UNIVERSITY OF CAMBRIDGE

Part III Essay 2024-25
Faculty of Mathematics, DAMTP

Applying Contrastive Learning to Stellar Spectra

May 8, 2025

Abstract

This paper presents StarCLIP, and its variant MockStarCLIP: a simple framework for contrastive learning of stellar spectra. We show that

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

Contents

1 Introduction

Absorption features exhibited on a star's spectrum encode its physical and chemical properties, which in turn provides a 'fossil record' of the formation history of the host galaxy in which that star is located (Sandford et al., 2020). The James Webb Space Telescope (JWST), deployed in 2021, exhibits unique spectroscopic capabilities, enabling efficient, high-quality resolved star spectroscopy in neighbouring galaxies which have thus far been too distant, faint or crowded for previous instruments to detect (Sandford et al., 2020). These capabilities have motivated the launch of several recent astronomical programmes, including a number of completed surveys observing $\sim$ 100 red giant branch (RGB) stars in M31 (Nidever et al., 2023) and $\sim$ 300 RGB stars in three globular clusters (Scalco et al., 2024), an ongoing survey – at the time of this paper – observing $\sim$ 300 RGB stars across three isolated dwarf galaxies (Space Telescope Science Institute (STScI), 2023) and a planned programme to observe $\sim$ 200 more M31 disc stars (STScI, 2024).

However, due to its novelty and uniqueness, the domain of JWST resolved-star spectroscopy remains largely unexplored, and many unanswered questions remain regarding optimal strategies for collecting, reducing and analysing its valuable data. In recent years, a growing subset of astronomical surveys have employed data-driven methodologies from machine learning (ML) to extract insights from optical data (Parker et al., 2024). To date, 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. Once obtained, the mapping can be used to perform predictive *classification* or *regression* tasks on unseen data. These methods have achieved tremendous success in data-rich settings, proving pivotal in image recognition tasks and natural language processing (NLP) (Fabbro et al., 2018). However, requiring extensive labelled training data inhibits their application to data-poorer domains, where labels are scarce, imbalanced or expensive to obtain. Additionally, supervised networks are generally *task-specific*, with each new task typically requiring a bespoke model trained or designed from scratch (Parker et al., 2024). This may lead to high computational costs, limited generalisability, and higher risk of overfitting.
- **Unsupervised** algorithms take in as inputs only the set of measured features, without accompanying labels. Their output is a non-linear, non-invertible transformation of the input features, consisting of (i) *clusters*, which are groups of objects sharing similar features, (ii) lists of detected *anomalies*, which are unusual objects in the sample, or (iii) low-dimensional representations of the objects (Baron, 2019). These algorithms can extract new knowledge from datasets that we did not know existed and, therefore, could not have directly searched for (Reis et al., 2021). To this extent, they may be arguably more useful for 'AI for science' research, where human-annotated labels are not always available in quality or quantity. However, in practice, unsupervised networks typically underperform compared to their supervised counterparts, especially in computer vision (CV), so they are seldom implemented in real-world applications (Zhuang et al., 2019).

Large-scale surveys such as Apache Point Observatory Galactic Evolution Experiment (APOGEE), Galactic Archaeology with HERMES (GALAH) and Large Sky Area Multi-object Fiber Spectroscopic Telescope (LAMOST) provide large ($n \sim 10^5$ to 10^7), homogeneous databases of labelled spectra that are virtually ideal for supervised ML applications (Fabbro et al., 2018). By contrast, the JWST survey provides a considerably smaller ($n \sim 100$) database, which limits its eligibility for supervised analysis. What is more, large differences in field-of-view and sensitivity with ground-based surveys make it difficult to acquire overlapping stars with previous datasets (Gardner et al., 2006). This poses challenges in exploring ways to use informative results from large surveys to facilitate analysis of JWST data, motivating the use of synthetic spectra to force overlap. What results is a database of complementary or *cross-modal* observations of the same optical spectra.

One promising ML approach to analyse cross-modal realisations is **contrastive learning**, a form of **self-supervised learning** (SSL). In this approach, a model simultaneously embeds cross-modal data into a shared latent space, creating pairs of high-quality embeddings, *i.e.*, low-dimensional representations of objects that preserve much of their physical information (Parker et al., 2024). With minimal processing, these embeddings can serve as a robust 'foundation' for downstream tasks, particularly, zero- and one-shot learning[1]. Hence, these models are often referred to as **foundation**

[1]In zero- and one-shot learning, a model leverages its learned representations to identify a new class of interest,

models (Bommasani et al., 2022). Notably, in computationally constrained settings where training bespoke supervised models from scratch is impractical, foundation models consistently demonstrate superior performance on zero- and one-shot tasks compared to their fully supervised counterparts (Bommasani et al., 2022).

In this work, we present StarCLIP, and its variant, MockStarCLIP, cross-modal foundation models designed for stellar spectroscopy. Our approach consists of two main steps. First, we train one-dimensional **convolutional neural networks** (1D-CNNs) on single-modal, labelled spectral datasets. Next, we deploy the CNNs as pre-trained encoders to embed spectra in a unified latent space. We apply contrastive learning to align embeddings around shared physical semantics: embeddings corresponding to the same underlying star are brought closer together, while those corresponding to different stars are pushed farther apart. The resulting embeddings can then be seamlessly used for a variety of downstream tasks, including semantic searches and zero- and one-shot predictive tasks, which ultimately facilitates analysis of JWST data.

Spectroscopic data consist of 19,000 pairs of APOGEE-2[2] and synthetic JWST spectra generated from `ATLAS12` and `synthe` codes maintained by R. Kurucz (Kurucz, 1970, 1993, 2005, 2013, 2017; Kurucz and Avrett, 1981). Stellar labels associated to each pair consist of atmospheric parameters, namely, effective temperature ($T_{\text{eff.}}$) and surface gravity ($\log g$)), and 18 elemental abundances ($[X/H]$). To emulate the real JWST data, we apply two distinct sets of stochastic transformations, resulting in 'mock' and 'extended-mock' versions. A small sample ($n = 59$) of representative spectral traces from JWST Cycle 3[3] is used in this data transformation pipeline. StarCLIP and MockStarCLIP result from applying our methodology to APOGEE–mock JWST pairs and downsampled mock–extended-mock JWST pairs, respectively.

A schema of the pipeline is depicted in Figure 1.

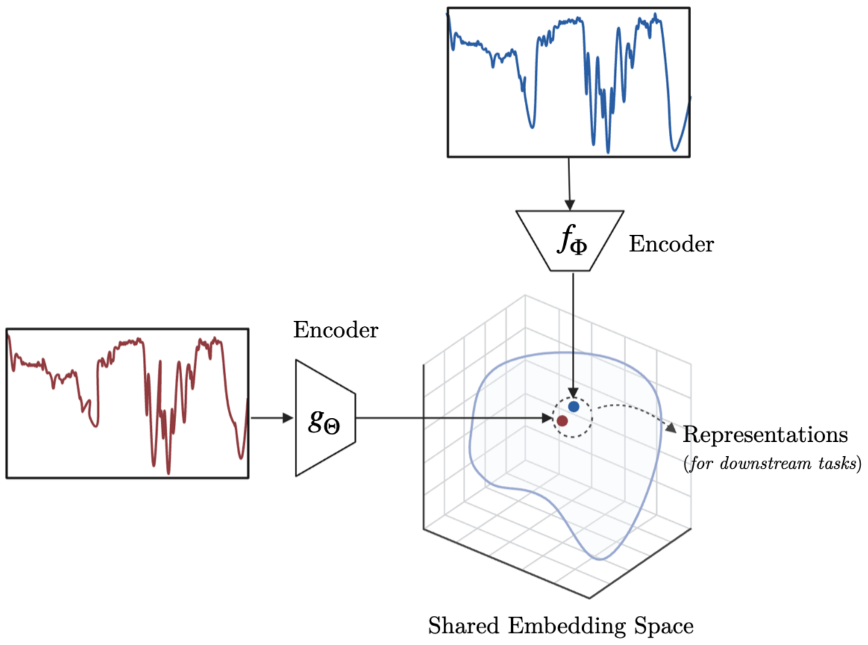

Figure 1: Illustration of the architecture of the StarCLIP and MockStarCLIP models. Two encoders, f_Φ and g_Θ, separately embed pairs of cross-modal spectra into a shared embedding space, where they are aligned under contrastive loss. Learned embeddings then contain high-level physical information that allow them to be seamlessly transferred for downstream tasks. Modalities are distinguished by colour; here, red and blue.

having previously seen zero or one examplar of that class during training, respectively.

[2]https://www.sdss4.org/dr17/

[3]https://www.stsci.edu/jwst/science-execution/approved-programs/general-observers/cycle-3-go

We summarise the contributions of this paper, as follows:

- We develop one of the first self-supervised foundation ML models for analysing JWST data, with a pipeline put into place to ensure seamless integration with *real* JWST data.
- We apply supervised learning to train robust CNNs 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.
- With minimal processing, we enable zero- and one-shot transfer of the models 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.
- We apply non-linear manifold learning on the latent space, visualising its intrinsic local geometry in a lower-dimensional setting.

1.1 Related Works

This paper builds on two recent works. Fabbro et al. (2018) present a deep CNN architecture, called StarNet, capable of determining stellar parameters – $T_{\text{eff.}}, \log g$, and $[Fe/H]$ – with high accuracy and precision from real APOGEE and synthetic APOGEE ASSϵT spectra. Elsewhere, Parker et al. (2024) present a cross-modal foundation model, called AstroCLIP, which aligns representations of galaxy images and spectra in a shared, unified latent space. Pre-trained transformer-based encoders are used to separately embed the modalities into latent space, where they are aligned under contrastive loss. The embeddings are then applied to a variety of downstream tasks, including semantic similarity searches, photometric redshift estimation, galaxy property estimation, and morphology classification, in which they demonstrate remarkable performance, even relative to supervised baselines. Both articles provide accompanying code in publicly accessible GitHub repositories at the following links: StarNet (Fabbro et al., 2018) and AstroCLIP (Parker et al., 2024), respectively.

We build our own foundation models – StarCLIP and its variant MockStarCLIP – in a two-step process involving carefully modified versions of StarNet and AstroCLIP.

1.2 Outline

Code for our models is publicly available on a GitHub repository here. Our paper is organised, as follows. In §2, we provide theoretical background on supervised and self-supervised methods relevant to this paper, addressing their merits and disadvantages. In §3, we provide the astronomical datasets used for analysis. In §4 and §**??**, 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 discuss 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 (SL), the learning algorithm is given a set of training examples $\{(x_i, y_i)\}_{i=1}^{T}$, drawn from the joint distribution of X and Y. Its objective is to learn a mapping $\hat{f} : \mathcal{X} \to \mathcal{Y}$, which minimises the expected loss, as measured by a specific *loss function* $L : \mathcal{Y}^2 \to \mathbb{R}$. That is,

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

over some space of hypotheses functions $\mathcal{F}$ (Koushik, 2016). In regression tasks, choosing the mean squared error (MSE) loss gives: $\hat{f} = \operatorname{argmin}_{f \in \mathcal{F}} \mathbb{E}\|Y - f(X)\|^2$.

2.1.1 1D-CNNs: Feedforward Stage

1D-CNNs, introduced by Boser et al. (1990), are a class of deep learning algorithms that solve equation (2.1) by adaptively learning spatial hierarchies of features in the input data (Yamashita et al., 2018). As shown in Figure .. , 1D-CNNs work by passing X through a series of computational *nodes*, arranged in five types of layers: (i) input, (ii) convolutional, (iii) pooling, (iv) fully connected and (v) output. To illustrate the sequence of operations performed, we consider a single training example $x \in X$ fed to the network. In what follows, denote by $\zeta^{(l)}$ the output vector of the lth hidden layer and $(\mathbf{W}, \mathbf{b})$ the collection of learnable weights/filters and biases associated to the network.

- **Input layer**. The input layer is simply the identity map: $\zeta_{\text{in}} := x$.
- **Convolutional layers**. Following Cacciari and Ranfagni (2024), consider a collection of p filters $\{W_j^{(1)}\}_{j=1}^{p} \in \mathbf{W}$ of size $s^{(1)}$. Each filter slides across the single axis of the input vector $\zeta_{\text{in}} := x$ of length n, aggregating local information from sections of $s^{(1)}$ consecutive elements. Assuming each filter iterates one element at a time (*i.e.*, stride = 1), the discrete *convolution* between its filter weights and the input data can be 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 $C_j^{(1)}, j \in [p]$, has length $n - s^{(1)} + 1$. As shown in Figure ..., we then have a 2D *multi-filter convolution* matrix $C^{(1)}$ with size $(n - s^{(1)} + 1) \times p$, whose columns are given by (2.2). Adding bias $b_j^{(1)} \in \mathbf{b}$ and applying an activation map $\phi : \mathbb{R} \to \mathbb{R}$ to the ijth element of $C^{(1)}$ then gives the ijth convolved feature at the output of the first convolutional layer:

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

  In general, the feature map $\zeta^{(1)}$ captures localised, low-level features from the input data. To perform an additional convolution, consider a third-order tensor $W^{(2)} \in \mathbf{W}$ of dimension $s^{(2)} \times p \times q$, where $s^{(2)}$ is the filter size and $p \times q$ denotes the number of filters. We define a 2D multifilter convolution matrix $C^{(2)}$ with size $(n - s^{(2)} + 1) \times q$, whose columns $C_j^{(2)}$ are given by the sum of p convolutions:

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

As before, adding bias $b_j^{(2)} \in \mathbf{b}$ and applying an activation ϕ to the ijth element of $C^{(2)}$ gives the ijth convolved feature at the output of the second convolutional layer:

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

Obtaining the second feature map $\zeta^{(2)}$ by performing a convolution across the first $\zeta^{(1)}$ enables the model to extract higher-order feature representations.

- **Pooling layer**. Using a 1D pooling operation (here, max-pooling), the pooling layer down-samples the feature map by a factor of w, thereby compressing its size and extracting only the strongest features learnt in the convolutional layers. To perform the max-pooling operation, a window of size w moves in strides of the same length along each column of the feature map, extracting the maximum value from each considered sub-region. In particular, its operation can be represented by:

$$\zeta_j^{(3)}(i) = \max\{\zeta_j^{(2)}(i_s)\}_{k=1,2,\ldots,w}, \tag{2.6}$$

  where $i_s = wi - k + 1$ (Cacciari and Ranfagni, 2024). This produces an output vector $\zeta^{(3)}$ of size $w \times q$.
- **Fully connected layers**. A flattening operation is performed, re-shaping the 2D output $\zeta_j^{(3)}$ into a 1D vector f, as follows:

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

  This produces an output vector f of length wq, which can be fed into the fully-conenected network. Assuming the first and second fully-connected layers admit weights and biases $(W^{(4)}, b^{(4)})$ and $(W^{(5)}, b^{(5)})$ in $(\mathbf{W}, \mathbf{b})$, then

$$\zeta^{(4)}(i) = \phi\left(\sum_{j=1}^{wq} W^{(4)}(i,j) f(j) + b^{(4)}(j)\right) \tag{2.8}$$

$$\zeta^{(5)}(i) = \phi\left(\sum_{j=1}^{|\zeta^{(4)}|} W^{(5)}(i,j) \zeta^{(4)}(j) + b^{(5)}(j)\right). \tag{2.9}$$

- **Output layer**. The output vector is simply given by $\zeta_{\text{out}} = \zeta^{(5)}$, corresponding to the network's prediction of the true label (*target*) associated to x.

Whilst common activation functions include sigmoid, tanh and the rectified linear unit (ReLU), our work makes use of the exponential linear unit (ELU) function:

$$\phi(z) = \begin{cases} z & \text{if } z > 0, \\ \alpha(e^z - 1) & \text{if } z \leq 0, \end{cases} \tag{2.10}$$

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

2.1.2 1D-CNNs: Backpropagation Stage

Altogether, the combination of these layers transform the input x into a non-linear output $\hat{f}(x; \mathbf{W}, \mathbf{b})$. 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 predicts and targets:

$$\hat{\mathbf{W}}, \hat{\mathbf{b}} = \text{argmin}_{\mathbf{W},\mathbf{b}} \frac{1}{T} \sum_{i=1}^{T} \|y_i - \hat{f}(x_i; \mathbf{W}, \mathbf{b})\|^2, \tag{2.11}$$

where we neglect regularisation for simplicity (Fabbro et al., 2018). During the training phase, a suitable choice of gradient-based optimiser is deployed to minimise empirical losses of the form (2.11). Using the *backpropagation algorithm*, the algorithm propagates the loss function backwards through the layers of the network, from the output to the input layer, successively computing its gradients with respect to each parameter in the network using the chain rule. As these gradients indicate how much each parameter must be changed to reduce the loss, the algorithm can make informed and iterative adjustments to those weights until reaching a minimum (Cacciari and Ranfagni, 2024). Of note, the *Adam* optimiser is particularly suitable to optimising loss functions, where the underlying datasets are large and/or the parameter spaces are high-dimensional (Kingma and Ba, 2017). With few memory requirements, Adam can efficiently minimise the objective function, using adaptive estimates of low-order gradient moments to adjust learning rates (Kingma and Ba, 2017).

Although 1D-CNNs have achieved state-of-the-art performance, particularly in CV tasks, their computational cost poses a challenge. Having a high number of learnable parameters, 1D-CNNs must be trained on a large volume of labelled training data, which limits their applicability to problems where data sizes are small (Keshari et al., 2018) or where labels are limited, costly or even unavailable.

2.2 Self-supervised Learning

An alternative to SL, SSL has drawn attention for its efficient training of large-scale models without the need for explicit labels. Intuitively, SSL leverages the input data itself as supervision: it uses encoders to find complex structures and co-occurence relationships in the data in order to generate high-quality, discriminative representations of the data. To illustrate, given the incomplete sentence 'I like ______ football', a well-trained NLP model might predict 'playing' as the appropriate blank in this Cloze Test, because 'playing' and 'football' frequently co-ocur in text corpora (Liu et al., 2023). These representations can then be applied to almost all types of downstream tasks, where their performance often exceeds even supervised learning (Chuang et al., 2020).

As summarised in Liu et al. (2023), SSL methods can be broadly classified into three categories:

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

2.2.1 Contrastive Learning

Contrastive learning (CL), the focus of this paper, aims to 'learn to compare' representations across different *modalities*, that is, types of data input. Its central idea is to contrast semantically similar (positive) and dissimilar (negative) pairs of possibly *augmented* views of data points, encouraging representations of similar pairs to be close together and dissimilar pairs to be far apart (Chuang et al., 2020). In this context, an *augmented view* refers to a synthetically modified version of an original data point in order to generate positive pairs for training (Lee et al., 2024). Augmentations introduce semantically meaningful variations to the data, forcing the model to learn invariant features that help it to generalise to downstream tasks (Liu et al., 2024). To illustrate, the Contrastive Language-Image Pretraining (CLIP) method, due to Radford et al. (2021), trains encoders to contrast language based-descriptions with their corresponding image pair, 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 CL, 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 JWST spectra and its APOGEE pair. Apply random augmentations $\mathcal{A}(\cdot)$ and $\tilde{\mathcal{A}}(\cdot)$ 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 $z_i := f_\Phi(\mathcal{A}(x_i))$ and $z'_j := g_\Theta(\tilde{\mathcal{A}}(y_j))$ 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 Information Noise Contrastive Estimation (InfoNCE) loss used to train the encoders is specified by (van den Oord et al., 2019):

$$\mathcal{L}_{\text{InfoNCE}}(f_\Phi, g_\Theta, \tau) = -\frac{1}{N}\sum_{i=1}^{N} \log\left[\frac{\exp(s_{ii}/\tau)}{\exp(s_{ii}/\tau) + \sum_{j\neq i}^{N}\exp(s_{ij}/\tau)}\right]. \tag{2.12}$$

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 particular, the *cosine* similarity measure can be used:

$$s_{ij} := s(z_i, z_j') = \frac{z_i^\top z_j'}{\|z_i\|^2\|z_j'\|^2}. \tag{2.13}$$

By minimising (2.12), the similarity between semantically related features (z_i, z_i') is expected to be large, whilst that between semantically unrelated features $(z_i, z_j'), j \neq i$, is expected to be small.

Of note, InfoNCE has an insightful connection to MI. A formal proof given by van den Oord et al. (2019) shows that:

$$I(z_i; z_j') \geq \log(k) - \mathcal{L}_{\text{InfoNCE}}, \tag{2.14}$$

where k is the number of negative samples to a given sample x_i (for which $z_i = f_\Theta(x_i)$). 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. (2020). Altogether, InfoNCE provides a good lower bound approximation to MI, enjoying greater stability and lower variance than most other competing approaches (Song and Ermon, 2020).

2.2.2 Role of Temperature in Contrastive Learning

Intuitively, the temperature, τ, controls the sensitivity of the InfoNCE loss function. A smaller τ close to 0 increases the penalty applied to *hard*[4] negative samples (HNS), pushing their latent features further away from those of the anchor (Wang et al., 2024). As a result, the local structure of each anchor tends to be more separated, and the embedding distribution appears to be more uniform (Wang and Liu, 2021). Conversely, a large τ results in more tightly-packed semantic clusters, but reduces the sensitivity to negative samples within each cluster (Wang et al., 2024). In practice, τ has been shown to crucially impact the quality of a model's learned embeddings (Wang and Isola, 2020; Wang and Liu, 2021; Robinson et al., 2021).

The phenomenon of larger τ inducing semantic structure in representation space has been demonstrated empirically (Wang and Liu, 2021; Robinson et al., 2021), but is otherwise poorly understood. To gain a little insight, we follow the work of Kukleva et al. (2023) for the remainder, of this section, performing a change-of-variables in (2.12), in which similarities s_{ij} are replaced by 'distances' d_{ij}:

$$d_{ij} = \frac{1 - s_{ij}}{\tau} \tag{2.15}$$

$$c_{ii} = \exp(d_{ii}), \tag{2.16}$$

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

$$\mathcal{L}_c^i = -\log\left[\frac{\exp(-d_{ii})}{\exp(-d_{ii}) + \sum_{j\neq i}^{N}\exp(-d_{ij})}\right] = \log\left[1 + c_{ii}\sum_{j\neq i}^{N}\exp(-d_{ij})\right]. \tag{2.17}$$

[4]A *hard negative sample* (HNS) is defined as a negative sample with high similarity to the anchor; as a corollary, it appears as a *nearest neighbour* to the anchor. On the opposite end, *easy negative sample* (ENS) are easily distinguishable to the anchor by simple patterns, and therefore are more distant in embedding space.

As such, the loss continuously and monotonically increases with the sum of exponential 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 terms d_{ij} in S_i occur when $s_{ij} \sim 1$, which are associated with HNS of the anchor. As a result, the loss function can be understood as maximising the average distance to HNS, leading to a more uniform distribution in embedding space.
- **Large** τ. When $\tau \geq 1$, say, all terms d_{ij} are roughly of the same order of magnitude, so contribute roughly equally to S_i. Hence, the loss function can be thought of as maximising the average distance over a wider range of neighbours. As the number of ENS typically exceeds the number of HNS, ENS collectively provide a larger contribution to this loss. Rather than learning 'hard' features that enable better *instance-discrimination* between HNS, the model is thus 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

We use 19,000 pairs of real APOGEE and synthetic JWST spectra, the latter generated *ab initio*[5] using the method described in Sandford et al. (2020). We use a small sample ($n = 59$) of representative traces of JWST Cycle 3 spectra for data augmentation. All data are described in detail below.

3.1 APOGEE Spectra

We use APOGEE and APOGEE-2 spectra, as provided by the Sloan Digital Sky Survey IV Data Release 17 (SDSS-IV DR17) from January 2021. These observations were captured by the 2.5-m SDSS telescope at Apache Point Observatory, New Mexico, and cover the near-infrared (IR) H-band wavelength range (1.51-1.70μm) at high spectral resolution $R \sim 22,500$ (Majewski et al., 2017). Targets primarily include red giant branch stars and other luminous post-main-sequence stars across the Milky Way, with a focus on obtaining representations from heavily dust-obscured parts of the Galactic bulge, disc and halo (Majewski et al., 2017). Using an automated data reduction pipeline, the raw spectra are then calibrated, continuum-normalised and finally subsampled onto a common wavelength grid with $\Delta\lambda/nR$, where n denotes the number of pixels per resolution element (Nidever et al., 2015).

Stellar parameters and individual chemical abundances are derived from processing APOGEE spectra through the APOGEE Stellar Parameter and Chemical Abundances Pipeline (ASPCAP). Central to ASPCAP is the construction of synthetic spectral grids whose dimensions cover the range of stellar parameters over which most survey stars are found (Holtzman et al., 2015). The synthetic spectral library is generated with 1D LTE `atlas9`/ASSϵT synthesis code, maintained by R. Kurucz (Kurucz, 1979; Kurucz et al., 2009). For each processed APOGEE spectrum, FORTRAN95 code `FERRE` (Prieto et al., 2006) conducts a search through all grids, adopting the parameters that best fit the observation via χ^2-minimisation (Majewski et al., 2017). Where ASPCAP results are likely not reliable, a wrapper sets warning bits in appropriate data quality flags (Holtzman et al., 2015). 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, we remove all bitmasked stars from the APOGEE catalogue. We further streamline our dataset to stars that lie safely within grid edges, for which SNR > 200, $4000 < T_{\text{eff.}} < 5000$ K, $0.5 < \log(g) < 2.0$, $v_{\text{scatter}} < 1.0$ km/s and [Fe/H] > -3. This leaves 19,000 samples for analysis.

[5]That is, generated from theoretical spectrum synthesis models by inputting a set of physical constants.

3.2 JWST Spectra

3.2.1 *Ab initio* Data

Sandford et al. (2020) compute 1D LTE model atmospheres from `atlas12` code maintained by R. Kurucz (Kurucz, 1970, 1993, 2005, 2013, 2017; Kurucz and Avrett, 1981). Assuming 1D mixing length theory (MLT) of convection with $\alpha = 1.25$ and no overshooting, Sandford et al. (2020) generate continuum-normalised spectra for these atmospheres at a high resolution of $R = 300,000$ via the `synthe` radiative transfer code, also due to R. Kurcuz. These spectra are then convolved down to the average resolution of the JWST Near Infrared Spectrograph (NIRSpec) ($R = 2,700$) and sub-sampled onto a common wavelength grid (Sandford et al., 2020).

Using this procedure, Sandford et al. (2020) generate a rectilinear grid of synthetic spectra. We sample 19,000 spectra at desired locations in this synthetic grid, each of whose stellar labels match a corresponding APOGEE spectrum in §3.1, performing third-order interpolation routines between spectra at existing grid points when necessary. In this way, we obtain a database of 19,000 paired JWST-APOGEE spectra.

3.2.2 Real Data

We use a small sample ($n = 59$) of representative traces on the NIRSpec detector, located in an orbit near the second Lagrange Point. The observations are collected as part of the JWST Cycle 3 General Observatory (GO) programme in a 3×3 arcmin field-of-view over near-IR wavelength ranges ($1.0 - 5.0\mu$m) at a resolution $R = 2,700$ (Jakobsen et al., 2022). As illustrated in Figure 2, the traces exhibit missing wavelength segments due to physical gaps between active areas of detector arrays.

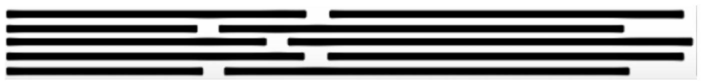

Figure 2: Sample of 5 representative traces of spectra on the JWST/NIRSpec detector, exhibiting wavelength segments lost to detector gaps. The traces span the full wavelength band and are widened for clarity.

4 Model Implementation

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

(i) We separately pre-train two single-modal spectral encoders in a supervised setting, using carefully modified versions of the StarNet CNN (Fabbro et al., 2018).
(ii) We 'fine-tune' our pre-trained encoders under CLIP loss to align cross-modal embeddings around shared semantics, using a carefully modified version of AstroCLIP (Parker et al., 2024).

The final StarCLIP model, and its variant MockStarCLIP, are a sort of *compositional* model that consists of both fine-tuned spectral encoders. Notably, this strategy of using pre-trained encoders as initialisers in training, rather than training the entire SSL model from scratch, has been shown to partially mitigate instabilities and high computational costs associated with CLIP (Sun et al., 2023). In what follows, we provide details on the implementation and training of these models.

4.1 Spectral Encoder Model

4.1.1 Implementation

Our spectral encoder is closely modelled after StarNet (Fabbro et al., 2018). The CNN model architecture we implement has two convolutional layers, followed by three fully-connected layers, which together perform stellar property prediction of 20 stellar parameters.

Figure ... shows the sequential operations required to transform *single-modal* spectral data to a stellar property prediction. The left-most block represents the input data $\{x\}$, $|x| := n$, collected from either APOGEE ($n = 8,575$) or JWST ($n = 8,192$). The first layer performs a 1D-convolution on the data using 1 input channel and 4 output channels, using $p = 4$ learnable filters of size $s^{(1)} = 8$ (and stride = 1). The second layer performs an additional 1D-convolution on each channel of the previous layer, resulting in 16 output channels, using $p \times q = 16$ learnable filters of size $s^{(2)} = 8$. Following each convolution, the ELU activation function is applied to introduce non-linearities in the model. Following the second convolution *only*, a max-pooling operation with window length $w = 4$ slides over the data vectors, reducing their dimensionality by a factor of 4, and a flattening operation further collapses them into a single dimension. The flattened data vectors are then mapped to three fully-connected layers with $N_1 = 256$, $N_2 = 128$ and $N_3 = 20$ hidden units, respectively, each with their own weight parameters and biases. ELU is applied following each operation except the last, which corresponds to the output stellar property $\hat{Y}$.

Though we adopt the model architecture and hyperparameter selection (p, q, w, N_1, N_2) from Fabbro et al. (2018), we distinguish our work in the following ways:

- We use ELU rather than ReLU (or sigmoid or tanh), having empirically found it to accelerate the convergence of the loss functions most rapidly.
- We introduce a learning rate scheduler to reduce training time.
- We predict 20 rather than 3 stellar parameters with high precision and accuracy.

4.1.2 Training Procedure

During training, the CNN regression model learns to leverage single-modal spectral data to predict stellar labels. The choice of loss function, $\mathcal{L}$, comparing the difference between true and predicted targets, is the MSE:

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

where N_{batch} is the number of data samples in the batch and where Y_i and $\hat{Y}_i$ are the true and predicted stellar property of interest, respectively. To have each parameter weighted equally, we normalise the distribution of each stellar property to have approximately zero mean and unit variance.

The single-modal training data are first batched into sets of $N_{\mathrm{batch}} = 64$ stars. While the model weights θ are randomly initalised, the network implements a training procedure to update the weights, as follows: (1) forward pass the batch through the network to compute $\hat{Y}_{\mathrm{batch}}$, (2) compute $\mathcal{L}$ according to Eq. (4.1), (3) backpropagate $\mathcal{L}$ through each layer of the network, (4) compute the gradient $\Delta_\theta \mathcal{L}$ and (5) update the weights to minimise $\mathcal{L}$ (Blancato et al., 2020). Steps (1) through (5) are repeated for each training batch iteration, resulting in a steady convergence of the training loss over $N_{\mathrm{epochs}} = 15$. Specifically, step (5) is carried out using `PyTorch`'s implementation of AdamW ($\eta_0 = 0.001$), with StepLR scheduler (step size = 5, $\gamma = 0.1$). Altogether, for 11,875 single-modal training spectra and 8.8 million NN parameters, the training converged in $\sim$10 minutes, using ... GPUs.

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 in order to provide a *diagnostic* of how generalisable the model is to unseen data. As a further use of the validation set, future work could explore performing grid searches over a heuristic choice of hyperparameters $\{(\eta_0, \text{step size}, \gamma)\}$, choosing the combination that yields the best performance on the validation set–a clear improvement over choosing the hyperparameters *ad hoc*.

4.1.3 Model Evaluation

The model is evaluated on the held-out test set of 4,751 spectra, and the residuals between true and predicted targets are shown in Figure ..., quantified with location (mean) and spread (standard deviation) summary statistics. To further assess its performance, we compute two evaluation metrics:

the bias Δ and the root-mean-squared error (RMSE), respectively computed as:

$$\Delta = \frac{1}{N}\sum_i \left(\hat{Y}_i - Y_i\right) \tag{4.2}$$

$$\text{RMSE} = \left[\frac{1}{N}\sum_i \left(Y_i - \hat{Y}_i\right)^2\right]^{\frac{1}{2}}, \tag{4.3}$$

where N is the number of observations in the validation or test set and Y and $\hat{Y}$ are the true and predicted targets of the stellar property of interest, respectively. In general, smaller bias and RMSE indicate better model performance.

4.2 StarCLIP Model

4.2.1 Spectrum Preprocessing

For each JWST spectrum, (a selection of) the following set of stochastic transformations is applied in order to simulate physical corruptions in *real* JWST data:

(i) *Template Imputation.* We identify the four Paschen series spectral lines that fall in the wavelength range covered by NIRSpec: $9548.6\mathring{A}, 10052.1\mathring{A}, 10941.1\mathring{A}$ and $12821.6\mathring{A}$. For each wavelength, we isolate an ϵ-neighbourhood of $\epsilon = 10$ data points in a *fixed*, arbitrarily-chosen spectrum of the database. Given another JWST spectrum, we then then replace the corresponding neighbourhoods with the appropriate 'template' values, thereby 'standardising' all spectral features near the Paschen series.

(ii) *Denormalisation.* We denormalise the spectrum by multiplying it by a low-amplitude third-order polynomial:

$$P(\lambda) = 1 + A\cdot\frac{\sum_{i=0}^{d} c_i\tilde{\lambda}^i - \mu}{\max|\sum_{i=0}^{d} c_i\tilde{\lambda}^i - \mu|}, \quad \tilde{\lambda} = \frac{\lambda - \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}} \in [0,1], \tag{4.4}$$

where $A = 0.01$, $d = 3$, $c_i \in \mathcal{U}(-1,1)$, $\lambda_{\min} = ..., \lambda_{\max} = ...$ and μ is the mean value of the unnormalised polynomial evaluated across all normalised wavelengths $\tilde{\lambda}$.

(iii) *Pixel Masking.* We select uniformly at random 0.1% of pixels to set to 1[6], as this is roughly the expected fraction of non-operable pixels in the NIRSpec detectors (Bker et al., 2023).

(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 SNR $\sim \mathcal{U}(30, 300)$, the expected signal-to-noise (SNR) range for NIRSpec detectors.

(v) *Wavelength Masking.* We convert the image of the traces (*cf.* Figure 2) to grayscale, applying an automated routine to locate the gap boundaries in each trace. We measure the gap's x-position and width, converting these from pixel space to wavelength space. For each parameter ($\{x_{\text{gap}}\}, \{w_{\text{gap}}\}$), we fit a Gaussian kernel density estimator (KDE) to obtain a smooth probability density curve describing the distribution of the empirical measurements. We then draw *non-negative* positions and widths of gaps, setting the identified intervals to 1 to simulate missing data between active areas of the detector arrays.

We augment the original JWST database five times, applying to each augmentation the transformations (iv) and (v), resulting in a 'mock' JWST dataset. We simultaneously replicate the APOGEE database five times in order to obtain a paired mock JWST–APOGEE dataset of 95,000 samples that we split into training, validation and test sets in a 5:1:2 ratio.

To mimic the small sample sizes of JWST ($n \sim 100$), we partition the original JWST database into 15 bins based on [Fe/H] values. We then randomly sample $n = 500$ indices that are distributed as uniformly as possible across all bins. Applying two distinct sets of transformations on this sampled data gives a mock JWST sample, obtained by strictly applying (iv) and (v), and a so-called 'extended' mock JWST sample, obtained by applying all transformations (i)-(v). As before, we split the paired mock JWST–extended-mock JWST into training, validation and test sets in a 5:1:2 ratio.

[6] We choose unity rather than to 0 (or NaN), so the masked data remain on the same (non-zero) scale as the unmasked data. This helps to mitigate discontinuities in the flux profile of the data.

4.2.2 Implementation

The compositional StarCLIP model is constructed as follows. Given an observation (x_i, y_i) with associated label $l = \{\text{‘Mock JWST’}, \text{‘APOGEE’}\}$, the model processes the input through the corresponding pre-trained spectral encoder, whose weights are fully *unfrozen* during training. The input is thus mapped to a pair of 20-dimensional embeddings in a shared, unified latent space, that can be subsequently aligned under CLIP training, discussed in §4.2.3.

MockStarCLIP initalises its spectral encoders with the learned weights of the StarCLIP mock JWST encoder. An input (x_i, y_i) with associated label $l = \{$‘Sample Mock JWST’, ‘Sample Extended-Mock JWST’$\}$ is fed into the appropriate spectral encoder. While the mock encoder is fully frozen, the fully-connected layers of the extended-mock encoder are *unfrozen*, allowing the model to adapt to physical corruptions in the data. The input is again mapped to a pair of 20-dimensional embeddings in the *same* latent space, with alignment performed during CLIP training.

4.2.3 Training

We train the StarCLIP (MockStarCLIP) model on the training split of our paired datasets. We use small batch sizes of $K =$ ($K =$) pairs, having empirically found it ...

Following Parker et al. (2024), we train StarCLIP using the Adam optimiser with a base learning rate of $\eta_0 = 0.007$ ... and a weight decay of ...

We train our model for 30 epochs on a single ... GPU, which results in roughly 1 hour of training time. We set the logit scale in our loss to a fixed value of 15.5.

By computing the InfoNCE Loss (*cf.* Eq. 2.12) between pairs of spectra,

5 Results

5.1 Spectral Encoder Model

5.2 StarCLIP Model

StarCLIP and MockStarCLIP extract useful latent representations that can be seamlessly deployed to downstream tasks, with zero-shot transfer or minimal task adaptation. To demonstrate this, we first embed paired spectra in the *held-out test database*, ℓ_2-normalising the resulting feature vectors.

5.2.1 Example Retrieval by Similarity Search

We perform example retrieval by cosine similarity search. This sort of search has proven useful in retrieving interesting objects and performing clustering, as exemplified in Popat et al. (2017).

Give some queried spectra, we assign a cosine similarity score (**??**) between its normalised embedding $\hat{\mathbf{z}}_Q$ and all other embeddings in the held-out test set. From the pool of scores, we retrieve the spectra with the highest similarity scores to its query, completing the task; no additional transformation or processing steps are required. We call a search constrained to same-modality embeddings (e.g., $S(\hat{\mathbf{z}}_Q^J, \hat{\mathbf{z}}^J)$) an **in-modality similarity search**, and a search constrained to cross-modality embeddings (e.g., $S(\hat{\mathbf{z}}_Q^J, \hat{\mathbf{z}}^A)$) a **cross-modality similarity search**. By design, an in-modality search requires the closest neighbour to the query to be the query itself.

For all four possible pairs of modalities, we present the spectra of the ‘closest’ match retrieved by StarCLIP and mStarCLIP in Figure ... We report the averages of the highest cosine similarity score obtained during searches in Table ... Overall, the closeness of these mean scores to 1 indicate StarCLIP and MockStarCLIP are efficient in retrieving semantically similar spectra within and across modalities.

5.2.2 Stellar Property Estimation

We evaluate StarCLIP and MStarCLIP's performance on stellar property estimation using JWST and mock JWST embeddings as inputs, respectively. In general, these tasks involve developing an entire pipeline and training bespoke models end-to-end from stratch Parker et al. (2024). However, because learned representations already capture key physical information about the input stars, we can easily use zero- and one-shot algorithms to predict stellar properties Parker et al. (2024). Specifically, we use zero-shot k-Nearest Neighbours (k-NN) and one-shot linear and Extreme Gradient Boosting (XGBoost) to regress 'true' stellar parameters against corresponding model predictions.

... some training details ...

We present our results in Figure ... . We report the R^2 performance of all regressors in Table ...

Consistent with Parker *et al.* Parker et al. (2024), strong zero- and one-shot performance of the models indicate that, under CLIP alignment, JWST and mock JWST embeddings are organised in latent space around stellar properties. This is consistent with our intuition that spectra should be naturally informative about the values of their stellar parameters.

5.2.3 Manifold Learning

The results in §5.2.1 and §5.2.2 indicate AstroCLIP and MockAstroCLIP are able to capture in its embeddings core physical properties of the input spectra. To further explore the structure of the embedding space, we apply a state-of-the-art **manifold learning** method: t-distributed Stochastic Neighborhood Embedding (t-SNE). Intuitively, t-SNE works by compressing high-dimensional data onto a lower dimensional manifold in such a way as to preserve much of the local geometric structure and reveal global structure, including the presence of clusters.

Low-dimensional representations are produced by minimising the K-L divergence between ...

Moreover, t-SNE has $O(n^2)$ complexity, making it computationally slow for the large datasets required in the AstroCLIP model.

to capture neighbourhood relationships that exist in the higher-dimensional latent space and

6 Conclusion

In this paper, we present StarCLIP and its variant MockStarCLIP, cross-modal foundation models for stellar spectroscopy.

We demonstrate the utility of StarCLIP in downstream tasks

6.1 Limitations

6.2 Broader Impact

StarNet Fabbro et al. (2018)

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 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 (Astropy Collaboration, 2022), h5py (Collette, 2013), matplotlib (Hunter, 2007), numpy (van der Walt et al., 2011; Harris et al., 2020), pandas (McKinney, 2010; Reback

et al., 2022), RANDOM (Van Rossum, 2020), SCIPY (Virtanen et al., 2020), SKLEARN (Pedregosa et al., 2011) and TORCH (Paszke et al., 2019). Illustrations were created using bioRender.[7]

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.

8 Data Availability

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

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

References

Astropy Collaboration. Astropy collaboration, 2022. *The Astrophysical Journal*, 935:167, 2022.

Dalya Baron. Machine learning in astronomy: a practical overview, 2019. URL https://arxiv.org/abs/1904.07248.

Kirsten Blancato, Melissa Ness, Daniel Huber, Yuxi Lu, and Ruth Angus. Data-driven derivation of stellar properties from photometric time series data using convolutional neural networks, 2020. URL https://arxiv.org/abs/2005.09682.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher R, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022. URL https://arxiv.org/abs/2108.07258.

Bernhard Boser, Yann LeCun, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*. Citeseer, 1990.

T. Bker, T. L. Beck, S. M. Birkmann, G. Giardino, C. Keyes, N. Kumari, J. Muzerolle, T. Rawle, P. Zeidler, Y. Abul-Huda, C. Alves de Oliveira, S. Arribas, K. Bechtold, R. Bhatawdekar, N. Bonaventura, A. J. Bunker, A. J. Cameron, S. Carniani, S. Charlot, M. Curti, N. Espinoza, P. Ferruit, M. Franx, P. Jakobsen, D. Karakla, M. Lpez-Caniego, N. Ltzgendorf, R. Maiolino, E. Manjavacas, A. P. Marston, S. H. Moseley, P. Ogle, M. Perna, M. Pea-Guerrero, N. Pirzkal, R. Plesha, C. R. Proffitt, B. J. Rauscher, H.-W. Rix, B. Rodrguez del Pino, Z. Rustamkulov, E. Sabbi, D. K. Sing, M. Sirianni, M. te Plate, L. beda, G. M. Wahlgren, E. Wislowski, R. Wu, and Chris J. Willott. 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, March 2023. ISSN 1538-3873. doi: 10.1088/1538-3873/acb846. URL http://dx.doi.org/10.1088/1538-3873/acb846.

I. Cacciari and A. Ranfagni. Hands-on fundamentals of 1d convolutional neural networksa tutorial for beginner users. *Applied Sciences*, 14(18):8500, 2024. doi: 10.3390/app14188500. URL https://doi.org/10.3390/app14188500.

Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning, 2020. URL https://arxiv.org/abs/2007.00224.

Andrew Collette. *Python and HDF5*. O'Reilly, 2013.

S. Fabbro, K. A. Venn, T. O'Briain, S. Bialek, C. L. Kielty, F. Jahandar, and S. Monty. An application of deep learning in the analysis of stellar spectra. *Monthly Notices of the Royal Astronomical Society*, 475(3):2978–2993, April 2018. doi: 10.1093/mnras/stx3298. URL https://doi.org/10.1093/mnras/stx3298.

J. P. Gardner, J. C. Mather, and M. et al. Clampin. The james webb space telescope. *Space Science Reviews*, 123(4):485–606, April 2006. doi: 10.1007/s11214-006-8315-7. URL https://doi.org/10.1007/s11214-006-8315-7.

Charles R. Harris, K. Jarrod Millman, Stfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernndez del Ro, Mark Wiebe, Pearu Peterson, Pierre Grard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357362, September 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. URL http://dx.doi.org/10.1038/s41586-020-2649-2.

Jon A. Holtzman, Matthew Shetrone, Jennifer A. Johnson, Carlos Allende Prieto, Friedrich Anders, Brett Andrews, Timothy C. Beers, Dmitry Bizyaev, Michael R. Blanton, Jo Bovy, et al. Abundances, stellar parameters, and spectra from the sdss-iii/apogee survey. *The Astronomical Journal*, 150(5):148, 2015. doi: 10.1088/0004-6256/150/5/148.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Peter Jakobsen, P. Ferruit, S. Arribas, G. Bagnasco, R. Barho, Tracy Beck, Stephan Birkmann, Torsten Bker, A. Bunker, S. Charlot, P. Jong, G. Marchi, R. Ehrenwinkler, M. Falcolini, R. Fels, M. Franx, David Franz, M. Funke, and C. Zincke. The near-infrared spectrograph (nirspec) on the james webb space telescope i. overview of the instrument and its capabilities, 02 2022.

Rohit Keshari, Mayank Vatsa, Richa Singh, and Afzel Noore. Learning Structure and Strength of CNN Filters for Small Sample Size Training. *arXiv e-prints*, art. arXiv:1803.11405, March 2018. doi: 10.48550/arXiv.1803.11405.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Jayanth Koushik. Understanding convolutional neural networks, 2016. URL https://arxiv.org/abs/1605.09081.

Anna Kukleva, Moritz Bhle, Bernt Schiele, Hilde Kuehne, and Christian Rupprecht. Temperature schedules for self-supervised contrastive methods on long-tail data, 2023. URL https://arxiv.org/abs/2303.13664.

R. L. Kurucz. Saosr, 309. *SAOSR*, 309, 1970.

R L Kurucz. Model atmospheres for g, f, a, b, and o stars. *Astrophys. J., Suppl. Ser.; (United States)*, 40:1, 05 1979. doi: 10.1086/190589. URL https://www.osti.gov/biblio/6183335.

R. L. Kurucz. *SYNTHE Spectrum Synthesis Programs and Line Data*. Smithsonian Astrophysical Observatory, Cambridge, MA, 1993.

R. L. Kurucz. Msais, 8, 14. *MSAIS*, 8:14, 2005.

R. L. Kurucz. Atlas12: Opacity sampling model atmosphere program. Astrophysics Source Code Library, 2013. ascl:1303.024.

R. L. Kurucz. Atlas9: Model atmosphere program with opacity distribution functions. Astrophysics Source Code Library, 2017. ascl:1710.017.

R. L. Kurucz and E. H. Avrett. Saosr, 391. *SAOSR*, 391, 1981.

Robert L. Kurucz, Ivan Hubeny, James M. Stone, Keith MacGregor, and Klaus Werner. Including all the lines. In *AIP Conference Proceedings*, page 4351. AIP, 2009. doi: 10.1063/1.3250087. URL http://dx.doi.org/10.1063/1.3250087.

J. Lee, I. Ham, Y. Kim, and H. Ko. Time-series representation feature refinement with a learnable masking augmentation framework in contrastive learning. *Sensors*, 24(24):7932, 2024. doi: 10.3390/s24247932. URL https://doi.org/10.3390/s24247932.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876, 2023. doi: 10.1109/TKDE.2021.3090866.

Z. Liu, M. Shu, and W. Zhu. Contrastive learning framework for bitcoin crash prediction. *Stats*, 7 (2):402–433, 2024. doi: 10.3390/stats7020025. URL https://doi.org/10.3390/stats7020025.

Steven R. Majewski, Ricardo P. Schiavon, Peter M. Frinchaboy, Carlos Allende Prieto, Robert Barkhouser, Dmitry Bizyaev, Basil Blank, Sophia Brunner, Adam Burton, and Ricardo Carrera. The apache point observatory galactic evolution experiment (apogee). *The Astronomical Journal*, 154(3):94, 2017. doi: 10.3847/1538-3881/aa784d.

W. McKinney. Data structures for statistical computing in python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010.

David L. Nidever, Jon A. Holtzman, Carlos Allende Prieto, Stephane Beland, Chad Bender, Dmitry Bizyaev, Adam Burton, Rohit Desphande, Scott W. Fleming, Ana E. Garca Prez, et al. The data reduction pipeline for the apache point observatory galactic evolution experiment. *The Astronomical Journal*, 150(6):173, 2015. doi: 10.1088/0004-6256/150/6/173.

David L. Nidever, Karoline Gilbert, Erik Tollerud, Charles Siders, Ivanna Escala, Carlos Allende Prieto, Verne Smith, Katia Cunha, Victor P. Debattista, Yuan-Sen Ting, and Evan N. Kirby. The prevalence of the α-bimodality: First jwst α-abundance results in m31, 2023. URL https://arxiv.org/abs/2306.04688.

Liam Parker, Francois Lanusse, Siavash Golkar, Leopoldo Sarra, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Geraud Krawezik, Michael McCabe, Rudy Morel, Ruben Ohana, Mariel Pettee, Bruno Rgaldo-SaintBlancard, Kyunghyun Cho, and Shirley Ho. Astroclip: a cross-modal foundation model for galaxies. *Monthly Notices of the Royal Astronomical Society*, 531(4):49905011, June 2024. ISSN 1365-2966. doi: 10.1093/mnras/stae1450. URL http://dx.doi.org/10.1093/mnras/stae1450.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Shraddha K. Popat, Pramod B. Deshmukh, and Vishakha A. Metre. Hierarchical document clustering based on cosine similarity measure. In *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, pages 153–159, 2017. doi: 10.1109/ICISIM.2017.8122166.

Carlos Allende Prieto, Timothy C. Beers, Ronald Wilhelm, Heidi Jo Newberg, Constance M. Rockosi, Brian Yanny, and Young Sun Lee. 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, 2006. doi: 10.1086/498131.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Jeff Reback, Jbrockmendel, Wes McKinney, Joris Van Den Bossche, Matthew Roeschke, Tom Augspurger, Simon Hawkins, Phillip Cloud, Gfyoung, Sinhrks, Patrick Hoefler, Adam Klein, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Shahar Naveh, JHM Darbyshire, Richard Shadrach, Marc Garcia, Jeremy Schendel, Andy Hayden, Daniel Saxton, Marco Edward Gorelli, Fangchen Li, Torsten Wörtwein, Matthew Zeitlin, Vytautas Jancauskas, Ali McMaster, and Thomas Li. pandas-dev/pandas: Pandas 1.4.3, June 2022.

I. Reis, M. Rotman, D. Poznanski, J.X. Prochaska, and L. Wolf. Effectively using unsupervised machine learning in next generation astronomical surveys. *Astronomy and Computing*, 34:100437, 2021. ISSN 2213-1337. doi: 10.1016/j.ascom.2020.100437. URL https://www.sciencedirect.com/science/article/pii/S2213133720300913.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples, 2021. URL https://arxiv.org/abs/2010.04592.

Nathan R. Sandford, Daniel R. Weisz, and Yuan-Sen Ting. Forecasting chemical abundance precision for extragalactic stellar archaeology. *The Astrophysical Journal Supplement Series*, 249(2):24, July 2020. ISSN 1538-4365. doi: 10.3847/1538-4365/ab9cb0. URL http://dx.doi.org/10.3847/1538-4365/ab9cb0.

M. Scalco, M. Libralato, R. Gerasimov, L. R. Bedin, E. Vesperini, D. Nardiello, A. Bellini, M. Griggio, D. Apai, M. Salaris, A. Burgasser, and J. Anderson. Jwst imaging of the closest globular clusters - iii. multiple populations along the low-mass main-sequence stars of ngc 6397. *Astronomy Astrophysics*, 689:A59, 2024. doi: 10.1051/0004-6361/202450589.

Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators, 2020. URL https://arxiv.org/abs/1910.06222.

Space Telescope Science Institute (STScI). Jwst cycle 2 general observer abstracts, 2023. URL https://www.stsci.edu/files/live/sites/www/files/home/jwst/science-execution/approved-programs/general-observers/cycle-2-go/_documents/jwst-cycle2-general-observer-abstracts.pdf. Accessed: 2024-12-15.

STScI. Jwst cycle 3 general observer abstracts, 2024. URL https://www.stsci.edu/files/live/sites/www/files/home/jwst/science-execution/approved-programs/general-observers/cycle-3-go/_documents/jwst-cycle3-general-observer-abstracts.pdf. Accessed: 2024-12-15.

Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. Eva-clip: Improved training techniques for clip at scale, 2023. URL https://arxiv.org/abs/2303.15389.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding, 2020. URL https://arxiv.org/abs/1906.05849.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011. doi: 10.1109/MCSE.2011.37.

Guido Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

P. Virtanen et al. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss, 2021. URL https://arxiv.org/abs/2012.09740.

Jing Wang, Wonho Bae, Jiahong Chen, Kuangen Zhang, Leonid Sigal, and Clarence W. de Silva. What has been overlooked in contrastive source-free domain adaptation: Leveraging source-informed latent augmentation within neighborhood context, 2024. URL https://arxiv.org/abs/2412.14301.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In Hal Daum III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceed-*

ings of Machine Learning Research, pages 9929–9939. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/wang20k.html.

R. Yamashita, M. Nishio, R.K.G. Do, and K. Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9:611–629, Aug 2018. doi: 10.1007/s13244-018-0639-9. URL https://doi.org/10.1007/s13244-018-0639-9.

Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings, 2019. URL https://arxiv.org/abs/1903.12355.