Applying Contrastive Learning to Stellar Spectra

April 7, 2025

Abstract

With its wide range of spectroscopic capabilities, the Near Infrared Spectrograph (NIRSpec) on the *James Webb* Space Telescope (JWST) is expected to usher in a new era of crowded-field extragalactic stellar spectroscopy. However, small sample sizes and limited overlap with ground-based surveys, such as APOGEE, pose challenges for NIRSpec data analysis. In this work, we present StarCLIP: a contrastive self-supervised learning framework that embeds observed APOGEE and *ab initio* NIRSpec spectra into a unified, physically meaningful latent space. Our approach consists of training convolutional neural networks (CNNs) to recover twenty fundamental stellar properties from single-modal spectroscopic data. We then adopt these pre-trained CNNs as encoders, aligning them via contrastive loss. To simulate realistic NIRSpec observations, we construct semi-empirical, stochastic NIRSpec catalogs and embed them into the shared latent space using MockStarCLIP, a modified CLIP-based framework. Both models enable seamless transfer to downstream tasks, including cosine similarity search and stellar property recovery. Notably, a linear regressor applied to StarCLIP embeddings recovers all twenty stellar properties of interest with r^2 scores typically exceeding 0.88, including $T_{\rm eff}$ (with uncertainty $\lesssim 200$ K), $\log g$ ($\lesssim$0.07 dex) and [Fe/H] ($\lesssim$0.03 dex) for RBG-like stars. Applying the regressor on MockStarCLIP embeddings yields modestly reduced precision– approximately 450 K for $T_{\rm eff}$, 0.11 dex for $\log g$ and 0.06 dex for [Fe/H]. Ultimately, our data-driven approach demonstrates that foundation models for NIRSpec and other spectral surveys with similar constraints is well within reach.

Key words: methods: data analysis – techniques: spectroscopic – stars: fundamental parameters

1 Introduction

Absorption features on a star's spectrum encode its physical and chemical properties, which in turn provide a 'fossil record' of the formation history of its host galaxy [1]. The Near Infrared Spectroscopic Instrument (NIRSpec) [2, 3] on board the *James Webb* Space Telescope (JWST) [4] exhibits unique spectroscopic capabilities, enabling efficient, high-quality resolved star spectroscopy of neighbouring galaxies that have so far been too distant, faint or crowded for previous instruments to detect. These capabilities, though currently limited to small sample sizes ($n \sim 100$), have motivated the launch of several recent astronomical programs, including (i) completed surveys of ~ 100 red giant branch (RGB) stars in M31 (JWST-GO-2609; PI: Nidever[1]) and Kuiper Belt regions of four nearby debris discs (JWST-GO-01563; PI: Chen[2]), (ii) an ongoing survey of ~ 200 RGB stars across three isolated dwarf galaxies (JWST-GO-3788; PI: Wiesz[3]) and (iii) an upcoming survey to observe ~ 100 more M31 disc stars (JWST-GO-4735; PI: Sandford[4]).

However, due to its novelty and uniqueness, the domain of JWST resolved-star spectroscopy remains largely unexplored, and many open questions remain regarding optimal strategies to collect, reduce and analyse its valuable data. In recent years, an increasing number of astronomical surveys have employed data-driven methodologies from machine learning (ML) to extract insights from spectroscopic data. These algorithms are generally divided into two groups:

- **Supervised** algorithms use input-output pairs, provided by a human expert, to learn the mapping from a set of features to target variables. This mapping enables classification or regression tasks on unseen data. While highly effective, supervised methods rely on large quantities of annotated data, limiting their use in low-data regimes, where labels may be scarce, imbalanced or expensive to obtain. Moreover, these models are typically *task-specific*, requiring a dedicated model trained from scratch for each new problem (e.g., [5, 6]). This can lead to high computational costs, limited generalisability, and an increased risk of overfitting.
- **Unsupervised** algorithms bypass the need for labelled data, by taking in a set of raw features as inputs and producing non-linear, non-invertible transformations of the features as outputs, including: (i) clusters of similar objects, (ii) lists of detected anomalies, or (iii) low-dimensional representations of the data [7]. By uncovering previously unknown patterns in data, they are arguably more useful for 'AI for science' research, where human-annotated labels are not always available in quality or quantity. However, unsupervised models typically underperform compared to their supervised counterparts, limiting their practical adoption (e.g., [8, 9]).

Large-scale surveys such as Apache Point Observatory Galactic Evolution Experiment (APOGEE) [10, 11], Galactic Archaeology with HERMES (GALAH) [12] and Large Sky Area Multi-object Fiber Spectroscopic Telescope (LAMOST) [13] provide large ($n \sim 10^5 - 10^7$), homogeneous databases of labelled spectra that are virtually ideal for supervised ML applications. In contrast, JWST/NIRSpec (hereafter, NIRSpec) programs provide considerably smaller ($n \sim 100$) databases, which limits their suitability for supervised analysis. Moreover, large differences in sensitivity and wavelength coverage between NIRSpec and ground-based surveys make it difficult to acquire overlapping stars with previous datasets [4]. Indeed, within a narrow 3.6×3.4 arcmin^2 field of view (FOV), NIRSpec achieves an order of magnitude *greater* emission line sensitivity in the wavelength range $\lambda \sim 1 - 2.5$ μm compared to existing ground-based surveys, while reaching an even greater sensitivity at wavelengths $\lambda \gtrsim 2.5$ μm currently inaccessible to other observatories [14, 15, 16]. Although this enables NIRSpec to detect

[1] https://arxiv.org/pdf/2306.04688
[2] https://ui.adsabs.harvard.edu/abs/2021jwst.prop.1563C/abstract
[3] https://ui.adsabs.harvard.edu/abs/2023jwst.prop.3788W/abstract
[4] https://ui.adsabs.harvard.edu/abs/2024jwst.prop.4735S/abstract

faint astronomical objects, it poses challenges in leveraging observational resources from large surveys to facilitate analysis of NIRSpec spectra. A preliminary solution is to generate *ab initio* spectra to encourage overlap; what results is a database of *cross-modal*[5] observations of real and synthetic spectra of the same stars.

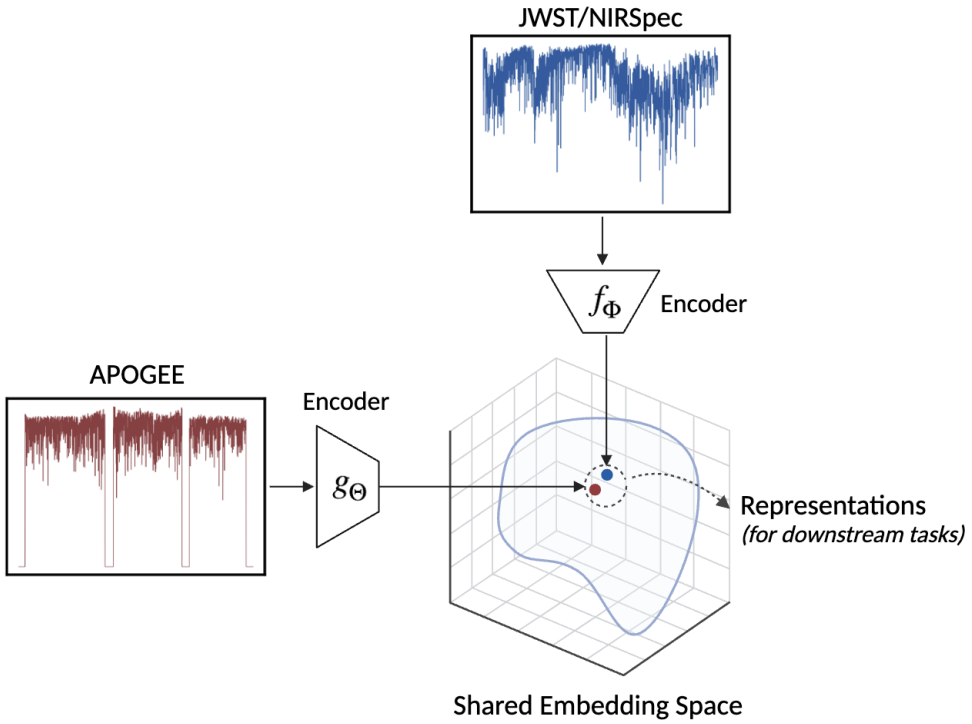


Figure 1: Illustration of StarCLIP model architecture; the MockStarCLIP architecture is analogous. Two encoders, f_Φ and g_Θ, separately embed cross-modal JWST/NIRSpec and APOGEE spectra into a shared embedding space, where they are aligned under contrastive loss. The learned embeddings encode high-level physical information, enabling seamless transfer to downstream tasks. The two modalities are distinguished by colour, with JWST/NIRSpec in blue and APOGEE in red.

One promising ML approach to analyse cross-modal realisations is **self-supervised learning** (SSL). In SSL, models are learned by solving auxiliary pretext tasks, in which supervision signals are acquired from the data itself without explicit labels. Once trained, these models can extract high-quality embeddings that serve as a 'foundation' for improving performance [17, 18], generalisation [19, 20] and robustness [21, 22] on downstream tasks. For this reason, they are often referred to as **foundation models** [23].

In this work, we introduce StarCLIP, and its variant MockStarCLIP, cross-modal foundation models for stellar spectroscopy. Our approach consists of two main steps. First, we train flexible 1-dimensional convolutional neural networks (1D- CNNs) on single-modal spectroscopic data to estimate a set of stellar properties. Second, we adopt the pre-trained CNNs as encoders that embed cross-modal data into a shared latent space. The encoders are jointly trained under a *contrastive* objective, whereby augmented[6] views of the same underlying star are brought closer together while those

[5]'Modality' refers to the type of data input. In this context, it refers to the astronomical program used to collect the spectra, such as APOGEE or JWST/NIRSpec.

[6]In this context, an 'augmented view' refers to a synthetically modified version of an original data point, generated to

of different stars are pushed apart. Ultimately, we show these embeddings are seamlessly transferrable across a wide range of downstream tasks, including semantic similarity searches and stellar property estimation, which ultimately facilitates analysis of NIRSpec data.

Spectroscopic data consist of 19,001 pairs of APOGEE Sloan Digital Sky Survey IV[7] Data Release 17 (SDSS-IV DR17) [25] and theoretical NIRSpec spectra, the latter generated *ab initio* from stellar atmosphere and spectrum synthesis models maintained by R. Kurucz [26, 27, 28, 29, 30, 31]. The stellar labels associated with each pair consist of atmospheric parameters $T_{\text{eff.}}$, $\log g$ and [Fe/H]—the primary determinants of RGB spectral flux—along with 17 other elemental abundances [X/H], including C, N, O, Na, Mg, Al, Si, S, K, Ca, Ti, V, Cr, Mn, Co, Ni and Ce. To approximate real NIRSpec observations, we apply two sets of semi-empirical, stochastic transformations to a subsample of 250 *ab initio* NIRSpec data, generating 'mock' and 'extended-mock' versions. A small sample ($n = 59$) of representative spectral traces from JWST-GO-3788 (PI: Weisz) is used in this data transformation pipeline. StarCLIP and MockStarCLIP models result from applying our methodology to cross-modal datasets comprised of augmented APOGEE × NIRSpec and extended-mock × mock NIRSpec pairs, respectively.

A schema of our pipeline is depicted in Figure 1. We summarise the contributions of this paper, as follows:

- To the best of our knowledge, we develop one of the first self-supervised foundation models for analysing NIRSpec data, along with a pipeline for seamless integration with observed data.
- We train robust CNNs in a supervised setting to predict stellar parameters with high precision and accuracy.
- We apply cross-modal contrastive training to align pre-trained encoders around shared physical properties, creating a discriminative latent space.
- We apply dimension-reduction to the latent space, visualising its intrinsic local geometry in a lower-dimensional setting.
- With minimal processing, our models enable accurate transfer to downstream tasks, including (i) in-modal and cross-modal cosine similarity searches and (ii) stellar property estimation from spectra. These tasks empirically show that latent embeddings capture key physical properties of underlying stars.

1.1 Related Works

This paper builds on two recent works. First, Fabbro et al. [5] introduce StarNet, a CNN model capable of determining atmospheric stellar parameters—$T_{\text{eff.}}$, $\log g$, and [Fe/H]—directly from observed APOGEE and *ab initio* APOGEE `ATLAS9`/ASSϵT [32, 33] spectra. Second, Parker et al. [22] develop AstroCLIP, a versatile foundation model that can embed cross-modal representations of galaxy images and spectra in a unified latent space. Their approach involves pre-training transformer-based encoders for images and spectra in a self-supervised setting, followed by aligning the encoders under contrastive loss. Parker et al. [22] show the resulting embeddings achieve state-of-the-art performance across a variety of downstream tasks—including photometric redshift estimation and physical property prediction—surpassing even supervised baselines. Both articles provide public access to their data and code on GitHub via the `StarNet` and `AstroCLIP` repositories.

In this work, we build our own foundation models in a two-step process involving carefully modified versions of StarNet and AstroCLIP.

increase the number of positive pairs for training (see, e.g., [24] for details).

[7] https://www.sdss4.org/

1.2 Outline

Code for our models is publicly available in the `StarCLIP` GitHub repository. Our paper is organised, as follows. In §2, we provide theoretical background on supervised and self-supervised methods relevant to this work, addressing their merits and disadvantages. In §3, we provide the spectrosopic datasets used for analysis. In §4, we discuss in detail the implementation and training process of the StarCLIP and MockStarCLIP models. We present our results on in-modal and cross-modal similarity searches, stellar property prediction, and non-linear manifold learning in §5. Finally, §6 contains some concluding remarks and further extensions to our work.

1.3 Notations

In this paper, we use $[n]$, where $n \in \mathbb{N}$, to denote the set $\{1, ..., n\}$. We use $|\cdot|$ to denote the dimension of vectors and $\|\cdot\|$ to represent their Euclidean ℓ_2-norm. We write $a \gg b$ whenever there exists a sufficiently small constant c such that $b/a < c$ holds. We denote by $a \otimes b$ the convolution operation between vectors a and b.

2 Machine Learning Methodology

In this section, we formalise the supervised and self-supervised methodologies underlying our models.

2.1 Supervised Learning

Let $X \in \mathcal{X} \subseteq \mathbb{R}^n$ and $Y \in \mathcal{Y} \subseteq \mathbb{R}^m$ be random variables, where $Y = f(X)$, for some unknown f. In supervised learning, the algorithm is provided with a set of training examples $\{(x_i, y_i)\}_{i=1}^T$, drawn from the joint distribution of X and Y. The goal is to learn a mapping $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that minimises the expected loss, as measured by a specific *loss function* $L : \mathcal{Y}^2 \to \mathbb{R}$. Formally, this can be expressed as:

$$\hat{f} = \text{argmin}_{f \in \mathcal{F}} \mathbb{E}[L(Y, f(X))], \tag{2.1}$$

where $\mathcal{F}$ is the space of hypotheses functions (see, e.g., [34]). In regression tasks, choosing the mean squared error (MSE) loss gives: $\hat{f} = \text{argmin}_{f \in \mathcal{F}} \mathbb{E}\|Y - f(X)\|^2$.

2.1.1 1D-CNNs: Feedforward Stage

1D-CNNs are a class of deep learning algorithms that solve equation (2.1) by adaptively learning spatial hierarchies of features in the input data. These networks process each training example $x \in X$ through a sequence of computational layers, which, in this analysis, consists of: (i) $N_{\text{CL}} = 2$ convolutional layers (CLs), (ii) an average-pooling layer and (iii) $N_{\text{FCL}} = 3$ fully connected layers (FCLs) (see Figure 5). Hereon, let $\zeta^{(l)}$ denote the output of the lth layer.

(i) **Convolutional layers**. Let $\{W_j^{(1)}\}_{j=1}^p$ denote a set of p filters, each of size $s^{(1)}$. Each filter slides along the input signal x, aggregating local information from contiguous segments of $s^{(1)}$ elements. Assuming a stride of 1, the discrete *convolution* between the filter weights and the input is computed as:

$$C_j^{(1)}(i) = (x \otimes W^{(1)})(i) = \sum_{u=1}^{s^{(1)}} x_{i+u-1} W_j^{(1)}(u), \tag{2.2}$$

where $C_j^{(1)}$ is a vector of length $n - s^{(1)} + 1$. As illustrated in Figure 2, applying all p filters produces a matrix $C^{(1)} \in \mathbb{R}^{(n-s^{(1)}+1)\times p}$, where the jth column corresponds to the convolution output $C_j^{(1)}(i)$ defined in Equation 2.2. Adding a bias term $b_j^{(1)} \in \mathbb{R}$ and applying an activation map $\phi : \mathbb{R} \to \mathbb{R}$ to the ijth element of $C^{(1)}$ gives the ijth convolved feature of the feature map $\zeta^{(1)}$:

$$\zeta_j^{(1)}(i) = \phi\left(C_j^{(1)}(i) + b_j^{(1)}\right). \tag{2.3}$$

To capture higher-order features, a second convolution is applied independently to each of the p columns $\zeta_j^{(1)}$ using a multi-filter represented by a third-order tensor of shape $s^{(2)} \times p \times q$, where $s^{(2)}$ is the filter size and $p \times q$ is the number of filters. The result is a matrix $C^{(2)} \in \mathbb{R}^{(n-s^{(2)}+1)\times q}$, where each column $C_j^{(2)}$ is computed by the sum of p convolutions:

$$C_j^{(2)}(i) = \sum_{l=1}^{p} \left(\zeta_l^{(1)} \otimes W_{l,j}^{(2)}\right)(i). \tag{2.4}$$

Adding bias $b_j^{(2)}$ and applying ϕ elementwise produces the feature map:

$$\zeta_j^{(2)}(i) = \phi\left(C_j^{(2)}(i) + b_j^{(2)}\right). \tag{2.5}$$

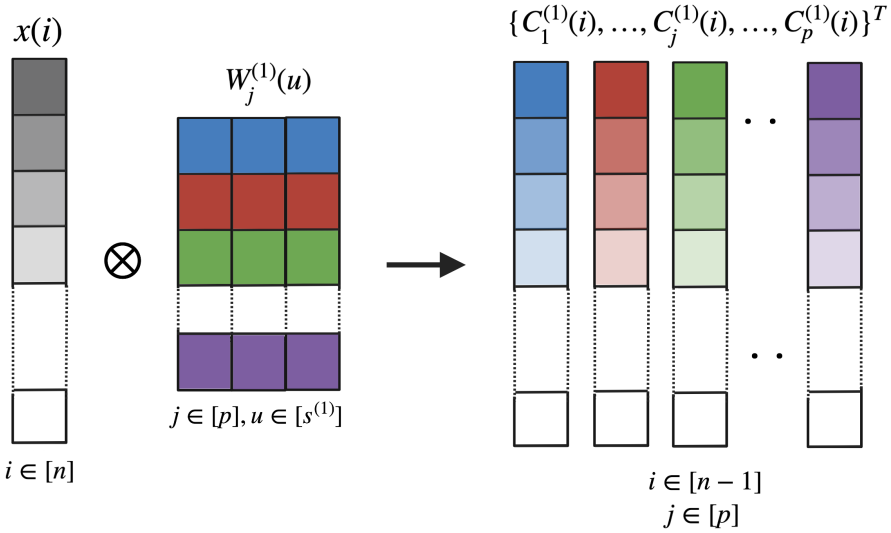


Figure 2: A 1D multi-filter convolution applied to an input signal x of length n, using a filter bank $\{W_j^{(1)}\}_{j=1}^{p}$ with p filters of size $s^{(1)}$. The result is a 2D matrix whose jth column $C_j^{(1)}$ is obtained by a single convolution (Equation 2.2). Illustration adapted from [35].

(ii) **Average-pooling layer**. The pooling layer downsamples the feature map $\zeta^{(2)}$ by a factor of w, thereby compressing its size while retaining only the strongest features learned in the CLs. Average-pooling is performed by sliding a window of size w along each column of $\zeta^{(2)}$, extracting the average value from each windowed sub-region, as follows:

$$\zeta_j^{(3)}(i) = \frac{1}{w}\sum_{k=1}^{w} \zeta_j^{(2)}(wi - k + 1). \tag{2.6}$$

This operation produces a downsampled output $\zeta^{(3)}$ of shape $w \times q$.

(iii) **Fully connected layers**. A flattening operation is applied to re-shape the 2D output $\zeta_j^{(3)}$ into a 1D vector f of length wq:

$$f(j + n(i-1)) = \zeta_j^{(3)}(i), \tag{2.7}$$

which can be fed into the fully-connected network. Assuming each FCL is parameterised by weights and biases $(W^{(\ell)}, b^{(\ell)})$, then:

$$\zeta^{(\ell)}(i) = \phi\left(\sum_{j=1}^{|\zeta^{(\ell-1)}|} W^{(\ell)}(i,j) f(j) + b^{(\ell)}(j)\right), \tag{2.8}$$

where $|\zeta^{(3)}| = wq$ and $|\zeta^{(5)}|$ is equal to the length of the true label or 'target' associated to x.

The final output of the network is $\zeta_{\text{out}} := \zeta^{(6)}$, representing the network's prediction of the target associated to x. While common activation functions include sigmoid, tanh and the rectified linear unit (ReLU), our work makes *ad hoc* use of the exponential linear unit (ELU) function:

$$\phi(z) = \begin{cases} z & \text{if } z > 0, \\ \alpha(e^z - 1) & \text{if } z \leq 0, \end{cases} \tag{2.9}$$

where $\alpha \in \mathbb{R}_{>0}$.

2.1.2 1D-CNNs: Backpropagation Stage

The layers of the network collectively transform the input x into a non-linear output $\hat{f}(x; \mathbf{W}, \mathbf{b})$, where $\mathbf{W}$ and $\mathbf{b}$ denote the set of filters/weights and biases, respectively. For each batch of T training examples $\{(x_i, y_i)\}_{i=1}^T$, $\mathbf{W}$ and $\mathbf{b}$ can be estimated by minimising the *empirical loss* between predictions and targets:

$$\hat{\mathbf{W}}, \hat{\mathbf{b}} = \text{argmin}_{\mathbf{W},\mathbf{b}} \frac{1}{T} \sum_{i=1}^{T} \|y_i - \hat{f}(x_i; \mathbf{W}, \mathbf{b})\|^2, \tag{2.10}$$

where regularisation is neglected for simplicity. During training, a suitable gradient-based optimiser is used to minimise the loss in Equation (2.10). Using *backpropagation*, the optimiser propagates the loss function backwards through the network, computing gradients of the loss with respect to each parameter via the chain rule. These gradients indicate the direction and magnitude by which each parameter should be adjusted to reduce the loss, enabling iterative updates until convergence. In this work, we use the *Adam* optimiser [36], which is particularly well-suited for optimising loss where the underlying datasets are large and/or the parameter spaces are high-dimensional. With low memory requirements, Adam efficiently minimises the objective function by leveraging adaptive estimates of first- and second-order moments to dynamically adjust learning rates [36].

Although 1D-CNNs have achieved state-of-the-art performance, particularly in computer vision tasks, their computational cost poses a challenge. Due to their high parameter count, these models typically require large volumes of labelled training data, which limits their applicability in settings where datasets are small [37, 38, 39] or where annotations are scarce, expensive or even unavailable [40].

2.2 Self-supervised Learning

An alternative to supervised learning, SSL has gained increasing attention for its ability to efficiently train large-scale models without explicit labels. Instead, SSL leverages the data *itself* as supervision by revealing complex structures and co-occurence relationships between data components or augmented views of the same instance [41]. As summarised in [42], SSL methods can be broadly classified into three paradigms:

- **Contrastive**: trains an encoder to map input pairs (x_1, x_2) to embeddings (z_1, z_2) to measure the similarity between them.
- **Generative**: trains an encoder-decoder pair to map input x to an embedding z and then reconstruct x from z.
- **Adversarial**: trains an encoder-decoder pair to generate false samples and a discriminator to distinguish them from real samples.

In this work, we focus exclusively on contrastive SSL algorithms, which have recently outperformed even SL [43, 44, 45].

2.2.1 Contrastive Learning

As defined in [46], contrastive learning differentiates between semantically similar (positive) and dissimilar (negative) pairs of augmented views of data points, encouraging representations of positive pairs to be mapped closer together in latent space, while pushing those of negative pairs apart These augmentations preserve the *semantic content* of the original observation while modifying other 'nuisance' aspects, promoting the learning of invariant features that enhance generalisation to downstream tasks [47, 48]. As an example, Contrastive Language-Image Pretraining (CLIP) [49] trains encoders to contrast images with their corresponding language-based descriptions, where the image is augmented with random square crops and the text with bi-grams with high pointwise *mutual information* (MI) for the pair.

To formalise this, let $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$ be the input space of two different modalities. Consider a batch of N unlabelled samples $\{(x_1, y_1), ..., (x_N, y_N)\} \sim \mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$, where (x_i, y_i) is semantically related, e.g., a NIRSpec spectrum and its APOGEE pair. Apply random augmentations $A(\cdot)$ and $\tilde{A}(\cdot)$ to inputs x_i and y_j, respectively. CLIP's objective is to learn a pair of encoders $f_\Phi : \mathcal{X} \to \mathbb{R}^d$ and $g_\Theta : \mathcal{Y} \to \mathbb{R}^d$, $d \ll \min(n, m)$, that compress the two modalities from raw space into a shared latent space, such that $z_i := f_\Phi(A(x_i))$ and $z'_j := g_\Theta(\tilde{A}(y_j))$ are close to each other if they share 'similar' semantics, otherwise far away. For a given x_i, we refer to z_i as the *anchor* and to z'_j as a *positive* sample if $i = j$ and as a *negative* sample otherwise.

The standard loss used to train these encoders is the Information Noise Contrastive Estimation (InfoNCE) [50]:

$$\mathcal{L}_{\text{InfoNCE}}(f_\Phi, g_\Theta, \tau) = -\mathbb{E}\left[\log \frac{\exp(s_{ii}/\tau)}{\exp(s_{ii}/\tau) + \sum_{j\neq i}^{N} \exp(s_{ij}/\tau)}\right]. \tag{2.11}$$

Here, $\tau \in \mathbb{R}_{>0}$ denotes a trainable temperature parameter, discussed in §2.2.2, and $s_{ij} := s(z_i, z'_j)$ is a similarity measure between the features z_i and z'_j. In our setting, we use cosine similarity:

$$s_{ij} := s(z_i, z'_j) = \frac{z_i^\top z'_j}{\|z_i\|^2 \|z'_j\|^2}. \tag{2.12}$$

Minimising Equation (2.11) encourages alignment for for positive pairs (z_i, z_i') while penalising similarity with negative ones $(z_i, z_j'), j \neq i$.

Of note, InfoNCE has an insightful connection to MI. A formal proof given by [50] and [51] shows that, for any two vectors $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$, with representations z_i and z_j', respectively,

$$I(z_i; z_j') \geq \log(k) - \mathcal{L}_{\text{InfoNCE}}, \tag{2.13}$$

where k is the number of negative samples associated to a given sample x_i. This shows that minimising $\mathcal{L}_{\text{InfoNCE}}$ maximises the variational lower bound on the MI $I(z_i; z_j')$, which is itself bounded above by $I(x_i; y_j)$ through the data-processing inequality. Moreover, the log-dependency on k implies that having more negative samples leads to improved representations, a fact formally proved by Tian et al. [52]. Altogether, InfoNCE provides a good lower bound approximation to MI, enjoying greater stability and lower variance than most other competing approaches [53].

2.2.2 Role of Temperature in Contrastive Learning

Intuitively, the temperature τ controls the sensitivity of the InfoNCE loss. A smaller τ tends to penalise much more on *hard*[8] negative samples (HNS), pushing their latent features further away from those of the anchor [54]. As a result, the local structure of each anchor tends to be more separated, and the embedding distribution appears to be more uniform [54]. Conversely, a larger τ reduces the loss' sensitivity to HNS, resulting in more compact, clustered latent features [55]. In practice, τ has been shown to crucially impact the quality of a model's learned embeddings (e.g., [13, 56, 57]).

The phenomenon of larger τ inducing semantic structure in representation space has been demonstrated empirically (e.g., [13, 57]), but is otherwise poorly understood. To gain a little insight, we follow the setup of [58] and perform a change-of-variables in Equation (2.11), in which similarities s_{ij} are replaced by 'distances' d_{ij}:

$$d_{ij} = \frac{1 - s_{ij}}{\tau} \tag{2.14}$$

$$c_{ii} = \exp(d_{ii}), \tag{2.15}$$

where it is clear $0 \leq d_{ij} \leq 2/\tau$. This allows us to rewrite each summand, $\mathcal{L}_c^i$, of the InfoNCE loss as:

$$\mathcal{L}_c^i = -\log\left[\frac{\exp(-d_{ii})}{\exp(-d_{ii}) + \sum_{j\neq i}^{N}\exp(-d_{ij})}\right] = \log\left[1 + c_{ii}\sum_{j\neq i}^{N}\exp(-d_{ij})\right]. \tag{2.16}$$

Hence, the loss increases monotonically with the sum of exponentiated distances $S_i := \sum_{j\neq i}^{N}\exp(-d_{ij})$. This means it is enough to examine the behaviour of S_i in the regimes of small and large τ:

- **Small** τ. When $\tau \ll 1$, the dominant contributions to S_i arise from distances d_{ij} where $s_{ij} \sim 1$, corresponding to HNS of the anchor. As a result, the loss function can be interpreted as maximising the average distance to these HNS, encouraging the model to push embeddings further apart. This leads to a more uniform and isotropic distribution in embedding space.

[8] A *hard negative sample* (HNS) is defined as a negative sample with high cosine similarity to the anchor; consequently, it appears as a *nearest neighbour* to the anchor. Conversely, *easy negative samples* (ENS) are easily distinguishable from the anchor by simple patterns and are therefore more distant in embedding space.

- **Large** τ. When $\tau \geq 1$, say, all distances d_{ij} are of comparable magnitude, contributing roughly equally to S_i. Hence, the loss function can be intrepreted as maximising the average distance over a broader range of neighbours. Since the number of ENS typically exceeds the number of HNS, ENS collectively have a greater influence on the loss. Rather than learning 'hard' features that enable better *instance-discrimination* among HNS, the model is instead encouraged to learn 'easier' patterns that allow for better *group-wise discrimination*. With this biased approach, the model tends to increase the margin between clusters of samples, inducing semantic structure in representation space.

3 Data[9]

3.1 APOGEE Spectra

We use the spectra and stellar parameters from the final data release of the APOGEE survey [11] in SDSS-IV DR17 [10, 25]. All observations are captured with two high-resolution APOGEE spectrographs [59]: one commissioned on the 2.5 m Sloan Foundation telescope [60] at Apache Point Observatory (APO), and another on the 2.5 m Irénée du Pont telescope [61] at Las Campanas Observatory (LCO). Both spectrographs can measure up to 300 objects simultaneously within 3° and 2° diameter FOVs, respectively, and operate in the near-IR H-band (1.51-1.7 μm) at a high spectral resolution of $R \sim 22,500$ [11, 62, 59]. Throughout the survey, APOGEE preferentially targets RGB stars and other luminous post-main-sequence stars across the Milky Way (MW), particularly in heavily dust-obscured parts of the Galactic bulge, disc and halo [63]. The automated data-reduction pipeline for APOGEE is detailed in [64]; briefly, it calibrates the raw data and subsamples them onto a common wavelength grid, with $\Delta\lambda/nR$, where $n = 8,575$ is the number of pixels per resolution element. The final wavelength grid covers $\lambda_{\text{min.}} = 15,100\mathring{A}$ to $\lambda_{\text{max.}} = 17,000\mathring{A}$.

APOGEE's stellar parameters and chemical abundances are derived from the APOGEE Stellar Parameter and Chemical Abundances Pipeline (ASPCAP) [65]. ASPCAP conducts searches in a 7D synthetic spectral grid, whose dimensions cover the range of stellar parameters over which most surveyed stars are expected to be found [66]. Given the region of the H-band where APOGEE spectra are recorded, these dimensions include effective temperature (T_{eff}), surface gravity ($\log g$), microturbulent velocity (v_{micro}), overall scaled-solar chemical abundance ([M/H]), carbon ([C/M]), nitrogen ([N/M]) and α-element abundances ([α/M]) relative to the solar abundance ratio [66]. The synthetic spectral library is generated using the 1D local thermodynamic equilibrium (LTE) branch of the `ATLAS9`/ASSϵT synthesis code, written and maintained by R. Kurucz [32, 33]. Using the FORTRAN95 code `FERRE` [67], observed APOGEE spectra are run through all synthetic grids and ASPCAP adopts the set of stellar parameters that best fit the observations via χ^2-minimisation. Wherever ASPCAP results are not reliable, a wrapper sets warning bits in appropriate data quality flags (see [66] for further details). In particular, stars whose derived parameters lie at spectral grid edges may have true parameters outside the grid, resulting in less precise stellar parameter estimation.

To minimise the propagation of ASPCAP errors, stars flagged for having unreliable chemical abundance measurements are removed from our dataset. Our dataset is further streamlined to high-quality, RGB-like stars that safely lie within grid edges, for which SNR > 200, $4000 < T_{\text{eff.}} < 5000$ K, $0.5 < \log(g) < 2.0$ and $v_{\text{scatter}} < 1.0$ km/s. Notably, spectra with high v_{scatter} likely correspond to unresolved binary stars [64], where duplicated or blended spectral lines may degrade ASPCAP abundance measurements. By applying these selection criteria to the full APOGEE survey, we obtain a refined sample of 19,001

[9] All datasets have been provided via private communication by Dr Nathan Sandford (nathan.sandford@utoronto.ca).

stars, denoted $\mathcal{A}$. Figure 3 shows a Kiel diagram, i.e., the $\log(g)$ versus $T_{\text{eff.}}$ plane, for the APOGEE spectra in $\mathcal{A}$.

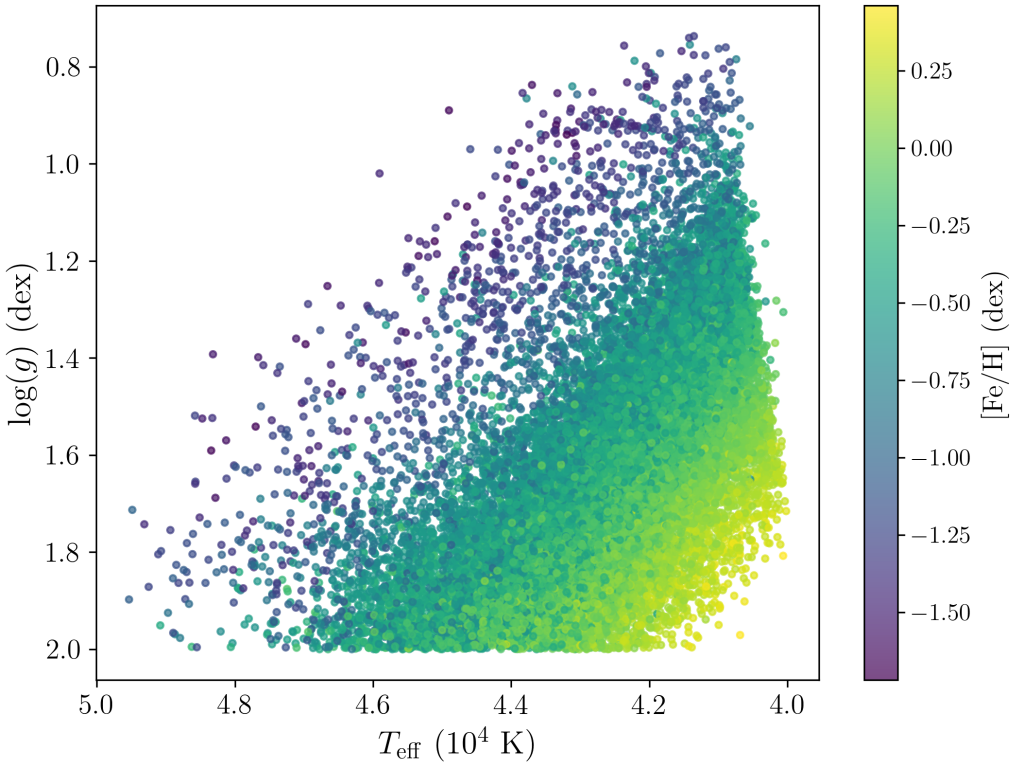


Figure 3: Kiel diagram of qualified APOGEE spectra, whose parameters are derived from ASPCAP. Note different colours indicate different values of [Fe/H] metallicity.

We split $\mathcal{A}$ into training, validation and test sets in a 5:1:2 ratio. This split is chosen to maximise the number of stars in the training set, while having at least 2,000 stars in the validation and tests sets to be representative of the full stellar parameter range.

3.2 JWST/NIRSpec Spectra

3.2.1 *Ab initio* Data

Ab initio are generated using the same method as described in Ting et al. [68] and Sandford et al. [69]. Briefly, 1D LTE model atmospheres are computed using the `ATLAS12` program, developed and maintained by R. Kurucz [26, 27, 28, 29, 30, 31]. Each model adopts one of 19,001 sets of APOGEE stellar parameters, along with solar abundances obtained from Asplund et al. [70]. Convection is modelled according to standard 1D mixing length theory (MLT) with $\alpha = 1.25$ and no overshooting. Spectra are computed with the `SYNTHE` program, also due to R. Kurucz, at a nominal resolution of $R = 300,000$. They are subsequently continuum-normalised, convolved down to the average resolution of NIRSpec ($R = 2,700$) and subsampled onto a common wavelength grid with $\Delta\lambda/nR$, where $n = 8,192$ is the number of pixels per resolution element. The final wavelength grid covers $\lambda_{\text{min.}} = 9,000\mathring{A}$ to $\lambda_{\text{max.}} = 18,000\mathring{A}$. We denote the set of synthetic NIRSpec spectra as $\mathcal{J}$, which we split into training, validation and test sets in a 5:1:2 ratio.

In this way, we obtain a database $\mathcal{J} \times \mathcal{A}$, consisting of 19,001 paired NIRSpec and APOGEE spectra,

each characterised by a common set of stellar labels.

3.2.2 Observed Data

We use a small sample ($n = 59$) of representative traces on the NIRSpec detector, located in an orbit near the second Lagrange Point (L2). The multi-object spectroscopic (MOS) observations are collected as part of the JWST-GO-3788 program (PI: Wiesz), using microshutter array (MSA) in a 3.6×3.4 arcmin^2 FOV over near-IR wavelength ranges (0.97-1.82 μm) at a resolution $R = 2,700$ [2]. Targets are predominately stars on the upper RGB branch in three isolated dwarf galaxies—Tucana, Leo A and IC 1613—for which [α/Fe] and [Fe/H] can be measured to 0.1-0.3 dex precision.

As shown in Figure 4(a), the MSA occupies NIRSpec's slit plane in a 2×2 mosaic of micro-shutter quadrants. It is further mounted on the detector plane, which consists of two adjacent detector arrays, NRS1 and NRS2, seprated by a 2.8 mm horizontal gap in the direction of dispersion. As shown in Figure 4(b), the presence of this gap causes portions of NIRSpec spectra to be lost if they happen to project onto it.

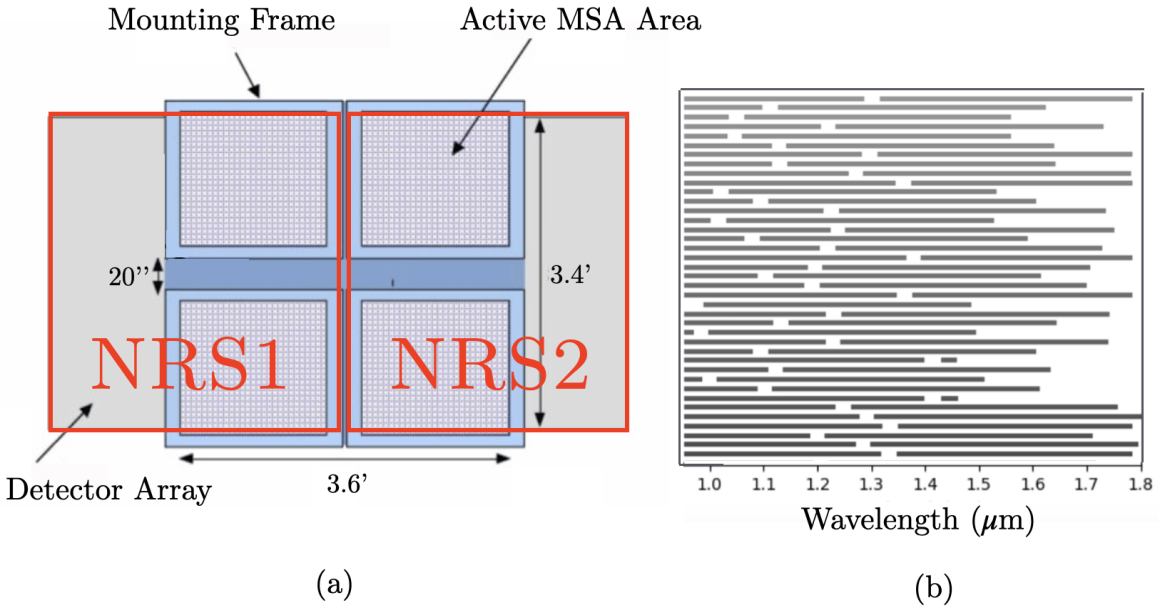


Figure 4: (a) Layout of the NIRSpec slit plane, adapted from [71]. The four MSA quadrants covering 3.6×3.4 arcmin^2 on the sky, are mounted via a frame onto the detector plane. The two large squares mark the outer boundaries of the active areas of the two detector arrays, NRS1 and NRS2. The dispersion direction is horizontal. (b) Sample representative spectral traces on the NIRSpec detector, adapted from [72]. Each trace, widened for clarity, exhibits missing wavelength segments lost to the detector gaps.

4 Model Implementation

We present a two-step procedure to train our cross-modal spectral encoders *in silico*:

(i) We pre-train single-modal CNNs in a supervised setting to predict stellar labels from spectroscopic data, using a modified StarNet [5].

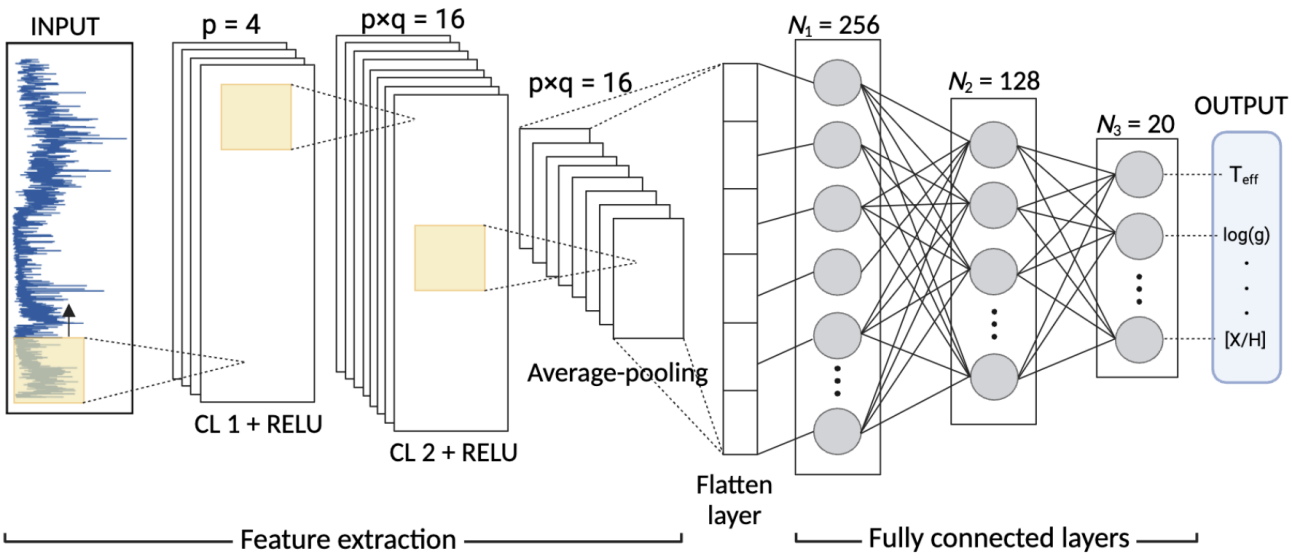


Figure 5: Schematic of the CNN architecture used to predict stellar properties from single-modal spectra. The input layer is followed by pooling layers with 4 and 16 filters, respectively. An average-pooling layer with a window length of 4 is applied next, followed by three fully connected layers with 256, 128, and 20 nodes. The final output layer provides the predicted stellar properties: $T_{\rm eff}, \log g$ and 18 elemental abundances [X/H].

(ii) We fine-tune our pre-trained CNNs in a contrastive setting, aligning their cross-modal embeddings in a shared latent space, using a modified AstroCLIP [22].

Notably, our strategy to initialise CLIP with pre-trained models–rather than to train the entire SSL framework from scratch—has been shown to stabilise training and reduce the computational costs associated with CLIP [73]. Further details on the implementation and training of these models are provided below.

4.1 Spectral Encoder Model

4.1.1 Implementation

Our spectral encoder is closely modelled after StarNet [5]. In particular, the CNN model we implement has $N_{\rm CL} = 2$ convolutional and $N_{\rm FCL} = 3$ fully-connected layers, which together perform stellar property prediction of 20 stellar parameters.

Figure 5 shows the sequential operations required to transform single-modal spectra to a stellar property prediction. The leftmost block represents the input spectra, either obtained from $\mathcal{A}$ or $\mathcal{J}$. The first operation applied to the input is a 1D-convolution using 1 input channel and $p = 4$ output channels, corresponding to the number of learnable filters of size $s^{(1)} = 8$ (and stride = 1). An additional 1D-convolution is performed on each channel of the previous layer, resulting in $p \times q = 16$ output channels, using 16 learnable filters of size $s^{(2)} = 8$. After each convolution, the ELU activation function is applied to introduce non-linearities into the model.

The output channels produced by the second convolution are then downsampled via average-pooling, with a window length of $w = 4$, and flattened to a single dimension. The flattened vectors are then passed to the fully connected part of the network. Here, they are mapped to three multiperceptron

(MLP) layers with $N_1 = 256$, $N_2 = 128$ and $N_3 = 20$ hidden units, successively, each with their own learnable sets of weights and biases. After the first two MLP layers, ELU activation is applied. After the last MLP layer, the model outputs the stellar property estimation $\hat{Y}$.

Though we adopt the model architecture and hyperparameter selection (p, q, w, N_1, N_2) from [5], we distinguish our work in the following ways:

- We use ELU rather than ReLU (or sigmoid or tanh), having found that it promotes training stability and enables convergence to lower loss values.
- We use average-pooling rather than max-pooling, having found that it better preserves features relevant to regression.
- We introduce a learning rate scheduler to reduce training time.
- We implement an early stopping criterion based on the validation loss to mitigate overfitting.
- We predict $N_3 = 20$ rather than 3 stellar parameters with high precision and accuracy.

4.1.2 Training Procedure

We implement our model architecture and training procedure in `PyTorch` [74]. For our training task, which is to regress stellar properties, the loss function we optimise is the MSE:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2, \tag{4.1}$$

where N_{batch} is the number of spectra in the batch, and Y_i and $\hat{Y}_i$ denote the true and predicted stellar property of interest, respectively. Note each Y_i is Z-score normalised to have zero mean and unit variance, ensuring that all properties are roughly equally weighted in $\mathcal{L}_{\text{MSE}}$.

We train two single-modal encoders, f_ϕ and g_θ, on APOGEE $\mathcal{A}$ and NIRSpec $\mathcal{J}$ data, respectively. For each encoder we train, the corresponding training data are batched into sets of $N_{\text{batch}}^{\text{train}} = 64$ stars. While the model weights ϕ (or θ) are randomly initalised, the network implements a training procedure to update these weights, as follows (see [6]):

(1) forward pass the batch through the network to compute $\hat{Y}_{\text{batch}}$,
(2) compute $\mathcal{L}_{\text{MSE}}$ according to Equation (4.1),
(3) backpropagate $\mathcal{L}_{\text{MSE}}$ through each layer of the network,
(4) compute the gradient $\Delta_\phi \mathcal{L}_{\text{MSE}}$ (or $\Delta_\theta \mathcal{L}_{\text{MSE}}$),
(5) update the weights to minimise $\mathcal{L}_{\text{MSE}}$.

Steps (1) through (5) are repeated for each training batch iteration, resulting in a steady convergence of the training loss over $N_{\text{epochs}} = 15$ or until an early stopping criterion is met. Specifically, step (5) is carried out using `PyTorch`'s AdamW optimiser ($\eta_0 = 0.001$), with a StepLR scheduler ($T_{\text{step}} = 5$, $\gamma = 0.1$). Altogether, for 11,875 training spectra of either modality and 8.4 million NN parameters, training proceeded in $N_{\text{steps}} = 2,775$ steps and converged in ∼20 seconds, using 1 NVIDIA TITAN Xp GPU.

During training, we also monitor the validation loss. Specifically, at the start of each training epoch, we carry out steps (1) and (2) on the validation set of 2,375 spectra, processed in batches of size $N_{\text{batch}}^{\text{val.}} = 128$. This provides a *diagnostic* of how generalisable the model is to unseen data. To mitigate overfitting, we also implement an early stopping criterion with the following policy: Should the validation loss not improve for $N_{\text{stop}} = 3$ consecutive epochs within a tolerance of $\epsilon_{\text{tol}} = 10^{-3}$, training is halted. Future work could explore performing grid searches over a heuristic choice of hyperparameters

$\{(\eta_0, T_{\text{step}}, \gamma)\}$, choosing the combination that yields the best performance on the validation set—a clear improvement over choosing the hyperparameters *ad hoc*.

All model and training parameters are summarised in Table 1.

Table 1: Model and training parameters for CNN Encoders

	Parameter	Setting(s)	Description
Architecture	n	8,575/8,192	number of pixels per resolution element in $\mathcal{A}/\mathcal{J}$
	N_{CL}	2	number of convolutional layers (CL)
	p	4	number of output channels of CL1
	$s^{(1)}$	8	filter size of CL1
	$p \times q$	16	number of output channels of CL2
	$s^{(2)}$	8	filter size of CL2
	stride	1	convolution stride of both CLs
	w	4	width of pooling kernel
	N_{FCL}	3	number of fully connected layers (FCL)
	N_1	256	number of hidden units in FCL1
	N_2	128	number of hidden units in FCL2
	N_3	20	number of hidden units in FCL3
	ϕ	ELU	activation function
Optimisation	optimiser	AdamW	—
	η_0	10^{-3}	base learning rate (LR)
	T_{step}	5	LR decay period/step size
	γ	10^{-1}	LR decay multiplicative factor
	$\mathcal{L}_{\text{MSE}}$	MSE	loss function
Training	$N_{\text{batch}}^{\text{train}}$	64	training batch size
	N_{epochs}	15	number of training epochs
	N_{steps}	2,775	number of training steps
	N_{stop}	3	number of epochs to halt training if no improvement
	ϵ_{tol}	10^{-2}	early stopping tolerance

4.1.3 Model Evaluation

Each single-modal encoder is evaluated on its corresponding held-out test set of 4,751 spectra, batched into sets of $N_{\text{batch}}^{\text{test}} = 128$. The residuals between true and predicted targets are then computed, as shown in Figure 7. To quantify the model performance, we compute three evaluation metrics: the coefficient of determination r^2, the bias Δ and the root-mean-squared error (RMSE), respectively given as:

$$r^2 = 1 - \frac{1}{N\sigma^2} \sum_i (Y_i - \hat{Y}_i)^2 \tag{4.2}$$

$$\Delta = \frac{1}{N} \sum_i \left(\hat{Y}_i - Y_i \right) \tag{4.3}$$

$$\text{RMSE} = \left[\frac{1}{N} \sum_i \left(Y_i - \hat{Y}_i \right)^2 \right]^{\frac{1}{2}}, \tag{4.4}$$

where N is the number of observations in the test set and Y and $\hat{Y}$ are the true and predicted targets of the stellar property of interest, respectively. The term σ^2 denotes the variance of $\vec{Y} = (Y_1, ..., Y_N)$. In general, $r^2 \simeq 1$ indicates the model explains most of the variation in Y, while $r^2 \simeq 0$ indicates it captures little to no variation. Additionally, in general, smaller bias and RMSE values indicate better model performance.

4.2 StarCLIP Model

4.2.1 Spectrum Preprocessing

To simulate physical corruptions in observed NIRSpec data, we apply a (sub)set of the following semi-empirical, stochastic transformations to each spectrum in $\mathcal{J}$:

(i) *Fixed-data Substitution.* The Paschen series have been long used for the determination of fundamental parameters of stellar atmospheres [75, 76, 77]. In fact, the Paschen H-lines contain some of the most informative pixels for estimating T_{eff} and $\log g$ (see, e.g., [78]). However, observational challenges, large uncertainties in emission-line measurements and limitations of LTE modelling can potentially introduce important systematic and model-dependent errors near the Paschen lines. As such, a *desideratum* in this work is to reduce the model's sensitivity to potentially poorly modelled or corrupted data near these regions. To this end, we fix an arbitrary spectrum in $\mathcal{J}$ and isolate four 'reference' neighbourhoods of radius $\sim 10\mathring{A}$ centred on the Paschen lines: 9,548.6, 10,052.1, 10,941.1 and 12,821.6 $\mathring{A}$. Given another spectrum in $\mathcal{J}$, we replace its flux values in these neighbourhoods with the reference values just obtained.

(ii) *Pixel Masking.* We select uniformly at random 0.1% of pixels to set to unity[10], roughly the expected fraction of non-operable pixels in the NIRSpec detectors [3].

(iii) *Denormalisation.* Observed spectra are distorted by broad telluric bands, instrumental effects of the spectrograph, and artifacts introduced during the pipeline reduction process (see, e.g., [79]). To emulate imperfections in continuum normalisation, we multiply the given spectrum in $\mathcal{J}$ by a low-amplitude eighth-order polynomial perturbation P, defined as follows:

$$P(\lambda) = 1 + \alpha \left(\frac{p(\lambda') - \mu_p}{\max |p(\lambda') - \mu_p|} \right), \tag{4.5}$$

where

$$p(\lambda) = \sum_{i=0}^{d} c_i (\lambda')^i, \quad \lambda' = \frac{\lambda_i - \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}} \in [0, 1], \quad \mu_p = \frac{1}{n} \sum_{i=1}^{n} p(\lambda_i') \tag{4.6}$$

and $\alpha = 0.01$, $d = 8$, $c_i \sim \mathcal{U}(-1, 1)$, $\lambda_{\min} = 9,000\mathring{A}$, $\lambda_{\max} = 18,000\mathring{A}$, $n = 8,192$. Note the choice of P is *heuristic*, serving as an initial approximation for modelling normalisation errors.

(iv) *Gaussian Noise.* We randomly add Gaussian noise to the spectrum by sampling the noise level from a normal distribution $\mathcal{N}(0, \text{SNR}^{-1})$, where $\text{SNR} \sim \mathcal{U}(30, 300)$, roughly the expected signal-to-noise (SNR) range for NIRSpec detectors [2].

(v) *Wavelength Masking.* We use an automated routine in Python's `OpenCV` [80] to detect gap boundaries in a grayscale image of 59 spectral traces, as exemplified in Figure 4b. Each trace admits two gaps: the first is characterised by its leftmost x-position $x_{\text{gap},1}$ and width $w_{\text{gap},1}$, while the second is defined solely by its width $w_{\text{gap},2}$, as its rightmost endpoint is fixed at $\lambda_{\max} = 18,000\mathring{A}$. To model the statistical distribution of these gaps, we fit nonparametric distributions to the empirical measurements $\{x_{\text{gap},1}\}$, $\{w_{\text{gap},1}\}$ and $\{w_{\text{gap},2}\}$ using Gaussian kernel density estimation

[10] We choose unity rather than to 0 (or NaN), so to help mitigate discontinuities in the flux profile of the data.

(KDE). When sampling new x-positions and widths of gaps from the KDEs, we apply a ReLU transformation to enforce non-negativity:

$$x'_{\text{gap},1} = \max(0, x_{\text{gap},1}) \tag{4.7}$$

$$w'_{\text{gap},i} = \max(0, w_{\text{gap},i}), \tag{4.8}$$

$i = 1, 2$. The sampled gaps are then applied to a given spectrum in $\mathcal{J}$, replacing the corresponding flux values within these missing intervals with unity.

The mock dataset $\mathcal{J}'$ is created by replicating each entry in $\mathcal{J}$ five times and augmenting each instance with observational signatures (iv) and (v). Its paired dataset, $\mathcal{A}'$, is constructed by similarly replicating each entry in $\mathcal{A}$ five times without applying further transformation. This gives an augmented dataset $\mathcal{J}' \times A'$ of 95,005 paired mock NIRSpec-APOGEE spectra. We split $\mathcal{J}' \times A'$ into training, validation and test subsets in a 5:1:2 ratio at the *instance-group level*, ensuring all augmented views of each original spectrum remain within the same subset.

To mimic the low-data regime of JWST, we partition $\mathcal{J}$ into 30 bins based on [Fe/H] values, where the number of bins is computed with Freedman-Diaconis' rule [81]. We randomly select a small subset of $n = 250$ spectra from $\mathcal{J}$, whose [Fe/H] values are distributed as uniformly as possible across all bins. Applying transformations (iv) and (v) to this subset gives the mock sample $\mathcal{J}'_{\text{m}}$; applying all transformations from (i) to (v) gives the extended-mock sample $\mathcal{J}'_{\text{e-m}}$. We split $\mathcal{J}'_{\text{m}} \times \mathcal{J}'_{\text{e-m}}$ into training, validation and test sets in a 5:1:2 ratio.

We further enrich the test split of $\mathcal{J}'_{\text{e-m}}$ by generating an additional 5,000 extended-mock spectra, whose [Fe/H] values remain approximately uniformly distributed across all bins. The resulting augmented test split, now comprising 5,062 spectra, reduces sparsity in visualisations and enhances the representativeness of the test set across the entire stellar parameter range.

Figure 6 illustrates the complete sequence of stochastic transformations (i)-(v) applied to a NIRSpec spectrum in $\mathcal{J}'_{\text{e-m}}$. It provides a close view of the fixed-data substitution process, described in (i), demonstrating how spectral regions near Paschen lines are replaced by fixed values from a reference spectrum in $\mathcal{J}$.

4.2.2 Implementation

Our CLIP models are closely modified versions of AstroCLIP [22]. As illustrated in Figure 1, each network is a sort of *compositional* model, consisting of a pair of cross-modal spectral encoders with 20 embedding dimensions. The output of each network is a pair of cross-modal embeddings that reside in a shared, CLIP-aligned latent space.

We train two versions of the model. The first, StarCLIP, employs pre-trained encoders f_θ and g_ϕ to initialise the mapping of $\mathcal{J}'$ and $\mathcal{A}'$ into latent space. Both encoders remain unfrozen, allowing StarCLIP to align the representation spaces of the two modalities (see, e.g., [82]). After training, the initial weights θ and ϕ are updated to learned weights Θ and Φ. The second version, MockStarCLIP, uses a pair of identically initialised encoders, both denoted f_Θ, to map $\mathcal{J}'_{\text{m}}$ and $\mathcal{J}'_{\text{e-m}}$ into the *same* latent space. While the encoder for $\mathcal{J}'_{\text{m}}$ is frozen, the fully-connected part of the encoder for $\mathcal{J}'_{\text{e-m}}$ is unfrozen, enabling MockStarCLIP to be fine-tuned on physical corruptions in the extended-mock data. After training, the learned weights of this latter encoder are thus updated from Θ to Θ^*.

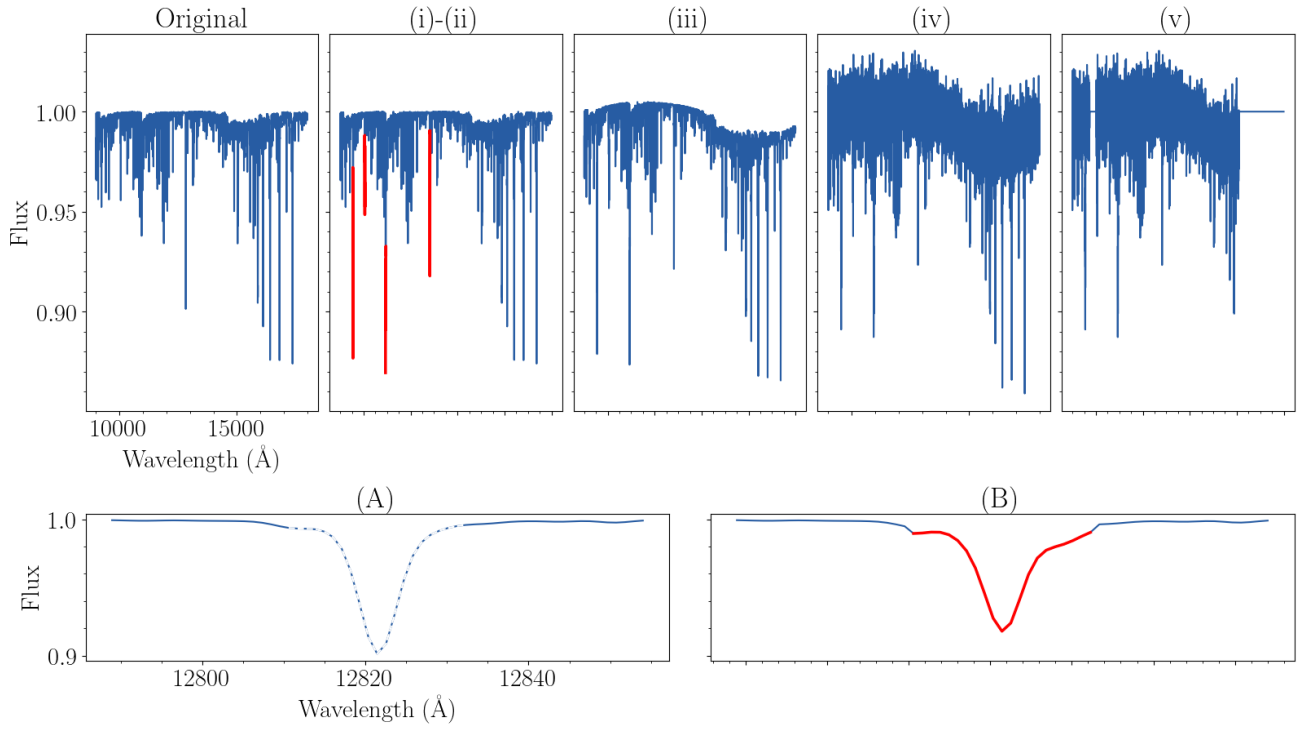


Figure 6: Sequence of stochastic transformations applied to a representative spectrum from $\mathcal{J}$, mapping it to an element of $\mathcal{J}'_{\text{e-m}}$. The **top panels** show the original spectrum, successively transformed through (i)-(ii) fixed-data substitution and pixel masking, (iii) denormalisation, (iv) Gaussian noise addition, and (v) wavelength masking. The red lines in panel (i)-(ii) indicate the four neighbourhoods substituted using the arbitrarily-chosen 'reference' spectrum in $\mathcal{J}$, thickened for clarity. The **bottom panel** (A) highlights the spectrum's original neighbourhood of radius $\sim 10\mathring{A}$ centred on the Paschen line 12,821.6$\mathring{A}$, marked with dashed lines. Panel (B) shows the same neighbourhood after applying (i)-(ii), with the substituted region from the reference spectrum highlighted in red.

4.2.3 Training Procedure

We implement our training procedure in `PyTorch` [74]. We first batch the training splits of $\mathcal{J}' \times \mathcal{A}'$ ($\mathcal{J}'_{\text{m}} \times \mathcal{J}'_{\text{e-m}}$) into mini-batches of $N^{\text{train}}_{\text{batch}} = 16$ (8) pairs of stars. Optimisation is performed using the Adam optimiser with base learning rates of $\eta_0 = 1 \times 10^{-4}$ (1×10^{-3}), controlled by a cosine annealing scheduler and weight decay parameter of $\lambda = 0.2$ (0). We train our models for $N_{\text{epochs}} = 30$ on 1 NVIDIA TITAN Xp GPU, which results roughly in $T_{\text{train}} \sim 1$ h (~ 20 min.) of training time.

Similar to the findings in Parker et al. [22], we observe that StarCLIP performs best when the value of τ in Equation (2.11) is fixed to 15.5. In contrast, following the approach of Radford et al. [49], we find MockStarCLIP achieves better performance when τ is treated as a learnable parameter in gradient descent, initialised to 0.07—the default starting value in CLIP. In view of §2.2.2, these choices of τ are well-motivated: augmented views with frequent semantics in $\mathcal{J}' \times \mathcal{A}'$ benefit from a larger τ to preserve local semantic structure, whereas a smaller τ would undesirably push semantically consistent samples apart. Conversely, the samples in $\mathcal{J}'_{\text{m}} \times \mathcal{J}'_{\text{e-m}}$, which exhibit rarer semantics, benefit from a smaller τ to make their features more discriminative and separable. By computing Equation (2.11) between all cross-modal pairs in $\mathcal{J}' \times \mathcal{A}'$ and $\mathcal{J}'_{\text{m}} \times \mathcal{J}'_{\text{e-m}}$, we maximise the similarity of embeddings corresponding to (augmented views of) the same spectrum while minimising similarity with embeddings of other spectra.

All model and training parameters are summarised in Table 2.

Table 2: Model and training parameters for CLIP models

	Parameter	Value(s)/Setting(s)		Description
		StarCLIP	MockStarCLIP	
Architecture	$\mathcal{X} \times \mathcal{Y}$	$\mathcal{I}' \times \mathcal{A}'$	$\mathcal{I}'_{\text{m}} \times \mathcal{I}'_{\text{e-m}}$	cross-modal input data
	$\mathcal{W}_{\text{init},1}$	f_θ	f_Θ	encoder 1 weight init.
	$\mathcal{W}_{\text{init},2}$	g_ϕ	f_Θ	encoder 2 weight init.
	τ	15.5 (fixed)	0.07 (learnable)	temperature parameter
Optimisation	optimiser	Adam	Adam	—
	η_0	10^{-4}	10^{-3}	base learning rate (LR)
	λ	0.2	0	LR weight decay
	scheduler	Cosine Annealing	Cosine Annealing	—
	T_{max}	80	80	max number of iterations
	η_{min}	5×10^{-6}	5×10^{-6}	minimum LR
	$\mathcal{L}$	InfoNCE	InfoNCE	loss function
Training	$N_{\text{batch}}^{\text{train}}$	16	8	training batch size
	N_{epochs}	30	30	number of training epochs

4.2.4 Model Evaluation

We embed the spectra in the held-out test sets, generating discriminative representations that generalise to downstream tasks without requiring model fine-tuning. To evaluate the effectiveness of similarity search, we adopt *ad hoc* versions of two widely used evaluation metrics in information retrieval: *SuccessRate@k* and Mean Reciprocal Rank (MRR) [83, 84, 85]. In our work, *SuccessRate@k* measures the proportion of queries for which the *true* cross-modal pair or any of its augmented views appear in the top k ranked results. Specifically,

$$SuccessRate@k = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \delta(FRank_q \leq k), \tag{4.9}$$

where Q is a set of tested queries and $\delta(\cdot)$ is the indicator function, and $FRank_q$ [86] denotes the rank of the first correct match (i.e., the true cross-modal pair or any of its augmented views) in the result list for query q. We report *SuccessRate@k* for $k = 1, 5, 10$ and 50, reflecting the typical sizes of results users consider in retrieval tasks [85]. The MRR is the average of the reciprocal ranks of results of Q, where the reciprocal rank is the inverse of the rank of the *FRank*. Thus, the formula is:

$$\text{MRR} = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{|FRank_q|}. \tag{4.10}$$

In general, higher *SuccessRate@k* and MRR values imply better search code performance. To assess the accuracy of stellar property inference, we compute the three evaluation metrics in Equations (4.2) to (4.4).

5 Results

5.1 CNN Stellar Property Recovery

We evaluate the performance of the NIRSpec and APOGEE CNN models in recovering 20 stellar parameters from 2,375 test spectra in $\mathcal{J}$ and $\mathcal{A}$, respectively. Figure 7 shows the predictions for *six* stellar properties derived from both CNNs, including three atmospheric parameters ($T_{\rm eff}, \log(g)$, [Fe/H]), along with three chemical abundances [α/H] selected to represent high, median and low RMSE scores. Following the approach of [6], the top panel in Figure 7 shows model-based predictions against the true targets, while the bottom panel displays residuals, highlighting regions where the models are biased. In this panel, we also mark the region within 1 standard deviation of a perfect prediction. For completeness, Table 4 in the Appendix reports the evaluation metrics $\{r^2, \Delta, \text{RMSE}\}$ for all 20 stellar parameters derived from both CNNs. It is important to recall these parameters are not independent but exhibit intrinsic degeneracies, as exemplified by the correlation between $T_{\rm eff}$ and $\log g$ in Figure 3.

As shown in Table 4, both CNNs recover all stellar properties well, with r^2 scores exceeding ~ 0.88 and RMSE precisions *comparable* to—if not better than—the estimated precisions of the SDSS-IV DR17 APOGEE/APSCAP measurements [87]. In particular, the NIRSpec (APOGEE) model predicts $T_{\rm eff}, \log g$ and metallicities to RMSE precisions of 280 (270) K, 0.09 (0.09) dex and $\lesssim 0.1$ dex, respectively. These are in strong agreement with the estimated precisions of calibrated ASPCAP results, which find $T_{\rm eff}$ is measured with a precision better than 150 K, $\log g$ better than 0.2 dex and metallicites better than 0.1 dex. Moreover, the overall biases are negligible, typically an order of magnitude smaller than the RMSEs. As shown in the bottom panels of Figure 7, most of the model bias occurs at the limits of parameter space, where predictions exhibit 'regression towards the mean', in which low values tend to be over-predicted and high values underpredicted. A potential reason for this behaviour is that stars with extreme values are underrepresented and of lower quality (low SNR, weak spectral features) in our catalogs, making it difficult for the models to learn this region of parameter space sufficiently well at training time. Another contributing factor may be due to degeneracies between data and labels or any discrete, abrupt changes in data-label relationships due to underlying physical processes [6]. To illustrate, for $T_{\rm eff}$ values larger than $\sim 4.5 \times 10^4$ K, the predictions are systematically negatively biased, especially for stars with relatively low SNR. This is likely caused by the reduced number of spectral features in hotter spectra and the increasing degeneracy between temperature and gravity measurements in warmer stars.

In general, the NIRSpec CNN is capable of predicting values closer to the ASPCAP stellar parameters than the APOGEE CNN. This arises from variations in the training data: observed spectra in $\mathcal{A}$ have different shapes and profiles than synthetic spectra in $\mathcal{J}$, characterised by noise, missingness, instrumental broadening, and other signature effects, which *ab initio* models can at most partially capture. As such, the NIRSpec CNN is better poised to capture relevant spectral features in the data, leading to more accurate parameter estimates. In §4.2.1, we describe a pipeline aimed at minimising the *synthetic gap* between the feature distributions of synthetic and observed spectra.

5.2 SSL Downstream Tasks

5.2.1 Cosine Similarity Search

We perform cross-modal cosine similarity search (CrossCSS) on the embedded datasets generated by StarCLIP—i.e., $f_\Theta(\mathcal{J}'_{\rm test}) \times g_\Phi(\mathcal{A}'_{\rm test})$—and MockStarCLIP—i.e., $f_\Theta(\mathcal{J}'_{\rm m,\ test}) \times f_{\Theta^*}(\mathcal{J}'_{\rm e\text{-}m,\ test})$. Given a search query z_q from one modality (e.g., $f_\Theta(\mathcal{J}'_{\rm test})$), CrossCSS computes its cosine similarity with each embedding z'_j from the complimentary modality (e.g., $g_\Phi(\mathcal{A}'_{\rm test})$). The results are ranked by

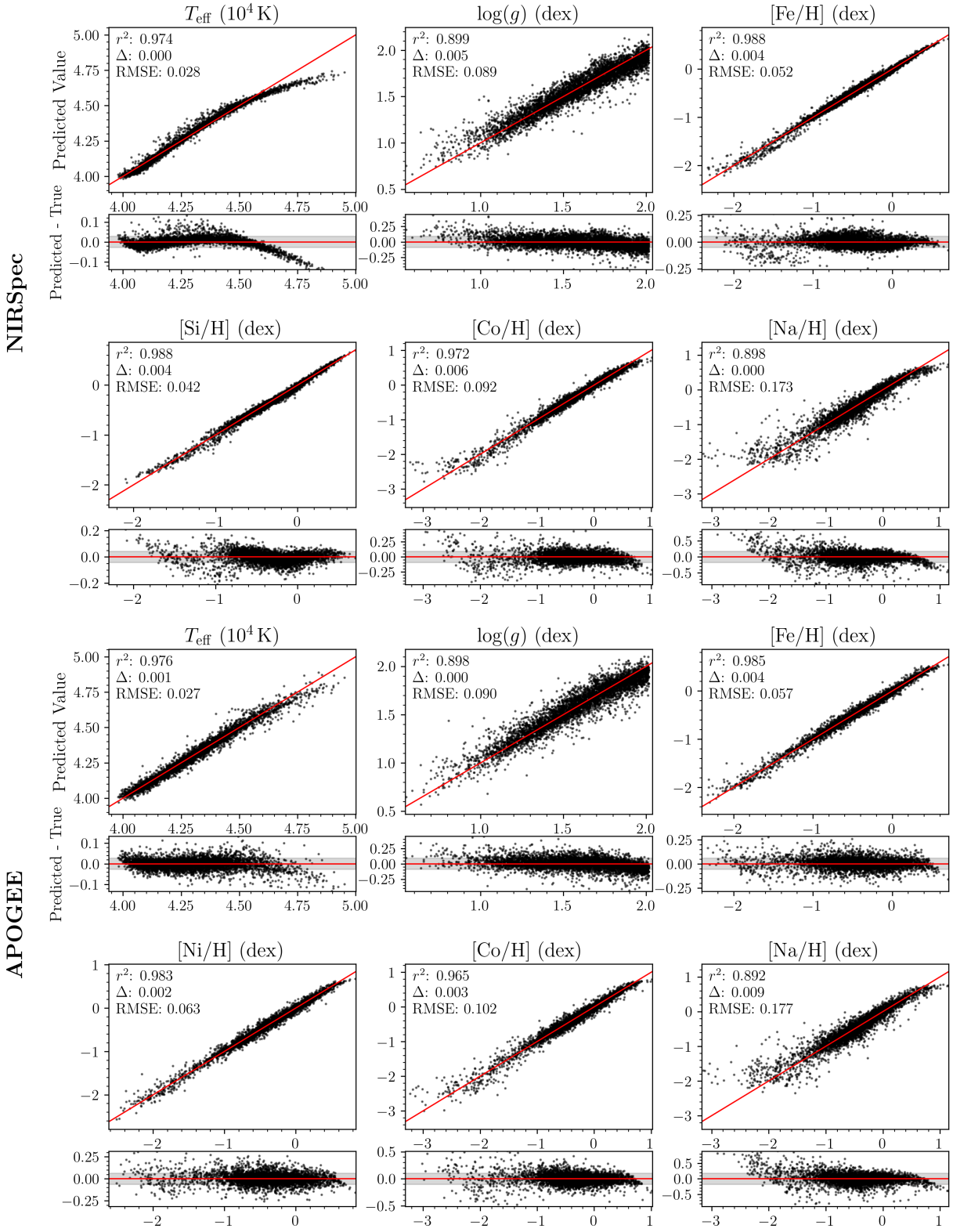


Figure 7: Performance of the single-modal CNN models in predicting fundamental atmospheric parameters along with three chemical abundances representative of high, median, and low RMSE scores. The **top subpanels** report the r^2, Δ, and RMSE of the predictions. The **bottom subpanels** show residuals, with shaded regions indicating one standard deviation. The red line denotes perfect prediction. Adapted from [6].

similarity, with higher scores indicating greater semantic relevance to the query z_q. The top-k most similar embeddings are then returned. To assess the impact of CLIP-alignment on retrieval performance, we compare CrossCSS results on StarCLIP and MockStarCLIP embeddings against baseline searches on the raw spectra: $\mathcal{J}'_{\text{test}} \times [\mathcal{A}'_{\text{test}}]$ and $\mathcal{J}'_{\text{m, test}} \times \mathcal{J}'_{\text{e-m, test}}$, respectively.[11]

Table 3: Overall Performance of CrossCSS on CLIP Embeddings and Baselines (best scores in boldface)

Model	**Query Type**	**Retrieved Type**	*R@1*	*R@5*	*R@10*	*R@50*	MRR
StarCLIP	$f_\Theta(\mathcal{J}'_{\text{test}})$	$g_\Phi(\mathcal{A}'_{\text{test}})$	0.14	0.14	0.24	0.57	0.17
	$g_\Phi(\mathcal{A}'_{\text{test}})$	$f_\Theta(\mathcal{J}'_{\text{test}})$	**0.15**	**0.41**	**0.56**	**0.86**	**0.28**
	$\mathcal{J}'_{\text{test}}$	$\mathcal{A}'_{\text{test}}$	0.00	0.00	0.00	0.00	0.00
	$\mathcal{A}'_{\text{test}}$	$\mathcal{J}'_{\text{test}}$	0.00	0.00	0.00	0.01	0.00
MockStarCLIP	$f_\Theta(\mathcal{J}'_{\text{m, test}})$	$f_{\Theta^*}(\mathcal{J}'_{\text{e-m, test}})$	0.06	**0.22**	**0.35**	**0.75**	**0.19**
	$f_{\Theta^*}(\mathcal{J}'_{\text{e-m, test}})$	$f_\Theta(\mathcal{J}'_{\text{m, test}})$	**0.08**	0.27	0.40	0.75	0.16
	$\mathcal{J}'_{\text{e-m, test}}$	$\mathcal{J}'_{\text{m, test}}$	0.00	0.02	0.03	0.11	0.02
	$\mathcal{J}'_{\text{m, test}}$	$\mathcal{J}'_{\text{e-m, test}}$	0.00	0.02	0.03	0.11	0.02

Table 3 reports the retrieval performance of CrossCSS for both CLIP models against their respective baselines, measured in terms of *SuccessRate@k* and MRR. The columns *R@1*, *R@5*, *R@10* and *R@50* show the results of *SuccessRate@k* when k is $1, 5, 10$ and 50, respectively. The column MRR shows the MRR values of each approach. For each CLIP model, the first two rows present CrossCSS performance on CLIP-aligned embeddings, with the modalities of the query and the retrieved alternated, while the following two rows correspond to baseline searches using raw spectra.

As shown in Table 3, CrossCSS on CLIP-aligned embeddings consistently outperforms all baseline methods. Notably, CrossCSS on raw spectra often *fails* to retrieve meaningful matches, reflecting its inabiity to capture the underlying semantic correspondence between modalities. Averaging across both query directions, CrossCSS on MockStarCLIP test embeddings obtains improvements of 1,000%, 1,067% and 582% in *SuccessRate@k*, when k is 5, 10 and 50, respectively, relative to its baseline. These improvements correspond to at least an order-of-magnitude increase in successful retrievals—i.e., cases where the true cross-modal pair (or one of its augmentations) appears within the top-k results. These results demonstrate that CLIP-alignment not only captures the shared semnatic structure across modalities but also enhances it, enabling CrossCSS to more accurately retrieve results that satisfy the user's query intent.

Interestingly, while both directional settings for MockStarCLIP and its baseline achieve similar search results, we observe an asymmetry in the retrieval results for searches with StarCLIP embeddings. In particular, querying APOGEE embeddings in $g_\Phi(\mathcal{A}'_{\text{test}})$ to retrieve $f_\Theta(\mathcal{J}'_{\text{test}})$ consistently outperforms the reverse direction. This may suggest that g_Φ produces more discriminative and higher-quality embeddings compared to f_Θ, potentially due to inherent differences in spectral characteristics of APOGEE and NIRSpec data.

[11] We define $[\mathcal{A}'_{\text{test}}]$ as the truncated version of $\mathcal{A}'_{\text{test}}$, where each spectrum is cropped by $8,575 - 8,192 = 383$ pixels to remove missing spectral regions (15,100-15,152$\mathring{A}$ and 16,969-17,000$\mathring{A}$). This ensures dimensional alignment with $\mathcal{J}'_{\text{test}}$ for computing cosine similarity in Equation (2.12).

5.2.2 Dimensionality Reduction

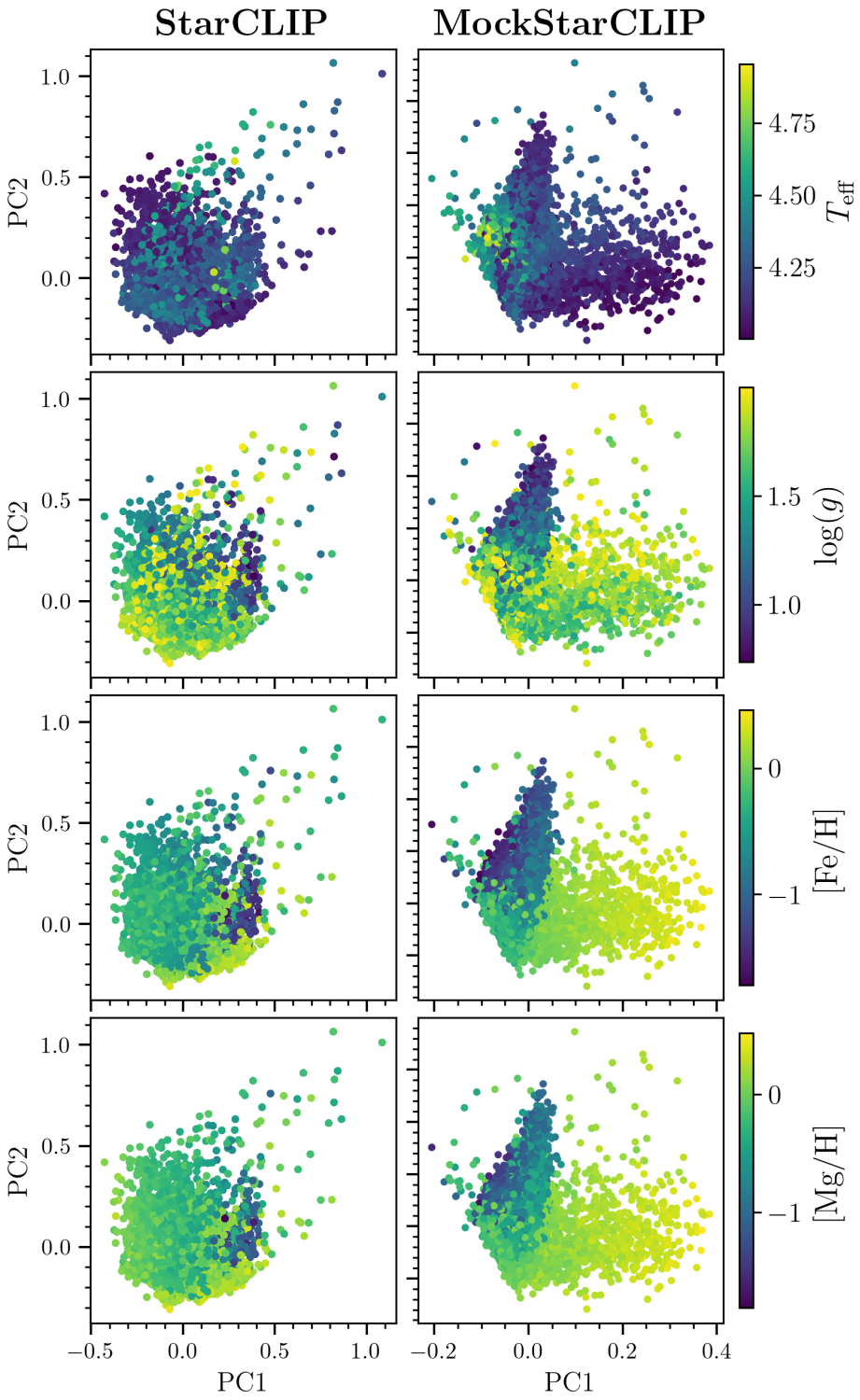


Figure 8: PCA projections of StarCLIP (*left*) and MockStarCLIP (*right*) test embeddings onto their first two principal components (PC). Each point represents a compressed embedding, colour-coded by one of four stellar parameters: $T_{\rm eff}$, $\log g$, [Fe/H], or [Mg/H], as indicated.

Parameterised models with many correlated parameters often amenable to dimensionality reduction (DR). Such methods aim to identify a lower-rank latent space that more directly reflects the underlying physical variables governing the system (see, e.g., [88] for details). Here, we apply Principal Component Analysis (PCA) [89] to StarCLIP (i.e., $f_\Theta(\mathcal{J}'_{\text{test}}) \cup g_\Phi(\mathcal{A}'_{\text{test}})$) and MockStarCLIP (i.e., $f_\Theta(\mathcal{J}'_{\text{m, test}}) \cup f_{\Theta^*}(\mathcal{J}'_{\text{e-m, test}})$) test embeddings to assess whether they encode core stellar properties. Briefly, given a set of test embeddings, PCA decomposes the sample covariance matrix into a set of eigenvectors sorted by their corresponding eigenvalue—that is, by the variance they account for. The eigenvectors that account for most of the variance are assumed to predominately represent real correlations between physical parameters, while those that explain minimal variance are typically attributed to measurement noise and are thus discarded (e.g., [88]). This yields a reduced embedding space, in which the distribution of stellar fundamental parameters can be more easily visualised.

Using the PCA implementation provided by `scikit-learn` [90], we project StarCLIP and MockStarCLIP test embeddings onto the linear subspace spanned by their top *two* principal components, which together explain ∼41% and ∼50% of their total sample variance, respectively. Figure 8 presents the resulting projections, in which each compressed embedding is colour-coded by one of three atmospheric parameters ($T_{\text{eff}}, \log g$, [Fe/H]) or a representative chemical abundance ([Mg/H]). The projections exhibit a noisy but broadly continuous evolution of stellar parameters across the reduced embedding space, suggesting that StarCLIP and MockStarCLIP arrange embeddings around shared physical semantics in the higher-dimensional latent space. Interestingly, MockStarCLIP projections exhibit clearer structure, characterised by two stellar parameter gradients that are closely aligned with its principal component axes. This suggests MockStarCLIP has learned more *disentangled* latent representations in the higher-dimensional space, in which individual axes correspond more directly to underlying astrophysical variables. Altogether, the roughly continuous variation of stellar parameters across the reduce space offers a qualitative indication that regression from CLIP-aligned embeddings to stellar labels should be possible. Equipped with this insight, we proceed in §5.2.3 to quantitatively recover stellar properties from the learned embeddings.

5.2.3 Stellar Property Recovery

We show StarCLIP and MockStarCLIP are capable of inferring stellar properties from learned embeddings in $f_\Theta(\mathcal{J}')$ and $f_{\Theta^*}(\mathcal{J}'_{\text{e-m}})$, respectively. Conventional ML approaches to this task would typically rely on bespoke CNNs, entailing the development of an end-to-end pipeline from scratch [5, 6]. In contrast, CLIP-aligned embeddings already capture expressive features of the input stars, enabling us to instead apply a simple multi-output linear probe to regress the catalog-reported stellar properties directly from the CLIP embeddings. As a supervised baseline, we include a modified StarNet CNN [5], described in §4.1.1, trained end-to-end on the spectral databases $\mathcal{J}'$ and $\mathcal{J}'_{\text{e-m}}$. To account for differences in dataset size, we adjust the following training hyperparameters:

- For the CNN trained on $\mathcal{J}'$, we use a reduced learning rate $\eta' = 5 \times 10^{-4}$.
- For the CNN trained on $\mathcal{J}'_{\text{e-m}}$, we enable weight decay at a rate of $\lambda' = 5 \times 10^{-4}$ and use batch sizes of $N^{\text{train}}_{\text{batch}} = 16, N^{\text{val.}}_{\text{batch}} = 4$ and $N^{\text{train}}_{\text{batch}} = 128$.

To mimic the MockStarCLIP setup, the CNN trained on $\mathcal{J}'_{\text{e-m}}$ is also initialised with weights pretrained on $\mathcal{J}'$ and its convolutional layers are frozen at training time. Importantly, in all analyses involving the augmented datasets $f_\Theta(\mathcal{J}'_{\text{test}})$ and $\mathcal{J}'$, we perform probing at the *instance-group level*: we mean-aggregate the predictions from all augmented views of the same star, ensuring each star contributes exactly one prediction in downstream evaluation. An empirical spread s is computed from the five predictions and added in quadrature, element-wise, to the estimated intrinsic scatter σ_{int} from

SDSS-IV DR17 APOGEE measurements [87]. This gives a total estimated uncertainty:

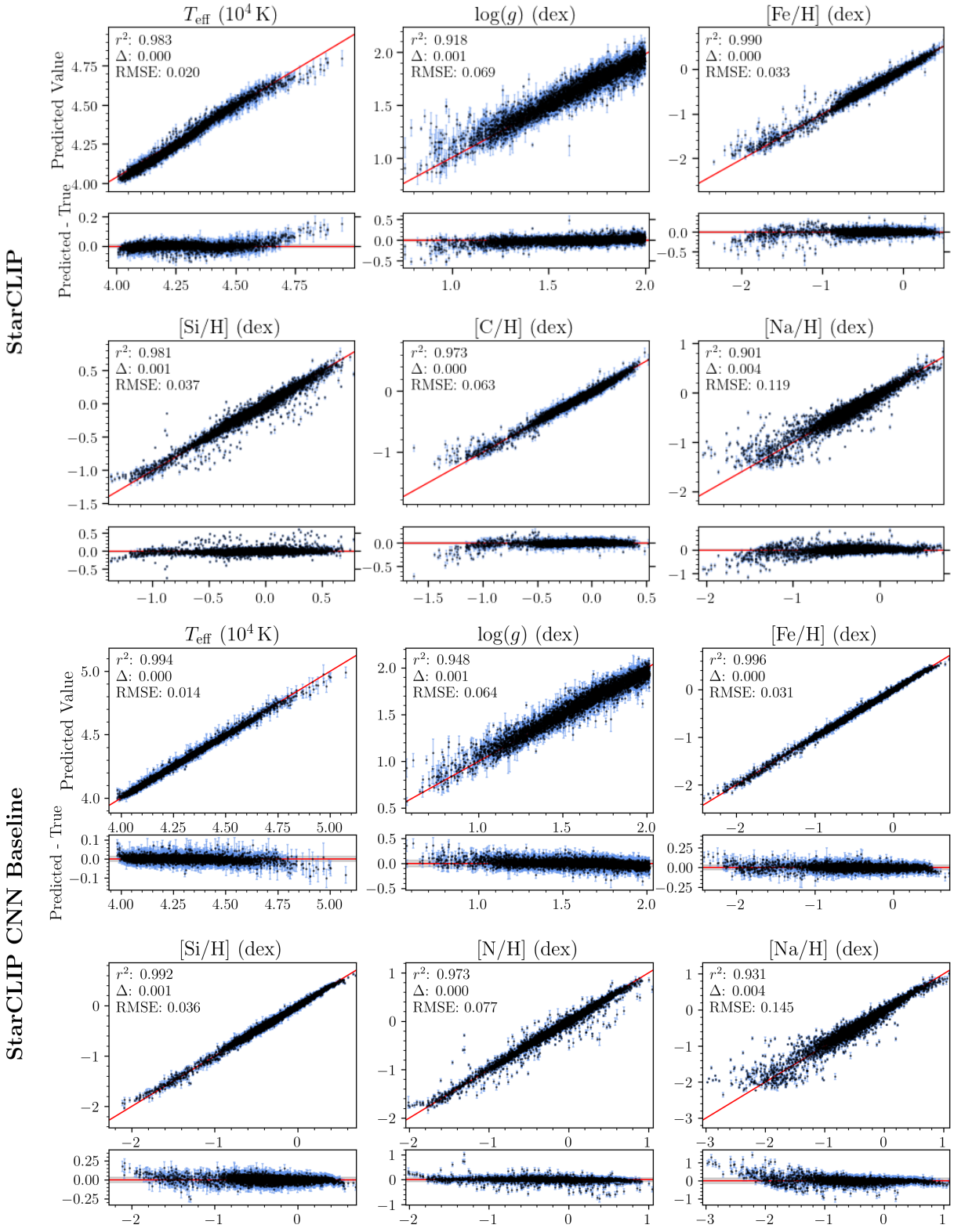
$$\sigma = \sqrt{s^2 + \sigma_{\text{int}}^2}. \tag{5.1}$$

Figure 9 presents predictions derived from (a) StarCLIP and (b) MockStarCLIP, along with their respective CNN baselines, across six representative stellar properties: three atmospheric parameters and three chemical abundances with high, median and low RMSE scores. For completeness, Tables 5 and 6 in the Appendix report the evaluation metrics for all 20 stellar parameters predicted by each model. Again, we find that both CLIP-based models recover all stellar properties with high accuracy, achieving r^2 scores typically exceeding $\sim$0.90, RMSE scores comparable to the estimated precisions of SDSS-IV DR17 APOGEE measurements [87], and minimal biases—generally one to two orders smaller than the corresponding RMSE values. Consistent with the findings of [22], our CLIP-based models successfully demonstrate an ability to encode core physical attributes of the input stars in their embeddings, despite undergoing no task-specific fine-tuning. Interestingly, while StarCLIP outperforms MockStarCLIP across most tasks, as anticipated, it achieves equal or slightly lower r^2 scores for [S/H], [Cr/H], [Co/H] and [Ce/H]. One potential reason for this behaviour is that MockStarCLIP, trained with smaller τ, induces a more separable and discriminative embedding space, which may help to resolve subtle or degenerate spectral features– and thereby improve predictions for parameters with weaker spectral signatures. Finally, although both CLIP-based models perform competitively, they typically underperform relative to their supervised counterparts, particularly in terms of r^2. Consistent with [91], [92] and [93], these findings suggest that specialised, pre-trained foundation models have not yet surpassed traditional supervised approaches trained directly on downstream task data. Nonetheless, the ability of our CLIP-based models to generalise well in low-data regimes, enrich shared semantic information, and transfer to downstream tasks without fine-tuning highlights important advantages of our approach over supervised analysis.

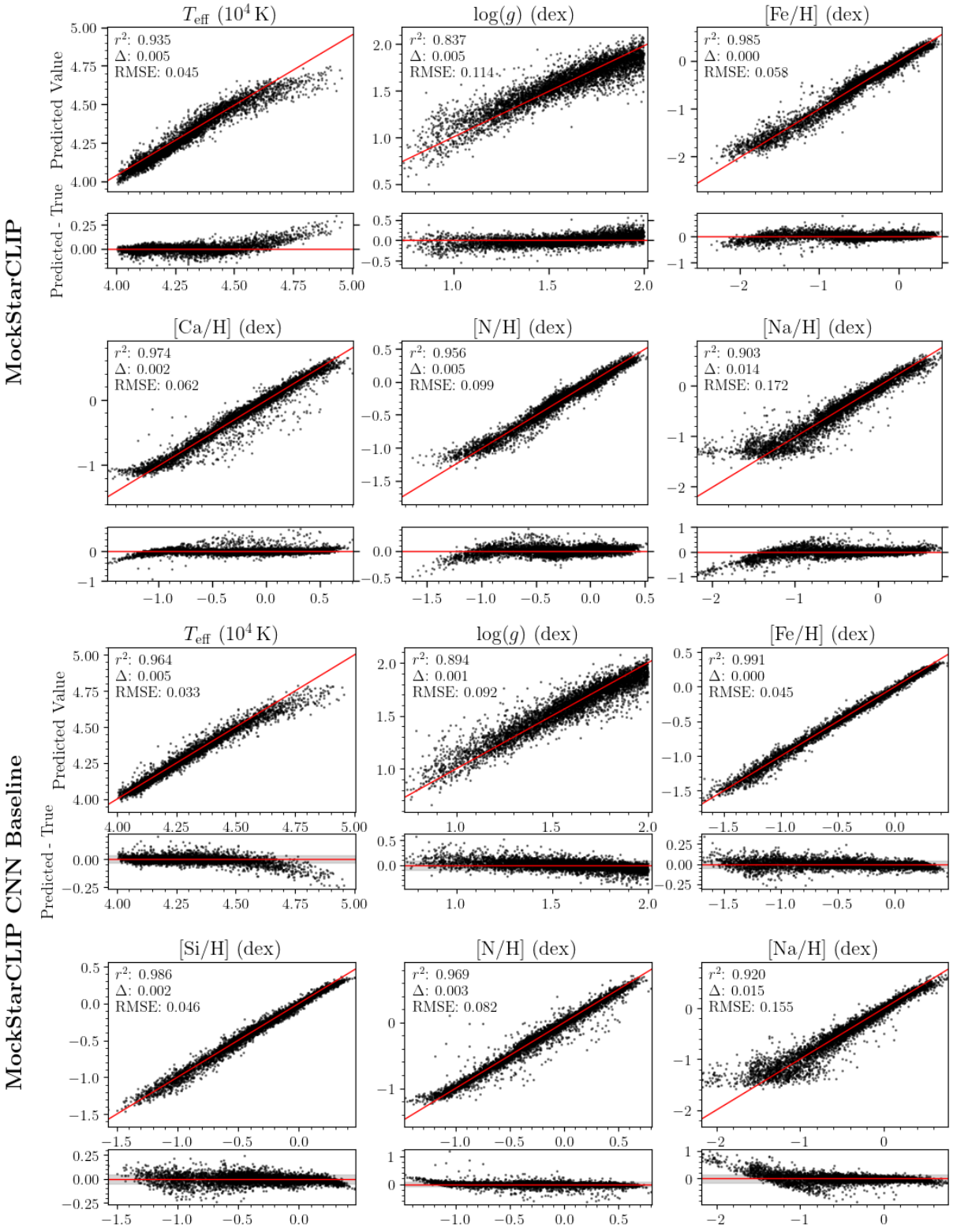
6 Conclusion

We have presented StarCLIP, and its variant MockStarCLIP, versatile cross-modal foundation models for stellar spectroscopy. Both models encode in their embeddings core physical information about the underlying stars, leading to strong performance across a variety of downstream tasks, including cosine similarity search and stellar property estimation. We believe these rich embeddings have the potential to serve as off-the-shelf tools for building higher-level models, facilitating analysis of NIRSpec data. This, in turn, will make accessible a vast wealth of chemical information in the spectra of stars outside the MW.

In the future, we plan to extend MockStarCLIP's capabilities to integrate observed NIRSpec data, validating its real-world applicability. While our pipeline implements semi-empirical, stochastic transformations to simulate physical corruptions in observed data, the fidelity of these simulations is not yet clear. Ultimately, fine-tuning the model to work well with observed datasets would help set the stage for the broader adoption of foundation models in extragalactic stellar spectroscopy, poising the field for transformative growth.



(a) StarCLIP and Supervised CNN Baseline Performance



(b) MockStarCLIP and Supervised CNN Baseline Performance

Figure 9: Performance of (a) StarCLIP and (b) MockStarCLIP, and their respective baselines, in predicting fundamental atmospheric parameters and representative chemical abundances. All error bars are computed using Equation (5.1). In general, both models are in agreement with but underperform relative to their respective supervised counterparts.

7 Acknowledgments

We thank Drs Nathan Sandford, Jo Bovy, Joshua Speagle and Ting Li of the University of Toronto and Dr Miles Cranmer of the University of Cambridge for their insightful discussions. We thank Dr Nathan Sandford for providing the APOGEE and JWST/NIRSpec databases, without which this work would not have been possible. Computations in this paper were run on the `Geir` server between May 1st, 2024 and May 1st, 2025, a resource provided by the Dunlap Institute of Astronomy and Astrophysics at the University of Toronto. The analysis is supported by a fully open-source software stack, including `astropy` [94], `h5py` [95], `matplotlib` [96], `numpy` [97, 98], `pandas` [99, 100], `scipy` [101], `sklearn` [90] and `torch` [74]. Illustrations were created using bioRender.[12]

Disclaimer. All code was written solely by the author between May 1st, 2024 and May 1st, 2025. All writing of this paper was undertaken solely by the author, strictly begun after the publication of the Part III essay titles.

Data Availability. The data and code underlying this article are available in a GitHub repository at this link.

References

[1] Matteucci, F. (2012) *Chemical Evolution of Galaxies*, Springer-Verlag Berlin Heidelberg

[2] Jakobsen, P. *et al.* (2022) *The Near-Infrared Spectrograph (NIRSpec) on the James Webb Space Telesope: I. Overview of the instrument and its capabilities*, *Astronomy & Astrophysics* **661**: A80

[3] Böker, T. *et al.* (2023) *In-orbit Performance of the Near-infrared Spectrograph NIRSpec on the James Webb Space Telescope*, *Publications of the Astronomical Society of the Pacific* **135** (1045): 038001

[4] Gardner, J. P. *et al.* (2006) *The James Webb Space Telescope*, *Space Science Reviews* **123** (4): 485–606

[5] Fabbro, S. *et al.* (2018) *An application of deep learning in the analysis of stellar spectra*, *Monthly Notices of the Royal Astronomical Society* **475** (3): 2978–2993

[6] Blancato, K. *et al.* (2022) *Data-driven derivation of stellar properties from photometric time series data using convolutional neural networks*, *The Astrophysical Journal* **933** (2): 241

[7] Baron, D. (2019) *Machine Learning in Astronomy: a practical overview*, arXiv:1904.07248 [astro-ph.IM].

[8] Noda, K. *et al.* (2015) *Audio-visual speech recognition using deep learning*, *Applied Intelligence* **42**: 722–737

[9] Carreira, J. & Zisserman, A. (2018) *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset*, arXiv:1705.07750 [cs.CV]

[10] Blanton, M. R. *et al.* (2017) *Sloan Digital Sky Survey IV: Mapping the Milky Way, Nearby Galaxies, and the Distant Universe*, *The Astronomical Journal* **154** (1): 28

[11] Majewski, S. R. *et al.* (2017) *The Apache Point Observatory Galactic Evolution Experiment (APOGEE)*, *The Astronomical Journal* **154** (3): 94

[12] https://www.biorender.com/

[12] Buder, S. *et al.* (2021) *The GALAH+ survey: Third data release*, *Monthly Notices of the Royal Astronomical Society* **506** (1): 150–201

[13] Wang, S. *et al.* (2021) *LAMOST Time-Domain survey: first results of four K2 plates*, *Research in Astronomy and Astrophysics* **21** (11): 292

[14] Chevallard, J. *et al.* (2018) *Simulating and interpreting deep observations in the Hubble Ultra Deep Field with the JWST/NIRSpec low-resolution 'prism'*, *Monthly Notices of the Royal Astronomical Society* **483** (2): 2621–2640

[15] Bagnasco, G. *et al.* (2007) *Cryogenic Optical Systems and Instruments XII*, in *Proc. SPIE Conf. Ser.* **6692**: 66920M, eds. Heaney, J. B. & Burriesci, L. G., SPIE, Bellingham

[16] Kannan, R. *et al.* (2022) *The thesan project: predictions for multitracer line intensity mapping in the epoch of reionization*, *Monthly Notices of the Royal Astronomical Society* **514** (3): 3857–3878

[17] Nguyen, T. D. *et al.* (2023) *AstroLLaMA: Towards Specialized Foundation Models in Astronomy*, arXiv:2309.06126 [astro-ph.IM]

[18] Smith, M. J. *et al.* (2024) *AstroPT: Scaling Large Observation Models for Astronomy*, arXiv:2405.14930 [astro-ph.IM]

[19] Mishra-Sharma, S. *et al.* (2024) *PAPERCLIP: Associating Astronomical Observations and Natural Language with Multi-Modal Models*, arXiv:2403.08851 [astro-ph.IM]

[20] Slijepčević, I. V. *et al.* (2024) *Radio galaxy zoo: towards building the first multipurpose foundation model for radio astronomy with self-supervised learning*, *RAS Techniques and Instruments* **3** (1): 19–32.

[21] Leung, H. W. & Bovy, J. (2024) *Towards an astronomical foundation model for stars with a transformer-based model*, *Monthly Notices of the Royal Astronomical Society* **527** (1): 1494–1520.

[22] Parker, L. *et al.* (2024) *AstroCLIP: a cross-modal foundation model for galaxies*, *Monthly Notices of the Royal Astronomical Society* **531** (4): 4990–5011

[23] Bommasani, R. *et al.* (2022) *On the Opportunities and Risks of Foundation Models*, arXiv:2108.07258 [cs.LG]

[24] Lee, J. *et al.* (2024) *Time-Series Representation Feature Refinement with a Learnable Masking Augmentation Framework in Contrastive Learning*, *Sensors* **24** (24): 7932

[25] Abdurro'uf *et al.* (2022) *The Seventeenth Data Release of the Sloan Digital Sky Surveys: Complete Release of MaNGA, MaStar, and APOGEE-2 Data*, *The Astrophysical Journal Supplement Series* **259** (2): 35.

[26] Kurucz, R. L. (1970) *SAOSR, 309*, *Smithsonian Astrophysical Observatory Special Report* **309**

[27] Kurucz, R. L.(1993) *SYNTHE Spectrum Synthesis Programs and Line Data*, Smithsonian Astrophysical Observatory, Cambridge, MA

[28] Kurucz, R. L. (2005) *MSAIS, 8, 14*, *Memorie della Società Astronomica Italiana Supplementi* **8**: 14

[29] Kurucz, R. L. (2013) *ATLAS12: Opacity sampling model atmosphere program*, Astrophysics Source Code Library, ascl:1303.024

[30] Kurucz, R. L. (2017) *ATLAS9: Model atmosphere program with opacity distribution functions*, Astrophysics Source Code Library, ascl:1710.017

[31] Kurucz, R. L. & Avrett, E. H. (1981) *SAOSR, 391, Smithsonian Astrophysical Observatory Special Report* **391**

[32] Kurucz, R. L. (1979) *Model atmospheres for G, F, A, B, and O stars*, *The Astrophysical Journal Supplement Series* **40** (1): 1–340

[33] Kurucz, R. L. *et al.* (2009) *Including All the Lines*, in *AIP Conference Proceedings*, AIP, pp. 43–51

[34] Koushik, J. (2016) *Understanding Convolutional Neural Networks*, arXiv:1605.09081 [stat.OT]

[35] Cacciari, I. & Ranfagni, A. (2024) *Hands-On Fundamentals of 1D Convolutional Neural Networks—A Tutorial for Beginner Users*, *Applied Sciences* **14** (18): 8500

[36] Kingma, D. P. & Ba, J. (2017) *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs.LG]

[37] Keshari, R. *et al.* (2018) *Learning Structure and Strength of CNN Filters for Small Sample Size Training*, arXiv:1803.11405 [cs.CV]

[38] Bharadwaj, S. *et al.* (2010) *Face recognition for newborn: A preliminary study*, in *IEEE 4th International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pp. 1–6. doi:10.1109/BTAS.2010.5634500

[39] Bharadwaj, S. *et al.* (2016) *Domain Specific Learning for Newborn Face Recognition*, *IEEE Transactions on Information Forensics and Security* **11**: 1–1

[40] Kiranyaz, S. *et al.* (2021) *1D convolutional neural networks and applications: A survey*, *Mechanical Systems and Signal Processing* **151**: 107398

[41] Gui, J. *et al.* (2024) *A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends*, arXiv:2301.05712 [cs.LG]

[42] Liu, X. *et al.* (2023) *Self-Supervised Learning: Generative or Contrastive*, *IEEE Transactions on Knowledge and Data Engineering* **35** (1): 857–876

[43] Hadsell, R., Chopra, S. & LeCun, Y. (2006) *Dimensionality Reduction by Learning an Invariant Mapping*, in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, **2**: 1735–1742

[44] Logeswaran, L. & Lee, H. (2018) *An efficient framework for learning sentence representations*, arXiv:1803.02893 [cs.CL]

[45] Hénaff, O. J. *et al.* (2020) *Data-Efficient Image Recognition with Contrastive Predictive Coding*, arXiv:1905.09272 [cs.CV]

[46] Chuang, C.-Y. *et al.* (2020) *Debiased Contrastive Learning*, arXiv:2007.00224 [cs.LG]

[47] von Kügelgen, J. *et al.* (2022) *Self-Supervised Learning with Data Augmentations Provably Isolates Content from Style*, arXiv:2106.04619 [stat.ML]

[48] Liu, Z. *et al.* (2024) *Contrastive Learning Framework for Bitcoin Crash Prediction*, *Stats* **7** (2): 402–433

[49] Radford, A. *et al.* (2021) *Learning Transferable Visual Models From Natural Language Supervision*, arXiv:2103.00020 [cs.CV]

[50] van den Oord, A. *et al.* (2019) *Representation Learning with Contrastive Predictive Coding*, arXiv:1807.03748 [cs.LG]

[51] Poole, B. *et al.* (2019) *On Variational Bounds of Mutual Information*, arXiv:1905.06922 [cs.LG]

[52] Tian, Y. *et al.* (2020) *Contrastive Multiview Coding*, arXiv:1906.05849 [cs.CV]

[53] Song, J. & Ermon, S. (2020) *Understanding the Limitations of Variational Mutual Information Estimators*, arXiv:1910.06222 [cs.LG]

[54] Manna, S. *et al.* (2024) *Dynamically Scaled Temperature in Self-Supervised Contrastive Learning*, arXiv:2308.01140 [cs.LG]

[55] Wang, J. *et al.* (2024) *What Has Been Overlooked in Contrastive Source-Free Domain Adaptation: Leveraging Source-Informed Latent Augmentation within Neighborhood Context*, arXiv:2412.14301 [cs.CV]

[56] Wang, T. & Isola, P. (2020) *Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere*, in *Proceedings of the 37th International Conference on Machine Learning*, **119**: 9929–9939

[57] Robinson, J. *et al.* (2021) *Contrastive Learning with Hard Negative Samples*, arXiv:2010.04592 [cs.LG]

[58] Kukleva, A. *et al.* (2023) *Temperature Schedules for Self-Supervised Contrastive Methods on Long-Tail Data*, arXiv:2303.13664 [cs.CV]

[59] Wilson, J. C. *et al.* (2019) *The Apache Point Observatory Galactic Evolution Experiment (APOGEE) Spectrographs*, *Publications of the Astronomical Society of the Pacific* **131** (999): 055001

[60] Gunn, J. E. *et al.* (2006) *The 2.5 m Telescope of the Sloan Digital Sky Survey*, *The Astronomical Journal* **131** (4): 2332–2359

[61] Bowen, I. S. & Vaughan, A. H. (1973) *The Optical Design of the 40-in. Telescope and of the Irenee DuPont Telescope at Las Campanas Observatory, Chile*, *Applied Optics* **12**: 1430–1435

[62] Wilson, J. C.. *et al.* (2010) *The Apache Point Observatory Galactic Evolution Experiment (APOGEE) high-resolution near-infrared multi-object fiber spectrograph*, in *Ground-based and Airborne Instrumentation for Astronomy III*, *SPIE Conference Series* **7735**: 77351C

[63] Zasowski, G. *et al.* (2017) *Target Selection for the SDSS-IV APOGEE-2 Survey*, *The Astronomical Journal* **154** (5): 198

[64] Nidever, D. L. *et al.* (2015) *The Data Reduction Pipeline for the Apache Point Observatory Galactic Evolution Experiment*, *The Astronomical Journal* **150** (6): 173

[65] García Pérez, A. E. *et al.* (2016) *ASPCAP: The APOGEE Stellar Parameter and Chemical Abundances Pipeline*, *The Astronomical Journal* **151** (6): 144

[66] Holtzman, J. A. *et al.* (2015) *Abundances, Stellar Parameters, and Spectra from the SDSS-III/APOGEE Survey*, *The Astronomical Journal* **150** (5): 148

[67] Allende Prieto, C. *et al.* (2006) *A Spectroscopic Study of the Ancient Milky Way: F- and G-Type Stars in the Third Data Release of the Sloan Digital Sky Survey*, *The Astrophysical Journal* **636** (2): 804–820

[68] Ting, Y.-S. *et al.* (2017) *Prospects for Measuring Abundances of ¿20 Elements with Low-resolution Stellar Spectra*, *The Astrophysical Journal* **843** (1): 32

[69] Sandford, N. R. *et al.* (2020) *Forecasting Chemical Abundance Precision for Extragalactic Stellar Archaeology*, *The Astrophysical Journal Supplement Series* **249** (2): 24

[70] Asplund, M. *et al.* (2009) *The Chemical Composition of the Sun*, *Annual Review of Astronomy and Astrophysics* **47** (1): 481–522

[71] Ferruit, P. *et al.* (2022) *The Near-Infrared Spectrograph (NIRSpec) on the James Webb Space Telescope: II. Multi-object spectroscopy (MOS)*, *Astronomy & Astrophysics* **661**: A81

[72] Sharpe, K. *et al.* (2024) *The First Alpha Abundances in Isolated Dwarf Galaxies from JWST Spectroscopy*, Slide deck presented at the University of Chicago, July 2024

[73] Sun, Q. *et al.* (2023) *EVA-CLIP: Improved Training Techniques for CLIP at Scale*, arXiv:2303.15389 [cs.CV]

[74] Paszke, A. *et al.* (2019) *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[75] Frémat, Y., Houziaux, L. & Andrillat, Y. (1996) *Higher Paschen lines in the spectra of early-type stars*, *Monthly Notices of the Royal Astronomical Society* **279** (1): 25–31

[76] Greve, A. (2010) *Extinction A_V, R towards emission nebulae derived from common upper level Paschen-Balmer hydrogen lines*, *Astronomy & Astrophysics* **518**: A62

[77] Reddy, N. A. *et al.* (2023) *Paschen-line Constraints on Dust Attenuation and Star Formation at $z \sim 1$–3 with JWST/NIRSpec*, *The Astrophysical Journal* **948** (2): 83

[78] Xiang, M. *et al.* (2022) *Stellar labels for hot stars from low-resolution spectra: I. The HotPayne method and results for 330,000 stars from LAMOST DR6*, *Astronomy & Astrophysics* **662**: A66

[79] Nicholls, C. P. *et al.* (2017) *CRIRES-POP: a library of high resolution spectra in the near-infrared. II. Data reduction and the spectrum of the K giant 10 Leonis*, *Astronomy & Astrophysics* **598**: A79

[80] Bradski, G. (2000) *The OpenCV Library*, *Dr. Dobb's Journal of Software Tools*

[81] Freedman, D. & Diaconis, P. (1981) *On the histogram as a density estimator: L_2 theory*, *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* **57**: 453–476

[82] Aczel, T. & Wattenhofer, R. (2023) *Efficient Multimodal Alignment: To Freeze or Not to Freeze?*, in *Proceedings of UniReps: the First Workshop on Unifying Representations in Neural Models*. Available at: https://openreview.net/forum?id=ul5OdS6pfU

[83] Ye, X., Bunescu, R. & Liu, C. (2014) *Learning to Rank Relevant Files for Bug Reports Using Domain Knowledge*, in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, pp. 689–699. doi:10.1145/2635868.2635874

[84] Lv, F. *et al.* (2015) *CodeHow: Effective Code Search Based on API Understanding and Extended Boolean Model*, in *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 260–270. doi:10.1109/ASE.2015.42

[85] Li, X. *et al.* (2016) *Relationship-Aware Code Search for JavaScript Frameworks*, in *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, pp. 690–701. doi:10.1145/2950290.2950341

[86] Raghothaman, M., Wei, Y., & Hamadi, Y. (2016) *SWIM: Synthesizing What I Mean: Code Search and Idiomatic Snippet Synthesis*, in *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*, ACM. doi:10.1145/2884781.2884808

[87] Hegedűs, V. *et al.* (2023) *Comparative analysis of atmospheric parameters from high-resolution spectroscopic sky surveys: APOGEE, GALAH, Gaia-ESO*, *Astronomy & Astrophysics* **670**: A107. doi:10.1051/0004-6361/202244813

[88] Nielsen, M. B. *et al.* (2023) *Simplifying asteroseismic analysis of solar-like oscillators: An application of principal component analysis for dimensionality reduction*, *Astronomy & Astrophysics* **676**: A117, doi: 10.1051/0004-6361/202346086

[89] Hotelling, H. (1933) *Analysis of a complex of statistical variables into principal components*, *Journal of Educational Psychology* **24**: 417–441, 498–520, doi: 10.1037/h0071325

[90] Pedregosa, F. *et al.* (2011) *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12**: 2825–2830

[91] Jakubik, J. *et al.* (2023) *Foundation models for generalist geospatial artificial intelligence*, arXiv:2310.18660 [cs.CV]

[92] Yang, K. K., Fusi, N., & Lu, A. X. (2024) *Convolutions are competitive with transformers for protein sequence pretraining*, *Cell Systems* **15** (3): 286–294.e2

[93] Xu, Z. *et al.* (2025) *Specialized foundation models struggle to beat supervised baselines*, arXiv:2411.02796 [cs.LG]

[94] Astropy Collaboration (2022) *Astropy Collaboration, 2022*, *The Astrophysical Journal* **935**: 167

[95] Collette, A. (2013) *Python and HDF5*, O'Reilly

[96] Hunter, J. D. (2007) *Matplotlib: A 2D Graphics Environment*, *Computing in Science & Engineering* **9** (3): 90–95, doi: 10.1109/MCSE.2007.55

[97] van der Walt, S., Colbert, S. C. & Varoquaux, G. (2011) *The NumPy Array: A Structure for Efficient Numerical Computation*, *Computing in Science & Engineering* **13**(2): 22–30, doi: 10.1109/MCSE.2011.37

[98] Harris, C. R. *et al.* (2020) *Array programming with NumPy*, *Nature* **585** (7825): 357–362, doi: 10.1038/s41586-020-2649-2

[99] McKinney, W. (2010) *Data structures for statistical computing in Python*, in *Proceedings of the 9th Python in Science Conference*, van der Walt, S. and Millman, J. (eds.), pp. 56–61

[100] Reback, J. *et al.* (2022) *pandas-dev/pandas: Pandas 1.4.3*, Zenodo, DOI: https://doi.org/10.5281/zenodo.6702671

[101] Virtanen, P. *et al.* (2020) *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, *Nature Methods* **17**: 261–272, doi: 10.1038/s41592-019-0686-2

A Appendix

Table 4: Single-modal CNN Performance on Stellar Property Recovery

Parameter	**NIRSpec**			**APOGEE**		
	r^2	Δ	RMSE	r^2	Δ	RMSE
$T_{\rm eff}$	0.974	0.000	0.028	0.976	0.001	0.027
$\log(g)$	0.899	0.005	0.089	0.898	0.000	0.090
[C/H]	0.976	0.004	0.090	0.970	0.003	0.102
[N/H]	0.947	0.000	0.107	0.944	0.008	0.109
[O/H]	0.982	0.001	0.051	0.969	0.003	0.068
[Na/H]	0.898	0.000	0.173	0.892	0.009	0.177
[Mg/H]	0.982	0.002	0.053	0.965	0.003	0.075
[Al/H]	0.952	0.002	0.105	0.939	0.008	0.119
[Si/H]	0.988	0.004	0.042	0.971	0.004	0.065
[S/H]	0.911	0.001	0.101	0.906	0.004	0.103
[K/H]	0.902	0.002	0.132	0.888	0.004	0.141
[Ca/H]	0.981	0.005	0.052	0.969	0.003	0.067
[Ti/H]	0.981	0.003	0.070	0.969	0.004	0.092
[V/H]	0.898	0.001	0.151	0.899	0.007	0.151
[Cr/H]	0.958	0.007	0.107	0.954	0.000	0.112
[Mn/H]	0.980	0.003	0.085	0.980	0.004	0.085
[Fe/H]	0.988	0.004	0.051	0.984	0.004	0.058
[Co/H]	0.972	0.006	0.092	0.965	0.003	0.102
[Ni/H]	0.989	0.003	0.049	0.983	0.002	0.063
[Ce/H]	0.877	0.005	0.170	0.906	0.003	0.149

Table 5: StarCLIP and Supervised CNN Baseline Performance on Stellar Property Recovery

Parameter	StarCLIP			**Supervised CNN**		
	r^2	Δ	RMSE	r^2	Δ	RMSE
$T_{\rm eff}$	0.983	0.000	0.020	0.994	0.000	0.014
$\log(g)$	0.918	0.001	0.069	0.948	0.001	0.064
[Fe/H]	0.990	0.000	0.033	0.996	0.000	0.031
[C/H]	0.973	0.000	0.063	0.985	0.000	0.074
[N/H]	0.955	0.000	0.070	0.973	0.000	0.077
[O/H]	0.975	0.001	0.042	0.990	0.001	0.040
[Na/H]	0.901	0.004	0.119	0.931	0.004	0.145
[Mg/H]	0.976	0.000	0.042	0.990	0.000	0.042

Continued on next page

Parameter	StarCLIP			**Supervised CNN**		
	r^2	Δ	RMSE	r^2	Δ	RMSE
[Al/H]	0.960	0.001	0.063	0.974	0.001	0.078
[Si/H]	0.981	0.001	0.037	0.992	0.001	0.036
[S/H]	0.914	0.001	0.070	0.931	0.002	0.092
[K/H]	0.904	0.001	0.091	0.921	0.002	0.121
[Ca/H]	0.981	0.000	0.037	0.989	0.000	0.041
[Ti/H]	0.983	0.000	0.047	0.990	0.000	0.053
[V/H]	0.916	0.001	0.094	0.923	0.002	0.130
[Cr/H]	0.961	0.002	0.072	0.970	0.003	0.093
[Mn/H]	0.987	0.001	0.048	0.993	0.001	0.051
[Co/H]	0.962	0.002	0.072	0.973	0.003	0.093
[Ni/H]	0.985	0.001	0.039	0.994	0.000	0.037
[Ce/H]	0.910	0.002	0.109	0.921	0.003	0.141

Table 6: MockStarCLIP and Supervised CNN Baseline Performance on Stellar Property Recovery

Parameter	MockStarCLIP			**Supervised CNN**		
	r^2	Δ	RMSE	r^2	Δ	RMSE
T_{eff}	0.935	0.005	0.045	0.964	0.005	0.033
$\log(g)$	0.837	0.005	0.114	0.894	0.001	0.092
[Fe/H]	0.985	0.000	0.058	0.991	0.000	0.045
[C/H]	0.964	0.006	0.114	0.981	0.007	0.082
[N/H]	0.956	0.005	0.099	0.969	0.003	0.082
[O/H]	0.962	0.004	0.077	0.981	0.003	0.054
[Na/H]	0.903	0.014	0.172	0.920	0.015	0.155
[Mg/H]	0.965	0.004	0.076	0.985	0.004	0.051
[Al/H]	0.940	0.015	0.121	0.967	0.012	0.090
[Si/H]	0.970	0.003	0.068	0.986	0.002	0.046
[S/H]	0.925	0.006	0.094	0.943	0.006	0.082
[K/H]	0.924	0.017	0.118	0.948	0.007	0.098
[Ca/H]	0.974	0.002	0.062	0.985	0.003	0.048
[Ti/H]	0.975	0.003	0.084	0.985	0.000	0.066
[V/H]	0.892	0.011	0.157	0.918	0.016	0.136
[Cr/H]	0.952	0.007	0.118	0.961	0.013	0.107
[Mn/H]	0.984	0.005	0.079	0.989	0.003	0.064
[Co/H]	0.960	0.011	0.112	0.970	0.003	0.098
[Ni/H]	0.981	0.003	0.066	0.991	0.002	0.047
[Ce/H]	0.928	0.002	0.135	0.939	0.006	0.124