In the difficult periods of writing the thesis, my dear friends provided me with strength and endurance. Special thanks are regarded to Seraphin, Filip and Leon. They were there for me during the course of my studies in physics. Especially in the last week, they did everything to support my work. I thank Lili for cheering me on and filling me with confidence in my work, as well as lifting me up when I was low. Last and most certainly not least I want to thank my supervisor Ayo. He was the most helpful, patient, and mentor I could have asked for. I enjoyed the endless discussions we had over the course of our project and learned much from him. He is responsible for much of the code implementation and had built a large code base before I started working on the project. Without his mentorship, this thesis would not have been possible.

Working on this project has been a pleasure, and I will not forget what I have learned throughout. I am excited to see what will be built on the back of our work on foundation models and look forward to seeing the field of informative Machine Learning grow.

Declaration

I hereby declare that I wrote this work independently and have not used any aids other than the ones indicated.

Heidelberg, 10. August 2024,

Aaron Johannes Nordmann