Imperial College London

Department of Computing

Imperial College of Science, Technology and Medicine

Multi-Modal Data Fusion Through Contrastive Learning in Geoscience

Supervisor:
Dr Sibo Cheng

Author:
Nathan Bailey

Second Marker:
Dr César Quilodrán-Casas

Submitted in partial fulfillment of the requirements for the MSc Computing - Artificial Intelligence and Machine Learning of Imperial College London

Abstract

Your abstract goes here

Acknowledgements

I want to express my sincere gratitude to my supervisor, Dr. Sibo Cheng. His insightful guidance and support have been invaluable to the project's success.

On a personal note, I am deeply grateful to my family and friends for their unwavering support and encouragement. I would especially like to thank my parents, whose enduring belief in me throughout the ups and downs of this degree was a foundational source of strength.

Contents

1 Introduction **2**
1.1 Motivation . . . 2
1.1.1 Self-Supervised Learning . . . 2
1.1.2 Application to Geoscience . . . 2
1.1.3 Project Scope . . . 3
1.2 Contributions . . . 3
1.3 Report Structure . . . 4

2 Literature Review **5**
2.1 Current Contrastive Learning Methods . . . 5
2.1.1 Computer Vision . . . 5
2.1.2 Time-Series Data . . . 8
2.1.3 Multi-Modal Fusion . . . 10
2.2 Contrastive Learning Methods in Geoscience . . . 12
2.3 Current Methods Tackling Sparsity in Geoscience . . . 14
2.4 Decoder Based Contrastive Learning . . . 15
2.5 Diffusion Models . . . 17
2.5.1 Latent Diffusion Models . . . 18
2.5.2 Conditional Diffusion Models . . . 18
2.6 Analysis of Latent Spaces . . . 19
2.6.1 Smoothness . . . 19
2.6.2 Generative Quality . . . 19
2.7 Summary and Research Area . . . 20

3 ERA5 Dataset **22**

4 Part 1 - Exploration of Contrastive Learning Methods **23**
4.1 Dataset Setup . . . 23
4.2 Contrastive Learning . . . 23
4.2.1 Barlow Twins . . . 24
4.2.2 SimCLR . . . 24
4.2.3 Supervised Contrastive Learning . . . 25
4.3 Autoencoder . . . 26
4.4 Conclusions . . . 27

5 Part 2 - Designing and Training a Decoder **28**
5.1 Dataset Setup . . . 28
5.2 Introduction of Sparse Data . . . 28
5.3 Introduction of Decoder . . . 29

5.4 Evaluating on Forecasting Downstream Task . . . 30
5.5 Investigation into Batch Normalisation . . . 31
5.6 Conclusions . . . 33

6 Part 3 - Investigation into Robustness 34
6.1 End-to-end evaluation of the Downstream Task . . . 34
6.1.1 Model Evaluation . . . 35
6.1.2 Matérn Kernel Noise . . . 35
6.1.3 Initial Results . . . 36
6.2 Improving the Robustness of Contrastive Learning . . . 37
6.2.1 Varied Mask Ratios . . . 37
6.2.2 Temporal Structure of Data Points . . . 38
6.2.3 Combining Approaches . . . 39
6.3 Results . . . 42
6.4 Conclusions . . . 43

7 Part 4 - Additional Downstream Tasks and Improvements to the Decoder 44
7.1 Improvements to the Decoder . . . 44
7.1.1 Specific Reconstruction Losses . . . 44
7.1.2 Comparison to Autoencoder . . . 45
7.2 Autoregressive Performance . . . 45
7.3 Further Improvements to Contrastive Learning . . . 46
7.3.1 Decoding SIMCLR Embeddings . . . 46
7.4 Conditional Latent Diffusion Model . . . 48
7.5 Latent Classification Model . . . 49
7.6 Conclusions . . . 50

8 Part 5 - Analysis of the Latent Space 51
8.1 Introduction . . . 51
8.2 Analysis Setup . . . 51
8.3 Initial Analysis . . . 51
8.3.1 Autoencoder . . . 51
8.3.2 SimCLR . . . 53
8.4 Improvements . . . 54
8.4.1 Batch Size . . . 54
8.4.2 Restricting the Decay of Alpha . . . 55
8.4.3 Tweaks to Cycle Consistency Loss . . . 57
8.4.4 Results . . . 58
8.5 Conclusions . . . 59

9 Part 6 - Multi-Modal Fusion 60

10 Conclusions and Future Work 61

11 Declarations 68

A Dataset and Augmentation Examples 69
A.1 Example ERA5 Data Sample . . . 69
A.2 Sparse Positive Pair . . . 69

B Core Model Architectures **71**
B.1 Hybrid Encoder-Decoder Architecture - Chapter 5 71
B.2 Final Robust Model Architecture with Cycle Loss - Chapter 6 71
B.3 Downstream LSTM Forecasting Model 72
B.3.1 Latent Space 72
B.3.2 Decoded to Full Space 73
B.4 Conditional Latent Diffusion U-Net Architecture 73

C Supplementary Results **74**
C.1 Histogram of Per-Channel Reconstruction Errors 74
C.2 Absolute Errors 74
C.2.1 Absolute Errors - Autoencoder 75
C.2.2 Absolute Errors - SimCLR 76
C.3 Autoregressive Results 76
C.3.1 Autoencoder - L1 + L2 Decay 77
C.3.2 Autoencoder - L1 + L2 Decay + Sampling 80
C.3.3 SIMCLR - New MSE Loss 83
C.3.4 SIMCLR - Decoding Embeddings 86
C.3.5 SIMCLR - 256 Batch Size, Cap Alpha, Improved Cycle Loss 89
C.4 Latent Analysis of Context Windows 92
C.4.1 Context Window 30, Stride 1 92
C.4.2 Context Window 5, Stride 10 92

D Multi-Modal Fusion Architectures **93**
D.1 Average Pooling Fusion Model 93
D.2 Self-Attention Fusion Model 93
D.3 Graph Neural Network Fusion Model 94

Chapter 1

Introduction

1.1 Motivation

1.1.1 Self-Supervised Learning

Self-supervised learning (SSL) is a common technique primarily used in computer vision to generate meaningful and discriminative representations of unlabelled data invariant to transformations (i.e., strong generalisation capabilities) [1]. One goal of SSL is to utilise data where labels are sparse, and leverage the inherent structures in the data to form useful representations, which can be used to improve the performance of downstream tasks on labelled data [2, 3].

Contrastive learning, a particular type of self-supervised learning, aims to learn data representations by contrasting negative and positive data samples. In this way, within the resulting representation space, it pulls positive samples close and pushes negative samples away, creating structure within the latent space [4]. SimCLR [5], for example, creates a positive pair of data by augmenting a data sample twice, and uses an NT-Xent loss function to encourage the representations of positive pairs to be close whilst pushing away negative samples. Through this method, the model learns high-quality embeddings of data that can be used for downstream tasks [4].

While primarily used in computer vision (CV) and natural language processing (NLP), contrastive learning has also been applied to time-series data. For example, Yue et al. [6] use the representations of the same timestamp as positives in TS2Vec and employ a joint temporal and instance contrastive loss to align positive samples in the latent space.

1.1.2 Application to Geoscience

Contrastive learning lends itself naturally to geoscience data, particularly weather analysis datasets, as these are often collected without labels. Weather analysis datasets are time-series data, whereby measurements of various data variables are collected across latitude and longitude at varying degrees of resolution [7].

Downstream tasks can be performed using weather analysis datasets. A common task is numerical weather prediction (NWP), which traditionally is performed by simulating the atmosphere by solving physical equations [7, 8]. Machine learning methods offer an alternative to this. For example, Bi et al propose a 3D transformer-based neural network for weather forecasting [9]. While downstream tasks, such as weather forecasting, can be performed on the raw data, due to the fine horizontal resolution at which this data is collected, the samples reside in high dimensions, which increases the challenge

of performing these tasks, given the curse of dimensionality. For example, the common reanalysis dataset ERA5 has a lower-bound resolution of 64x32, reaching a maximum of 1440×721. Bi et al [9] use some level of compression, but still operate in high dimensions.

Moreover, weather analysis datasets often consist of multiple data variables (called modes). ERA5 consists of 62 data variables, some of which span up to 137 pressure levels [10]. This substantially increases the dimensionality of the data, making downstream performance on the raw data samples substantially harder.

1.1.3 Project Scope

The majority of current self-supervised learning methods focus on a single mode of data, such as images or text. The presence of multiple data modes introduces a further challenge within contrastive learning, and therefore, within weather analysis datasets, on how to best merge these modes to create a stable and structured representation that improves downstream performance. Fusing modes is important, since it combines the strengths of each mode, improving performance [11].

While contrastive learning for weather analysis datasets has been explored in current work, an extensive investigation into this topic has not been conducted. Therefore, contrastive learning provides a means to create significantly smaller and more structured representations of weather data, thereby reducing dimensionality and enhancing downstream performance.

Finally, current self-supervised methods for geoscience data do not account for the fact that incoming data may be sparse and not all samples will be accessible. Typically, satellite-borne sensors yield sparse observations of the Earth's atmosphere, posing challenges when utilising the data for specific applications [12]. Having a robust framework that can align sparse data to full observations within the latent space would, in theory, enable little to no performance change for resulting downstream tasks.

This project aims to explore contrastive learning for geoscience data, developing a scalable and robust framework for multimodal data fusion to enhance predictive accuracy for downstream tasks. We strive to create a structured latent space that is invariant to data sparsity, enabling downstream tasks to be performed even with sparse observations.

1.2 Contributions

As mentioned above, this project aims to explore contrastive learning in the context of geoscience, highlighting the potential performance gains that a structured latent space through contrastive learning could create over a simple autoencoder. We focus on weather data, specifically the ERA5 dataset.

To achieve this goal, we make the following contributions:

1. **We construct a end-to-end contrastive learning model**. In chapter 4, we first explore various contrastive learning methods. In chapter 5, we build upon the literature, integrating a decoder into the training process to create an end-to-end system that enables downstream predictions to be utilised in the original space. Through this, we highlight how batch normalisation harms the embeddings produced by contrastive learning.

2. **We explore and contrast various methods of improving the robustness of the latent space**. In chapter 6, we investigate and improve the robustness of the latent space to the downstream task of forecasting. Demonstrating that a focus on how points are sampled in a batch is crucial for creating a more robust latent space.

3. **We highlight the benefits of contrastive learning to a multitude of downstream tasks**. To achieve our goal of creating a robust model and, consequently, a robust latent space, we extend the model to include tasks such as autoregressive forecasting, conditional latent diffusion, and latent classification. We demonstrate that contrastive learning yields a superior latent space for all tasks.

4. **We show that there is a direct correlation between the smoothness of the latent space and downstream performance**. Through further analysis of the latent space in chapter 8 and subsequent improvements to the model architecture, we demonstrate that contrastive learning outperforms a simple autoencoder. This is explained through a detailed quantitative analysis of the latent space, which shows that contrastive learning increases the space's smoothness compared to an autoencoder.

5. **We show that late fusion offers substantial benefits over early fusion, and show that integrating methods across disciplines such as self-attention and graph neural networks (GNN) can further improve performance**. Finally, chapter 9 integrates late fusion techniques into our model. We implement methods from a wide range of literature and demonstrate that a novel graph neural network (GNN) approach provides performance benefits when compared to state-of-the-art self-attention methods.

1.3 Report Structure

First, chapter 2 provides an extensive overview of the current work in this field. This includes an overview of current contrastive learning methods, contrastive learning applied to geoscience data, methods of multi-modal fusion, and how to train a decoder with contrastive learning encoders jointly.

The core project is broken into six parts. Chapter 4 explores various contrastive learning methods on ERA5 data. Chapter 5 extends contrastive learning to sparse data and explores the joint training of a decoder with a contrastive learning encoder. Chapter 6 aims to further enhance the robustness of the model by exploring various methods to make the latent space more resilient to diverse input data, thereby improving downstream performance. Chapter 7 explores additional downstream tasks, including conditional diffusion and classification, and further iterates on the decoder to improve performance. Chapter 8 analyses the latent space in detail, quantifying the improvement in smoothness that contrastive learning offers over an autoencoder. Lastly, Chapter 9 explores multimodal fusion techniques to enhance downstream performance further.

Throughout each chapter, we detail our experimental setup and compare our models against a baseline autoencoder to highlight the improvements that contrastive learning offers. Finally, we conclude our work and provide suggestions for future work in chapter 10.

Chapter 2

Literature Review

The literature review presents an overview of current research on contrastive learning in computer vision and time-series data. We also summarise the current ongoing contrastive learning research within geoscience and detail the gaps this project aims to fill.

2.1 Current Contrastive Learning Methods

2.1.1 Computer Vision

SimCLR

The first major proposal within computer vision contrastive learning was SimCLR [5]. SimCLR takes a batch of data samples and, for each, creates a positive pair (i, j) by augmenting the data sample twice. These are used to create intermediate representations h_i, h_j and projections z_i, z_j. Within the batch, SimCLR optimises the network using the NT-Xent loss, as shown in equation 2.1. This maximises the cosine similarity (denoted by sim$()$) between normalised positive projections and minimises the cosine similarity between normalised negative projections. This is computed across all positive pairs (both (i, j) and (j, i)) in a mini-batch and averaged across all $2N$ pairs. SimCLR demonstrated that utilising representations derived from this contrastive learning approach in supervised learning, combined with a simple linear classifier, achieved performance comparable to that of a ResNet-50 approach on labelled data.

$$\ell_{i,j} = -\log \frac{\exp\left(\text{sim}(z_i, z_j)/\tau\right)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp\left(\text{sim}(z_i, z_k)/\tau\right)} \tag{2.1}$$

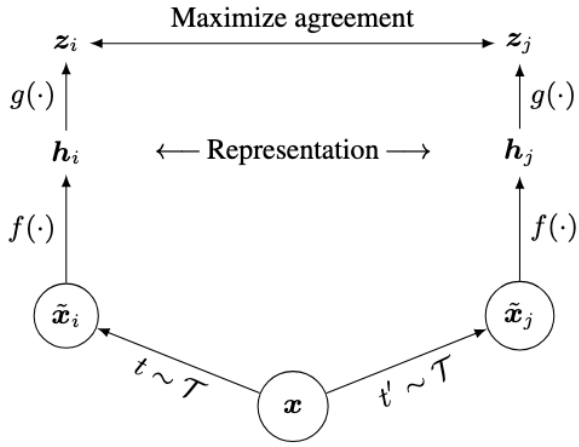


Figure 2.1: SimCLR Model

Due to the setup of SimCLR, the quality of the representations and, therefore, the downstream performance are dependent on the batch size and the quality of the data augmentations used. A larger batch size increases the number of negative samples, making the separation task harder but improving the quality of the representations. A stronger data augmentation yields a similar result, forcing the model to produce higher-quality representations, improving downstream performance.

Another consideration is the temperature hyperparameter τ introduced in the loss function; a smaller value, below 1, increases the similarity between samples. This increases the similarity of the sample to negatives, encouraging the model to push negatives further away from the sample, creating more diverse representations. Tuning the temperature hyperparameter is key, and Chen et al demonstrate that a tuned τ can help the model learn from hard negatives, that is, negatives that are closely similar to the data point. They find that a temperature parameter of 0.1 yields the best performance.

Supervised Contrastive Learning

Supervised Contrastive Learning (SupCon) [13] follows a similar approach to SimCLR, but is extended to leverage existing labels that may be present. In contrast to SimCLR, where there is only one positive sample j for a given chosen anchor sample i, labels are used to define positive samples as data points with the same label. This enables multiple positive samples for a given anchor, as shown in the updated loss function in equation 2.2, where each sample has an arbitrary number of positives p.

$$\mathcal{L}_{\text{out}}^{\text{sup}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(\mathbf{z}_i \cdot \mathbf{z}_a / \tau\right)} \tag{2.2}$$

SupCon uses the same framework as SimCLR, augmenting each sample twice and using representations and projections. They show that using labelled data with contrastive learning can lead to an increase in top-1 accuracy for ImageNet compared to SimCLR.

Barlow Twins

As mentioned above, existing contrastive methods, such as SimCLR, are dependent on larger batch sizes to improve downstream performance. Barlow Twins [14] proposes a novel approach to create

diverse feature representations that do not depend on batch size by utilising the cross-correlation matrix between representations.

As in SimCLR, and shown in figure 2.2, Barlow Twins data augmentations are applied to a sample to create a positive pair.

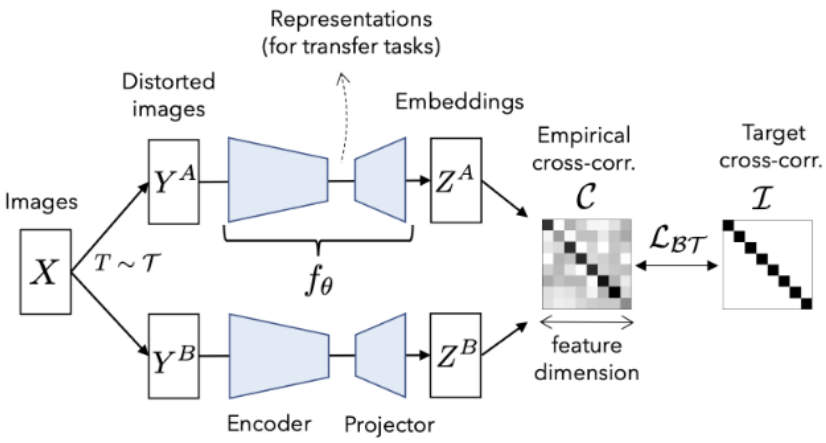


Figure 2.2: Barlow Twins Model

The projected embeddings for a batch of data are centred along the batch dimension, and the loss in equation 2.3 is computed.

$$\mathcal{L}_{\mathcal{BT}} = \underbrace{\sum_i (1 - c_{ii})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_i \sum_{j \neq i} c_{ij}^2}_{\text{redundancy reduction term}} \qquad c_{ij} = \frac{\sum_b z_{b,i}^A z_{b,j}^B}{\sqrt{\sum_b (z_{b,i}^A)^2} \sqrt{\sum_b (z_{b,j}^B)^2}} \tag{2.3}$$

The invariance term equates the diagonal elements of the cross-correlation matrix to 1, which makes the embedding invariant to the augmentations applied, by ensuring that the same dimensions of the augmented projections are the same. The redundancy reduction term attempts to equate the non-diagonal terms to zero, encouraging diversity within the dimension of the projections. In this way, the projections of the positive pairs are aligned, but collapse is prevented by encouraging non-redundancy. This involves no negative samples and therefore means that Barlow Twins is more invariant to batch size than SimCLR. Barlow Twins works well with batches as small as 256, as highlighted in figure 2.3.

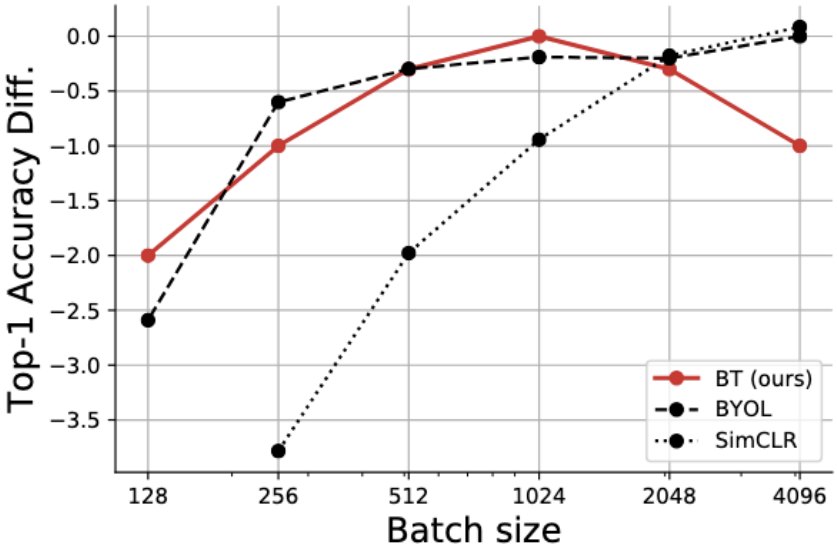


Figure 2.3: Barlow Twins Batch Size vs SimCLR

2.1.2 Time-Series Data

In addition to computer vision, contrastive learning has also been widely applied to time-series data. Eldele et al [15] propose a time-series contrastive model called Temporal and Contextual Contrasting (TS-TCC), as shown in figure 2.4. Two contrastive losses are used; the first contextual contrasting term utilises positive pairs created by augmenting a single timestep. The model also creates a context vector from t timesteps, which is then used to predict the next k timesteps. This is aligned to the context vector through the second temporal contrastive loss term. In this way, the model learns discriminative features from the contextual contrasting term and temporal features from the temporal contrastive loss.

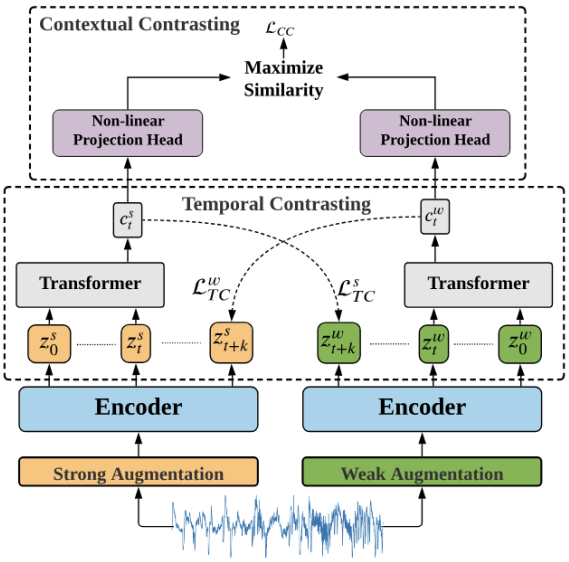


Figure 2.4: Contextual and Temporal Contrastive Loss [15]

One important consideration when comparing time-series data to image data is that timesteps close to the current selected timestep should lie closer to the latent space than timesteps further away. This data is likely to be more similar to the current timestep than to those further away. Lee et al [16] consider this by introducing soft contrastive learning. The samples are weighted according to their similarity in non-embedding space, which is precomputed to save time during network training. This similarity is then incorporated in the contrastive loss.

This concept is shown in equations 2.4 - 2.6, where a positive pair is formed by $(t, t+T)$. As the timestep increases, the resulting weight becomes smaller; points that are closer to the current timestep are assigned more weight and deemed more positive than points farther away. This also has the benefit that for a given sample, it forms a soft positive pair with every other sample.

$$w_T(t, t') = 2 \cdot \sigma(-\tau_T \cdot |t - t'|) \tag{2.4}$$

$$p_T(i, (t, t')) = \frac{\exp(r_{i,t} \circ r_{i,t'})}{\sum_{\substack{s=1 \\ s \neq t}}^{2T} \exp(r_{i,t} \circ r_{i,s})} \tag{2.5}$$

$$\ell_T^{(i,t)} = -\log p_T(i, (t, t+T)) - \sum_{\substack{s=1 \\ s \neq \{t, t+T\}}}^{2T} w_T(t, s \bmod T) \cdot \log p_T(i, (t, s)). \tag{2.6}$$

Finally, Bach et al [17] present a dynamic contrastive method that samples both adjacent timesteps as positive samples and optimises them in the same loss term, as shown in figure 2.5. In this way, there are 2 SimCLR losses, as shown in equation 2.7, combined to form the loss term. They also introduce a masking term to encourage masked projections of timesteps to be close to the projection of the unmasked timesteps through MSE loss. This enables feature invariance and stabilises the training process. Finally, they introduce an additional margin term to increase the distance between features in the projection space. They demonstrate that the inclusion of the masking term is crucial for ensuring diverse feature representations. The addition of both the masking and margin terms increases unsupervised clustering performance compared to the state of the art, while the introduction of adjacent samples enhances downstream classification.

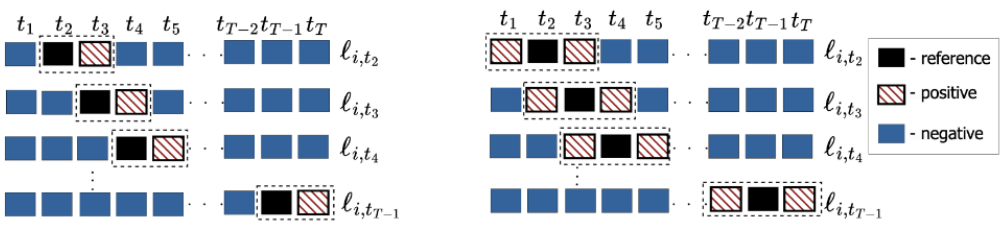


Figure 2.5: Adjacent Positive Samples

$$\ell(i,t) = -\log \frac{\exp\left(\frac{\text{sim}(z_{i,t}, z_{i,t-1})}{\tau}\right) + \exp\left(\frac{\text{sim}(z_{i,t}, z_{i,t+1})}{\tau}\right)}{\sum_{k=1}^{T} \mathbb{1}_{[k \neq t, t-1, t+1]} \exp\left(\frac{\text{sim}(z_{i,t}, z_{i,k+1})}{\tau}\right) + \sum_{l=1}^{T} \mathbb{1}_{[l \neq t, t-1]} \exp\left(\frac{\text{sim}(z_{i,t-1}, z_{i,l})}{\tau}\right)} \tag{2.7}$$

2.1.3 Multi-Modal Fusion

The contrastive learning approaches presented above focus on single modes of data, e.g. images. However, it is possible to collect data from multiple types of sensors, such as LIDAR, video, and infrared [18]. In this case, to perform contrastive learning, it is necessary to fuse the data modes. As explained by Tang et al and shown in figure 2.6, there are three ways to do this. Early fusion fuses the modes of data before passing them through the model, while late fusion passes each mode separately through the model and then fuses the resulting representations. Early fusion can be better at capturing inter-modal relationships and performs best when the relationship between modes is correlated. However, it cannot capture deep semantic relationships across modes [19]. Late fusion is better suited when overfitting is an issue and when some modalities may be missing [11], but requires a larger model and therefore may require a trade-off with the batch size. If modes are uncorrelated, late fusion can outperform early fusion. Finally, a hybrid approach involves combining the two fusion mechanisms, which can be more flexible but adds additional complexity when training the model [18].

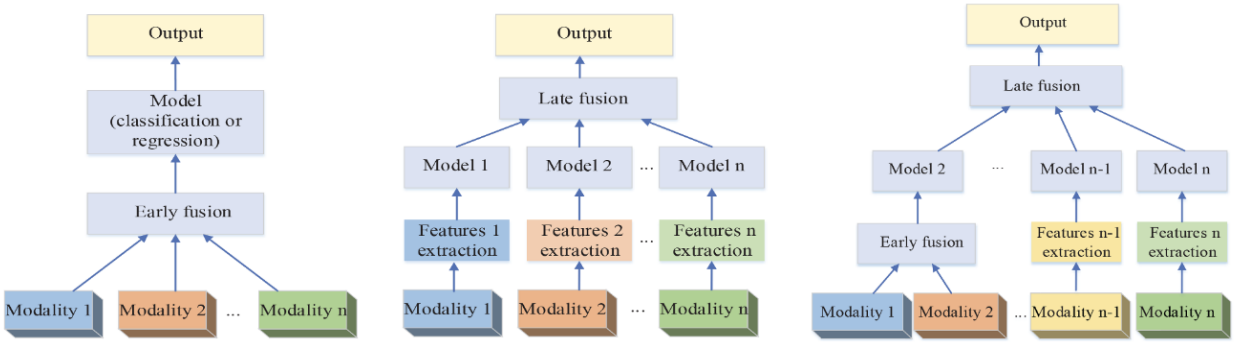


Figure 2.6: Fusion Methods

Fusion Methods

Current research on modal fusion methods primarily focuses on situations involving two modalities. Standard methods for late fusion involve concatenation (stacking) or tensor operations such as average pooling [11]. These methods are simple, but insufficient for modelling the interaction between modalities, as they treat each mode as equal, ignoring the deep semantic relationships between them [11, 19]. Attention mechanisms are the primary focus of this research area. They are more powerful than concatenation and average pooling, as they can dynamically focus on specific modes through the attention mechanism. This enables it to handle noise and uncertainties in multimodal data better. However, due to the quadratic complexity of attention, it also increases computational complexity [11].

Dufumier et al [20] utilise late-fusion modal attention to create a shared representation for two modes of data after passing them through modality-specific encoders. The representations from each encoder are passed through token adapters to serve as tokens in a self-attention block, which creates the final representation.

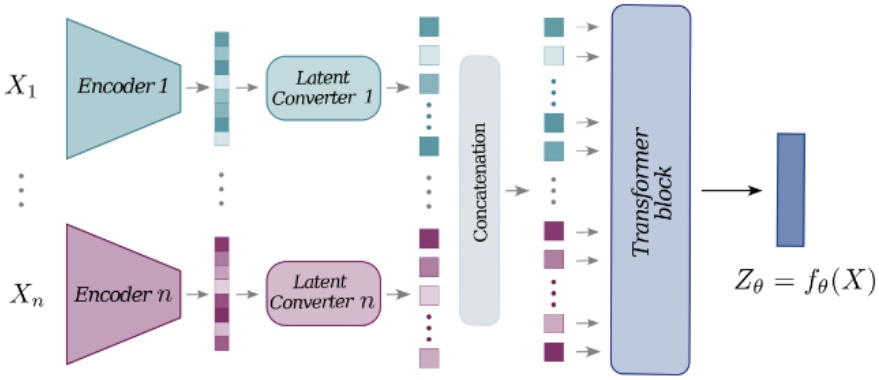


Figure 2.7: Self-Attention Fusion Mechanism [20]

Saioni et al [21] employ a similar approach using text and image data. Instead of using self-attention, they employ cross-modal attention, where the representations for one mode serve as both the values and keys for the other mode. They combine this with skip connections to form the final representation.

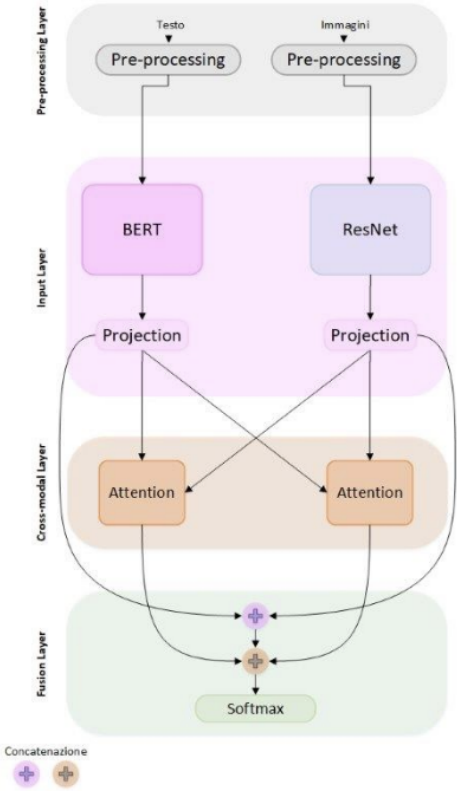


Figure 2.8: Cross Modal Attention

One downside of this approach is that it does not scale beyond two modes unless certain modal interactions are omitted. For example, if there were three modes, to model all modal interactions, $3 \times (3 - 1) = 5$ cross-attention models would be needed. Bahaduri et al [22] extend this cross-modal attention mechanism to 4 modes, but choose only one mode to act as keys and values for each mode.

One possible solution to this could extend Guo et al's [23] work, which groups modes based on modal

distances in TSNE. They utilise grouped mode representations to form a positive pair and perform contrastive learning on the other samples, similar to SimCLR.

Contrastive Learning for Multimodal Alignment

The approaches above utilise attention to fuse the latent embeddings of modes into a joint representation, which can then be used in conjunction with contrastive learning to align augmented samples. Contrastive Language-Image Pre-training (CLIP) [24], as shown in figure 2.9, on the other hand, aims to align the latent embeddings of individual modes within the same sample. To achieve this, they process each mode (text and image) through their encoders and project the resulting representations into a shared latent space, aligning the modes from the same sample using contrastive learning, while pushing away modes from different samples. In the original work, they apply this to image-text pairs, aligning the images with their corresponding captions in latent space. The resulting model can then be used in a zero-shot manner for classification tasks, requiring no additional training. Jia et al [25] and Cepeda et al [26] extended this work to larger, noisier datasets and geo-localisation, respectively.

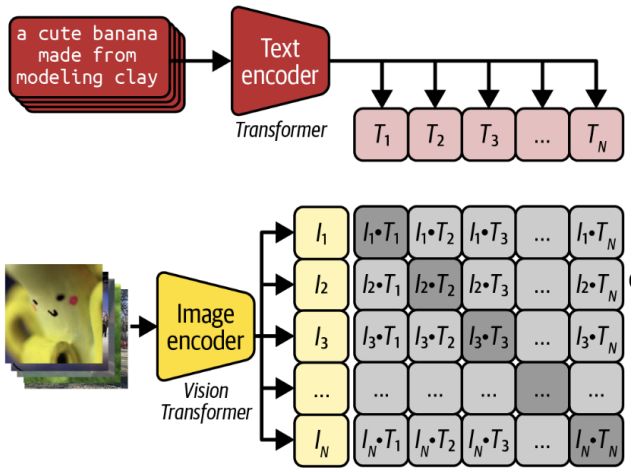


Figure 2.9: Contrastive Language-Image Pre-training (CLIP) [27][p363]

2.2 Contrastive Learning Methods in Geoscience

While contrastive learning methods within geoscience, especially the ERA5 dataset, are underexplored, some existing work exists, which is summarised below.

Wang et al [28] utilise the ERA5 dataset in a SimCLR-style model to construct latent representations for classifying weather systems in the latent space. Positive pairs are created by sampling a data point and then randomly sampling a nearby data point with a monotonically decreasing probability distribution. These two samples are augmented using resizing, random cropping, and a 5x5 mean filter. A shared ResNet-18 encoder is used with early fusion. The representations are evaluated on the classification of weather systems, showing that the contrastive learning approach outperforms baseline methods such as k-means and self-organising maps.

Wang et al also use a contrastive learning method on a range of datasets, including ERA5 data in the downstream task of tropical cyclone prediction [29]. They opt for a hybrid fusion approach using a separate encoder for each modality, but also complementing the representations with a shared

encoder. All representations are concatenated before being used in the downstream task. Instead of optimising a general contrastive loss function, they optimise the model directly on the downstream task of tropical cyclone prediction. Whilst this tailors the model to the specific downstream task, it is unlikely to build generic representations that could be used for a multitude of tasks.

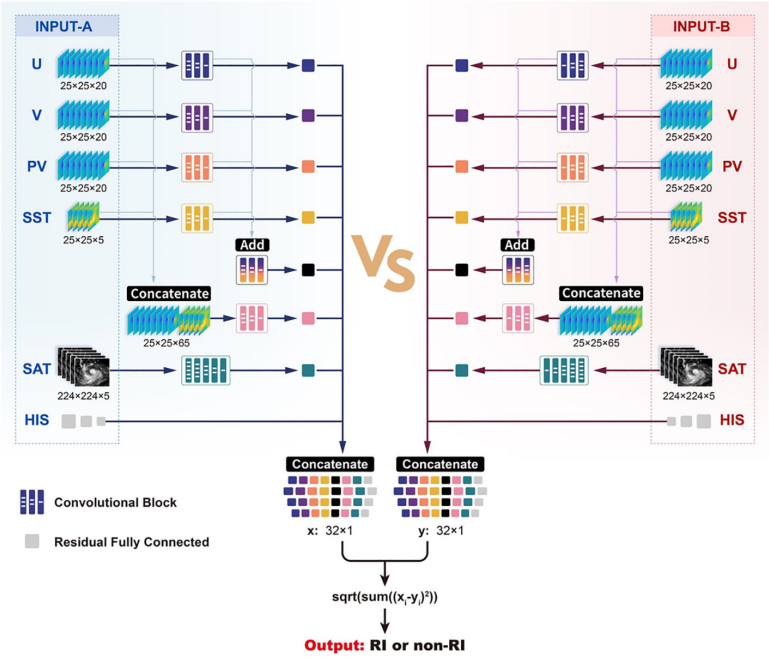


Figure 2.10: Contrastive Learning for Cyclone Prediction

Ballard et al opt to use contrastive learning in the task of climate model bias correction [30]. This refers to a set of techniques used to adjust the output of climate models so they better match observed historical data. Their model uses a super-resolution contrastive unpaired translation GAN to perform bias correction and spatial resolution enhancement. This model introduces an InfoNCE-style contrastive loss that operates on patch-level features from output images. It samples patches of the output image and ensures that the samples are similar to corresponding patches from the input. At the same time, it discourages patches from being too similar to other patches of the input image.

Shi et al [31] employ a multi-group multi-attention (MGMA) encoder-decoder model to predict atmospheric variables (AVP) over time using ERA5 data. Whilst this is built on a self-supervised learning idea and their model compresses to a latent space, neither of these ideas are used in downstream tasks. The model performs the spatiotemporal prediction in the original space, and the latent space is not explored for downstream tasks.

Finally, Han et al create a compressed version of ERA5, called CRA5 [32]. A transformer-based dual variational autoencoder (VAE) is used to compress ERA5 data to a latent space. The goal of this paper was to create a compressed version of ERA5, thereby widening the availability of weather data. The full ERA5 dataset can reach up to 226 TB, which limits its availability due to the large storage overhead. The downstream task of weather forecasting was used to evaluate the quality of the compressed data, showing that the compression had a negligible impact on training a forecasting model. While this method does not include contrastive learning, it demonstrates that building a latent space for weather datasets can yield performance comparable to that of using the full dimensionality.

2.3 Current Methods Tackling Sparsity in Geoscience

Existing work focusing on sparse observations in geoscience is limited, and to the best of our knowledge, current literature only focuses on training autoencoder-style approaches to reconstruct masked observations.

For example, Goh et al propose MAESSTRO [33] and Agabin et al propose a similar model [12], both of which use masked autoencoders to reconstruct sea surface temperature (SST) data using the LLC4320 dataset. Masked autoencoders were originally presented by He et al [34] and involve masking out patches of an image before encoding them, and training a decoder to reconstruct the masked patches. Similar to contrastive methods outlined above, this enables the encoder to form a robust representation of the data, which can then be used in downstream tasks such as classification or semantic segmentation.

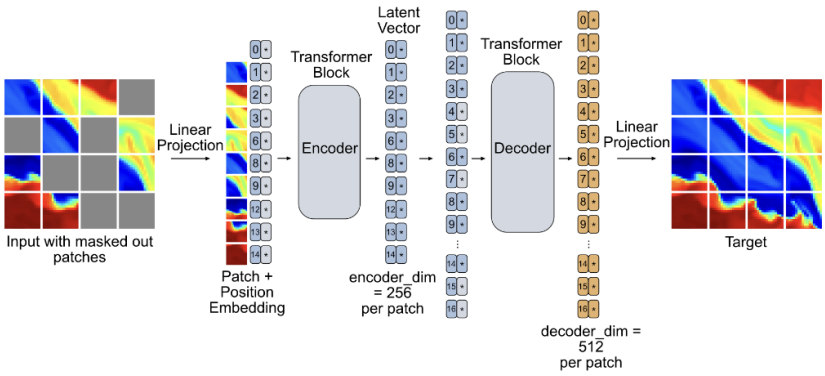


Figure 2.11: MAE Approach for SST data [12]

The MAE models used by both Goh and Agabin focus on developing a model that can reconstruct sparse samples. This is useful as it mitigates against issues that arise if sparse climate data is sampled. However, it does not evaluate the approach for downstream latent space tasks, only for reconstruction.

Vandal et al [35] extend this MAE approach to multi-modal data. They create EarthNet, a multimodal masked autoencoder model, to reconstruct sparse data across multiple modalities. EarthNet is trained as an alternative to data assimilation, which merges observations to produce an optimal initial state of the atmosphere. EarthNet accepts multiple sparse modes, encodes them into tokens, which are then utilised within an attention mechanism to form representations for the mode-specific decoders to reconstruct the modes. Like Goh and Agabin, this approach is not evaluated for the downstream task of weather forecasting in the latent space.

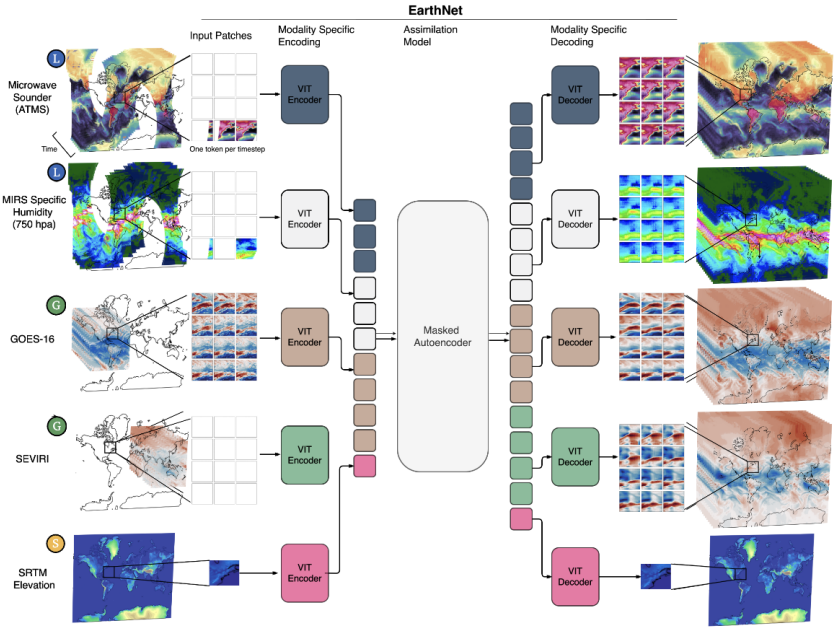


Figure 2.12: Multi-Model MAE - EarthNet

2.4 Decoder Based Contrastive Learning

While contrastive learning has been key to accelerating performance in downstream tasks, there has been little research on how to train a decoder to map latent representations back to the original space. This is likely because the majority of downstream tasks in the field of computer vision do not require the resulting data to be decoded; for example, classification tasks can be performed fully in the latent space. However, for the downstream task of weather forecasting, resultant latent space predictions must be decoded back to the original dimensionality to be usable.

Kadeethum et al [36] investigate this and create a unified framework for a Barlow Twins model in the domain of flow and transport phenomena (see figure 2.13). They propose a joint training procedure, where the encoder is trained with contrastive learning using an outer loop, and then jointly trained with the decoder in the inner loop. The outer loop utilises a batch size of 512 to leverage the fact that contrastive learning prefers larger batch sizes, while the inner loop employs a batch size of 32. The decoder is trained to reconstruct the non-augmented data sample separately, whilst augmented inputs are used in the contrastive loss as before. They find that by employing Barlow Twins self-supervised learning, the model maximises the information content within the latent space, leading to more robust and generalizable representations.

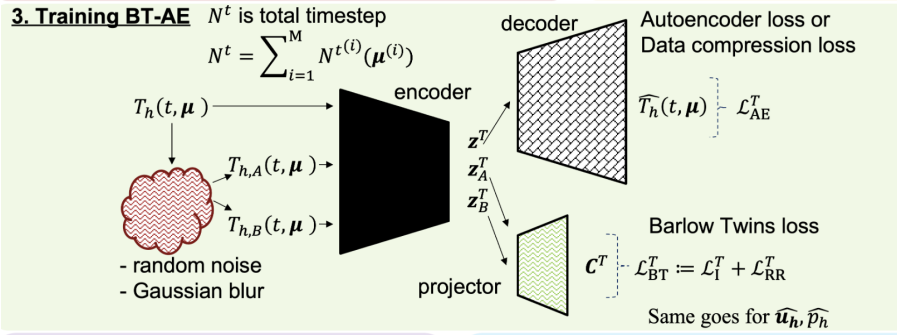


Figure 2.13: Barlow Twins Decoder

Kadeethum et al [37] also extend upon this idea by using an ensemble of encoders, where the average of the representations is used before being passed through the encoder. The same training process is used, but they find that combining multiple encoders in an ensemble model can enhance prediction accuracy and mitigate data imbalance in multiphase flow problems.

Similar works [38, 39, 40] also investigate how to train a decoder with a contrastive learning encoder. Quetin et al [38] train a contrastive encoder along with a decoder (figure 2.14). The following joint loss function is used: $L = \alpha \times L_{enc} + (1 - \alpha) \times L_{dec}$ where alpha is slowly decayed over the training epochs. They find that training the encoder and decoder in this joint fashion enhances the representational power of the encoder and improves downstream performance compared to training the decoder on the frozen representations from the encoder.

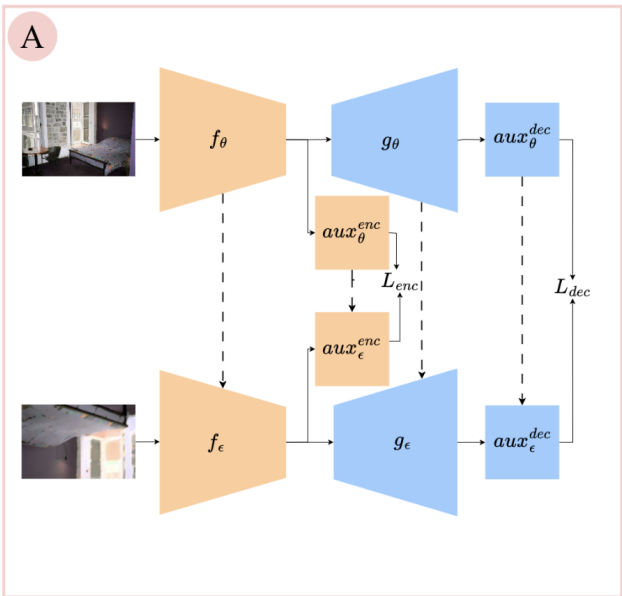


Figure 2.14: Contrastive Decoder Framework from [38]

Yao et al [39] attempt to create a contrastive masked autoencoder model optimising the following joint loss: $L = L_{con} + \alpha \times L_{dec}$. As in SimCLR, they augment the image twice to create a positive pair and then mask both samples to create a total of four samples. Contrastive loss (L_{con}) is optimised over both the masked and non-masked pairs separately, with a decoder then trained jointly to reconstruct

the masked patches of the augmented images. They find that this contrastive MAE approach yields better performance than a plain MAE or common contrastive-only approaches, such as MoCo [41].

Dippel et al [40] extend the SimCLR framework with a decoder to add a self-reconstruction loss on masked inputs, following a similar approach to [39]. The framework masks the input, and the decoder attempts to reconstruct the masked input based on the encoded representations. Like Yao et al [39], the decoder is trained to reconstruct the augmented image, which contrasts with [36], whereby the decoder is trained to reconstruct the non-augmented data point.

In addition to the encoder-decoder structure, they also present an attention-weighted pooling to replace the average global pooling typically used in SimCLR. They argue that this global operation discards salient features. They instead introduce attention-weighted pooling that computes attention weights of the feature map, which is used in global average pooling: $a(\mathbf{x}) = g(A_\phi \odot \phi) \odot \frac{1}{g(A_\phi)}$. Where A_ϕ are the attention weights and ϕ are the feature maps. They demonstrate that both attention pooling and decoder training yield improved vector representations for images, outperforming regular SimCLR on classification datasets.

Finally, Huang et al [42] use a momentum encoder to train contrastive and reconstruction losses jointly. Reconstructed images are generated by passing a masked image through an online encoder-decoder network. Positive projected embeddings are created by passing an augmented pair through the online encoder and the momentum encoder. I.e the similarity is generated via $\rho = \frac{y_s^p \cdot z_t^p}{\|y_s^p\|_2 \, \|z_t^p\|_2}$. Where y_s^p is the output from the online encoder and z_t^p is the output from the momentum encoder. These are optimised in the usual SIMCLR loss and combined with a reconstruction loss in the following formula: $L = L_r + \lambda_c L_c$, where L_c is contrastive loss and L_r is MSE reconstruction loss.

2.5 Diffusion Models

One possible downstream task that our resulting model can perform is latent diffusion, which is summarised below.

Diffusion models were first introduced by Ho et al [43]. A class of generative models, they produce new images by denoising a signal over a series of time steps [27][p208].

To achieve this, data is noised by adding a small amount of Gaussian noise over a fixed number of timesteps t. Specifically, given an image x_0, the noised image at time t is given by $\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$. Where $\bar{\alpha}_t$ is given by a diffusion schedule. A common schedule is a linear diffusion, which was used in the original work [27][p208]. A neural network, typically a U-Net, is then used to predict the noise added to the image. Given the trained network, a noisy signal is transformed into an image over a fixed number of timesteps by undoing the noise predicted by the model. We show both training and sampling methods in algorithms 1 and 2, respectively.

Algorithm 1 Training Algorithm

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t \right) \right\|^2$$

6: **until** converged

Algorithm 2 Sampling Algorithm

1: $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ **if** $t > 1$, **else** $\mathbf{z} = 0$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

2.5.1 Latent Diffusion Models

Rombach et al [44] extend upon this original work by performing diffusion in the latent space. This has computational benefits over regular diffusion models, since the operations are now performed in a space with reduced dimensionality. The data is first encoded through a variational autoencoder (VAE) to provide an embedded dataset, allowing the diffusion process to occur. During sampling, a latent code is generated and then decoded back to the full space to produce the sample.

2.5.2 Conditional Diffusion Models

Rombach et al [44] further extend their work by introducing conditional diffusion models, as shown in figure 2.15. This aims to condition the generation on a given input, such as text or representations, which can be achieved through concatenation or cross-attention. In the cross-attention mechanism, the condition is encoded to an intermediate representation to match the intermediate U-Net layers. It is used in cross-attention as keys and values, with the queries coming from the hidden states, which are inputs to the U-Net blocks. As Rombach et al show, this was a success, enabling images to be generated based on prompts.

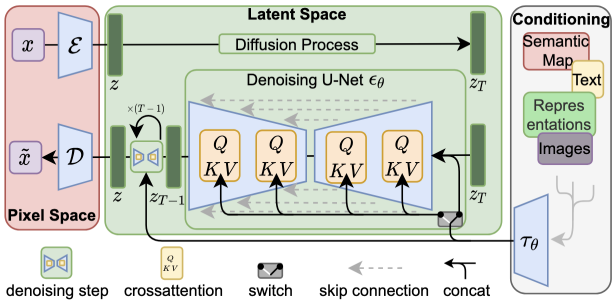


Figure 2.15: Conditional Latent Diffusion Model [44]

In the context of geoscience data, Zhuang et al [45] apply a conditional diffusion model to sparse observations for complete field reconstruction. To achieve this, they utilise cross-attention as described above and introduce a conditional encoding block to integrate sparse observations and their corresponding spatial positions.

2.6 Analysis of Latent Spaces

2.6.1 Smoothness

As one downstream task that will be explored is weather forecasting, it is essential to have measures that quantify and evaluate the latent space. This enables us to correlate our latent space with improvements in downstream performance. We initially hypothesise that the temporal smoothness of the latent space, i.e., how smoothly the temporal points are distributed in the latent space, will have a direct impact on downstream forecasting.

One measure used to evaluate the temporal smoothness of a latent space is proposed by Tzaban et al [46], and also used by Kim et al [47]. They measure the similarity between adjacent video frames, showing that higher values indicate a smoother space, with less local jitter. In their case, they utilised a pre-trained network to evaluate the similarity between frames; however, this approach could easily be extended to use cosine similarity or Euclidean distance to measure the distance between points in the latent space.

Xie et al [48] propose a jerk penalty, shown in equation 2.8, arguing that an autoencoder neglects temporal correlations when creating the latent space. Jerk, the derivative of acceleration, is optimised during the training of the autoencoder to directly promote smoothness, encouraging small changes in acceleration temporally throughout the latent space.

$$L^t_{\text{jerk}} = \|z(t+3\Delta t) - 3z(t+2\Delta t) + 3z(t+\Delta t) - z(t)\|_2^2\,, \tag{2.8}$$

Therefore, since jerk is aimed to be minimised here and directly promotes smoothness, we can use this as a measure of the smoothness of latent spaces.

In addition, Xie et al train a neural ODE in the latent space after training of the autoencoder. Neural ODEs were first introduced by Chen et al [49] and can be used to model the continuous-time evolution of a latent state z. The neural network in a neural ODE is used to output the derivative $\frac{dz(t)}{dt}$, the instantaneous velocity, which is then used in an ODE solver to produce $z(t_1)$ as shown in equation 2.9. It is trained to reconstruct trajectories by minimising MSE loss between the output path and the ground truth.

$$z(t_1) = z(t_0) + \int_{t_0}^{t_1} f(z(t), t; \theta)\, dt. \tag{2.9}$$

Xie et al show that when the jerk penalty is applied to the autoencoder, it enhances the accuracy and speeds up the convergence of the training process for both the autoencoder and the neural ODE.

Expanding on this, this implies that a better fitting neural ODE implies a smoother latent space, and as such, we can train neural ODEs on the latent space to evaluate their smoothness. Moreover, given a trained neural ODE, we can now estimate the speed of any point along the trajectory. The resulting output of the neural ODE can therefore be used to estimate speed, acceleration, and jerk of a latent space, which can be used to characterise its smoothness.

2.6.2 Generative Quality

As this project aims to extend the resulting model to multiple tasks, one of which will be latent diffusion, a generative downstream task. It is essential to have a metric to determine the quality of the

generated samples. Currently, two metrics have been identified for use.

Xu et al [50] propose a latent density score, shown in equation 2.10. Based on the training data $\mathcal{Z}$, the latent density of the generated sample z_g can be evaluated. The score measures the average Gaussian kernelised Euclidean distance between the generated sample and each training sample, with a higher score indicating a higher quality generated sample.

$$D(z_g, \mathcal{Z}) = \frac{1}{|\mathcal{Z}|} \sum_{z_i \in \mathcal{Z}} \exp\left(-\frac{\|z_g - z_i\|^2}{2\sigma^2}\right), \tag{2.10}$$

Kynkäänniemi et al [51] propose a realism score, as shown in equation 2.11. ϕ_g denotes the feature vector of a generated image, $\phi_r \in \Phi_r$ denotes the set of ground truth feature vectors and $\mathrm{NN}_k(\phi_r, \Phi_r)$ denotes the kth nearest neighbour of ϕ_r in Φ_r. This score measures how closely the sample aligns with the image manifold; the higher the score, the closer the generated image is to the image manifold.

$$R(\phi_g, \Phi_r) = \max_{\phi_r} \left\{ \frac{\|\phi_r - \mathrm{NN}_k(\phi_r, \Phi_r)\|_2}{\|\phi_g - \phi_r\|_2} \right\}. \tag{2.11}$$

These two measures provide quantitative assessments of generated samples in the latent space and serve as evaluation metrics where, by design, no ground truth is available for comparison. These measures will be used to evaluate the quality of the latent codes, while conventional measures, such as MSE, will be used to evaluate the quality of the data in the decoded space.

2.7 Summary and Research Area

According to the literature review presented, contrastive learning is widely used to leverage the underlying structure of data and build robust representations, thereby improving downstream performance. While applied in geoscience and weather prediction, there is scope to explore this further.

Wang et al [28] build a SimCLR model to classify weather systems in the latent space, demonstrating the strength of contrastive learning on weather datasets compared to traditional approaches. They also exhaustively evaluate various augmentation techniques using ERA5 data, demonstrating conclusively that resizing and smoothing yield the best results. However, they only consider the classification of weather systems by using clustering or a linear classifier on the latent representations. The task of weather forecasting and decoding back from the latent representations is not considered. Moreover, this task is only performed on a subset of the data's latitude and longitude, and only early fusion is considered.

Wang et al [29] consider contrastive learning on ERA5 data using a different hybrid fusion method. However, they optimise the model directly on the chosen downstream task of tropical cyclone prediction, rather than using a contrastive learning function. So, we cannot evaluate the method for other tasks.

Han et al [32] create a VAE model to compress the ERA5 dataset to a latent space. This is done to create a smaller dataset, thereby widening access to weather datasets. They demonstrate that this method can achieve comparable forecasting performance on latent data compared to full dimensionality. Whilst this is an important first step, they do not consider the benefit that contrastive learning methods could bring. Furthermore, they do not decode the resulting latent predictions back to the original dimensionality, which is important for use in real-world applications. They simply show that

using a compressed latent representation of ERA5 data does not downgrade the forecasting ability.

All of these methods consider the reanalysis dataset as-is and do not account for the fact that weather data is often sparse or noisy. Existing methods addressing sparsity within weather datasets [33, 12, 35] primarily focus on developing models that can reconstruct sparse data by training masked autoencoders. While this is an important area of focus within weather datasets, none of these methods consider creating robust representations of sparse or noisy data within the latent space for downstream tasks.

These approaches do not thoroughly explore the various multimodal fusion approaches available, which we highlighted in section 2.1.3. More importantly, they do not consider that latent representations will need to be decoded back to the original dimensionality for analysis.

Therefore, we can conclude that contrastive learning within geoscience applications such as weather forecasting has not been widely applied or explored. This project will focus on exploring this area more thoroughly, creating an end-to-end model that fuses multiple modalities to create robust latent representations. We will incorporate a decoder model into our system that can accurately decode the latent representations, enabling analysis after downstream tasks have been performed. Finally, we will also incorporate sparse and noisy data to create a robust latent representation that is more applicable to real-world data.

Chapter 3

ERA5 Dataset

The main dataset used in this project is the WeatherBench 2 Dataset. Specifically, the ERA5 dataset has been selected for initial trials and experiments. ERA5 is the fifth-generation European Centre for Medium-Range Weather Forecasting (ECMWF) reanalysis dataset, containing time-series data from 1959 onwards. A reanalysis dataset combines observations and model data to create a complete dataset [52]. It therefore provides consistent data on which to perform our experiments. Moreover, it has also been used within the early work on contrastive learning within geoscience [28, 29]. ERA5 consists of 62 data variables (modes) that span the complete longitude and latitude of the globe [10]. Specific variables such as humidity are also measured at multiple pressure levels. The maximum being 13 levels. In addition, ERA5 is available at various resolutions, allowing the initial work to utilise lower-dimensional data (64x32) before transitioning to higher-resolution data (721x1440) [10].

For our experiments, we use the following five variables: '2m_temperature', 'geopotential', 'u_component_of_wind, 'v_component_of_wind', 'specific_humidity'. All but temperature have associated pressure variables. We use the lowest-dimensional data (64x32), which spans the full range of latitude and longitude, resulting in a dataset of size $X \in \mathbb{R}^{B\times5\times64\times32}$, where each variable is stacked as a channel. This means the total dimensionality of a single data sample is 10240. We show all modes of a randomly selected data sample in figure A.1 appendix A.1.

Chapter 4

Part 1 - Exploration of Contrastive Learning Methods

In this chapter, we evaluate three different contrastive learning methods: Barlow Twins [14], SIMCLR [5] and supervised contrastive learning [13]. This chapter aims to compare and contrast the representations produced by each type of model on ERA5 data and to compare them against a baseline autoencoder. We begin this chapter with our initial experimental setup and then detail the implementation and results of each method.

4.1 Dataset Setup

The ERA5 dataset is used as detailed in section 3. All variables except temperature are measured at varying pressure levels. In relevant literature [28], it is common to select a single pressure value for these experiments. For ease of initial experiments, we select the pressure at level 0. ERA5 data is collected at 6-hour intervals; we resample the data to obtain daily values by averaging across the four samples per day. All available values are used from 1959-2022, dedicating 80% of the samples for training and 20% for validation.

We pre-process the data by computing the mean and standard deviation for each data variable across all training samples and standardise the data: $z_i^c = \frac{x_i^c - \mu}{\sigma}$

4.2 Contrastive Learning

As mentioned above, we have chosen three contrastive learning methods to evaluate: Barlow Twins, SIMCLR and Supervised Contrastive Learning. Similar to [28], we use a simple ResNet-18 encoder for all methods and sweep hyperparameters to find the best fit for our data. As the loss value often depends on the individual hyperparameters chosen, we instead evaluate each run by comparing the cosine similarity of the 1000-D embeddings of positive and negative pairs. We also examine the mean variance of the embeddings, as well as a TSNE plot of a random batch of validation data, to assess how well positive pairs cluster in the latent space. We opt to focus our evaluation on the 1000-D embeddings, rather than the projections, as these are used in the downstream task, so the quality is of higher importance than the projections. Results for each method are summarised in table 4.1.

We make no changes to the ResNet-18 network, except for altering the first layer to accept 5 input channels, which conforms to our data. The output size is maintained as is, resulting in a latent space of 1000; we chose a projection space of 128. This results in a total compression of around 10 times

compared to the original dataset space. A batch size of 128 was used for all trials.

To create positive pairs, we follow the augmentation procedure used by [28]. We sample a random timestep and augment it twice, by resizing to size 160x80 and randomly cropping to 144x72. Each channel in the resulting cropped image is then smoothed with a 5x5 average filter.

We use the Adam optimiser [53], with no weight decay and train each network for 100 epochs.

4.2.1 Barlow Twins

As per the original work [14], we design the projection network such that it uses three linear layers, each followed by batch normalisation. The final output used in the loss function is followed by a batch normalisation layer to give the embeddings a mean of 0 over the batch.

For hyperparameters, we tune learning rate and λ, choosing a learning rate of $1e^{-3}$ and λ values of 0.001, 0.01 and 0.1. We find that the model producing the best separation, both quantitatively and qualitatively, has a λ value of 0.01. This model produces positive embeddings that have a mean cosine similarity of 0.989, negative embeddings that have a mean cosine similarity of 0.354 and overall embeddings that have a mean variance of 45.6. An example PCA plot of the embeddings is shown in figure 4.1

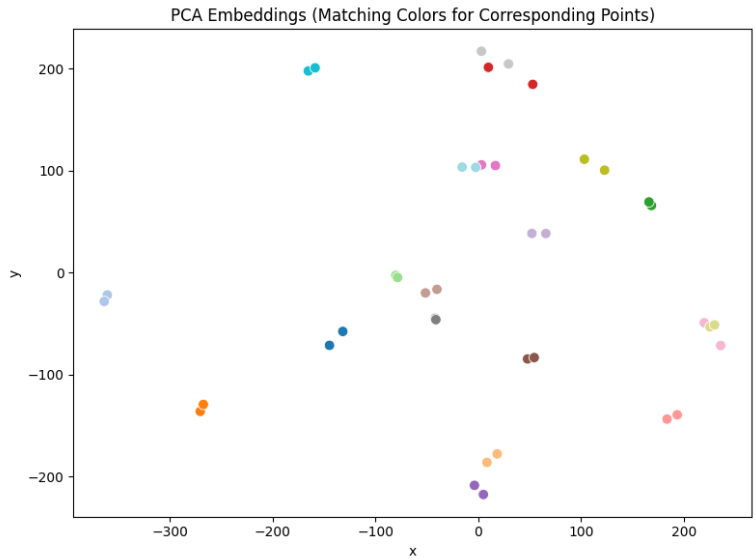


Figure 4.1: Barlow Twins PCA Embeddings

4.2.2 SimCLR

For the SimCLR model, we use the same projection network as in Barlow Twins. We sweep temperature values in the following list, using a learning rate of $1e^{-4}$:
$[0.01, 0.03, 0.05, 0.07, 0.08, 0.1, 0.3]$, finding the best temperature value to be 0.3. This produces positive embeddings that have a mean cosine similarity of 0.999, negative embeddings that have a mean cosine similarity of 0.30 and overall embeddings that have a mean variance of 0.68. An example PCA plot of the embeddings is shown in figure 4.2

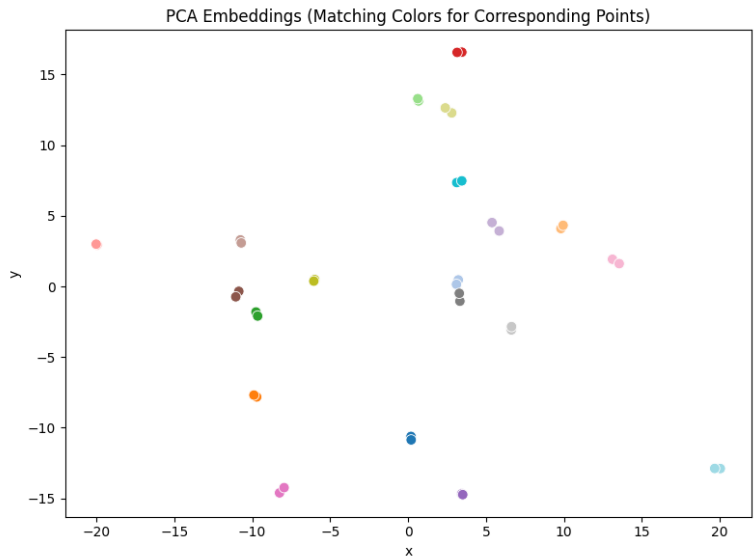


Figure 4.2: SimCLR PCA Embeddings

4.2.3 Supervised Contrastive Learning

As outlined in section 2, supervised contrastive learning utilises labels to increase the amount of positive samples for a given anchor point. To enable this for the ERA5 dataset, we assign pseudo-season labels (winter, spring, summer, autumn) for each sample depending on the month it falls into.

We use the same projection network as in SimCLR and sweep the same hyperparameters, finding the best temperature value to be 0.1. This produces positive embeddings that have a mean cosine similarity of 0.931, negative embeddings that have a mean cosine similarity of 0.23 and overall embeddings that have a mean variance of 44.24. An example PCA plot of the embeddings is shown in figure 4.3.

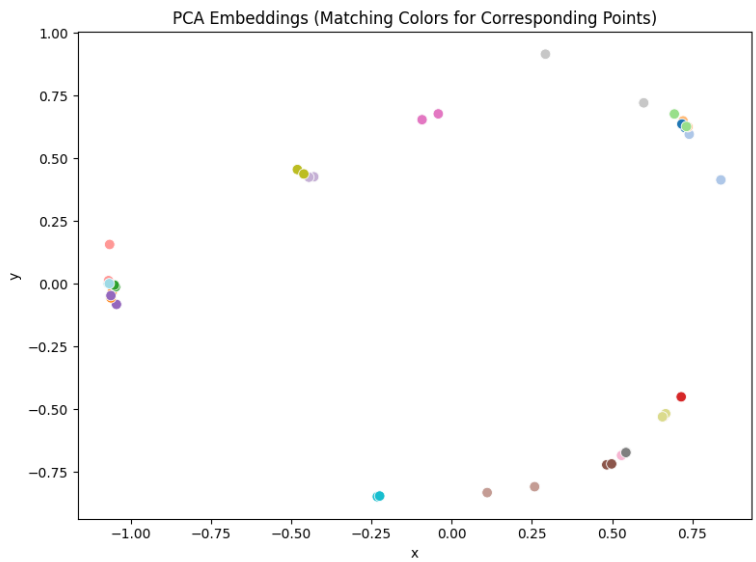


Figure 4.3: Supervised Contrastive Learning PCA Embeddings

Table 4.1: Cosine Similarity and Variance Results

Model	Mean Positive Cosine Similarity	Mean Negative Cosine Similarity	Mean Variance of Embeddings
Barlow Twins	0.989	0.354	45.6
SimCLR	0.999	0.30	0.68
Sup Con	0.931	0.23	44.24

4.3 Autoencoder

To establish a baseline for comparison with the contrastive methods, we first train a simple deterministic autoencoder. We use a ResNet-18 model as the encoder, compressing the data to a 1000-D space, and build a simple decoder to reconstruct the original data. We chose not to project down to the 128-D latent space, as done in contrastive learning methods. This is because the 1000-D contrastive learning embeddings will be used in our downstream tasks, and therefore will need to be decoded from. Hence, constructing an autoencoder in this way, removing the projection to 128-D, provides a fairer comparison to our contrastive learning methods.

The autoencoder is trained for 300 epochs, using the Adam optimiser [53] and a learning rate of $1e^{-3}$.

We show the training loss curve in figure 4.4, which highlights that the model can reconstruct the data well, with minimal loss.

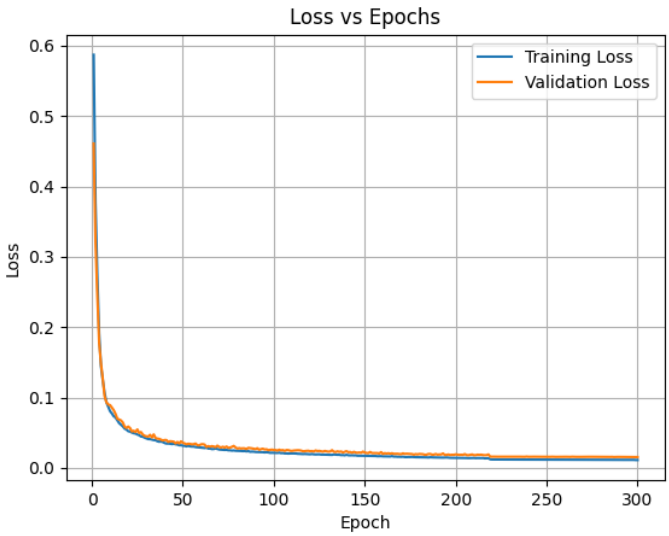


Figure 4.4: Autoencoder Loss Curve

We also show the structure of the embeddings for 20 random validation samples in figure 4.5. We can see that the samples exhibit no structure in the latent space.

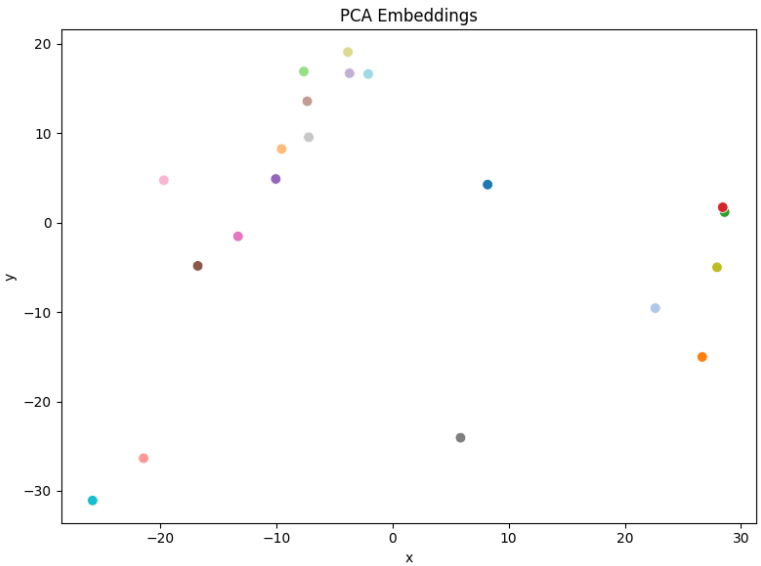


Figure 4.5: Autoencoder PCA Embeddings

4.4 Conclusions

From the evaluated contrastive learning approaches, we determine that Barlow Twins produces the best results. SimCLR produces good separation between positive and negative pairs. Supervised contrastive learning produces good separation between pairs and high variance of embeddings; however, as seen from the PCA plots, it tends to produce embeddings that collapse together in the latent space. This is expected because supervised contrastive learning has multiple positives for each sample. Since the goal of this project is to create a structured latent space, where ideally we foresee nearby timesteps to be closer to one another, we can deem that supervised contrastive learning is not as suitable for our task as SimCLR or Barlow Twins. Barlow Twins produces embeddings with the highest variance while still maintaining good separation. As mentioned in section 2, Barlow Twins is not as sensitive to the batch size and can work without explicit negative samples. Henceforth, we will use a Barlow Twins model in our future experiments.

Chapter 5

Part 2 - Designing and Training a Decoder

As discussed in our literature review, the topic of training a decoder with contrastive learning is an active field of research. In this chapter, we expand our current setup by training a Barlow Twins model on sparse data samples and by blending a decoder to reconstruct masked data into the model.

5.1 Dataset Setup

For this chapter and all subsequent chapters, we refine our dataset setup slightly to ensure sufficient samples for training and evaluating downstream tasks. We maintain the process of daily values, selecting all data from 1959 to 2023. The same five variables are used; however, in this case, we choose 60% of the samples for training, 20% for validation and the remaining 20% as test data, which will be used to train and evaluate the downstream models in latent space. To align with the relevant literature, notably [28], we select variables at 850 hPa and preprocess the data in the same manner as in chapter 4.

5.2 Introduction of Sparse Data

As mentioned in chapter 2, the current methods addressing contrastive learning in geoscience consider the ERA5 dataset as is, without accounting for sparse weather data. One of the project's goals is to utilise contrastive learning to align sparse samples with full samples, thereby improving forecasting ability in the latent space.

To facilitate this and set up a positive pair, we first follow the same procedure as in chapter 4 by randomly cropping and smoothing the image to produce two augmented views. Following this, for one augmented view, we randomly mask out a percentage of the pixels. The percentage of data in real-world observations that is sparse varies, but collected data used in ERA5 typically has large ratios of missing data [54]. Therefore, to account for this, we opt for a higher masking ratio of 70% to better simulate real-world collected data. We hypothesise that this will increase the robustness of our methods, as if full data samples can be aligned to very sparse samples, they should also align to less sparse samples, and this would not hold if posed the other way round. We show an example of a positive pair for the temperature mode in figure A.2.

We train an initial Barlow Twins model on the new sparsely augmented data with the hyperparameters found in chapter 4. We evaluate the model by passing masked positive pairs without augmentation

generated from our test dataset through our model to generate 1000-D embeddings.

The model produces positive embeddings with a mean cosine similarity of 0.996, negative embeddings with a mean cosine similarity of 0.307, and overall embeddings with a mean variance of 2.34. We show a PCA plot of positive pairs of 70% masked data and complete data in figure 5.1. From the figure, we can see that the masked data is aligned in the latent space with the non-masked data, and data points from different samples are well separated, as confirmed by the cosine similarity figures above.

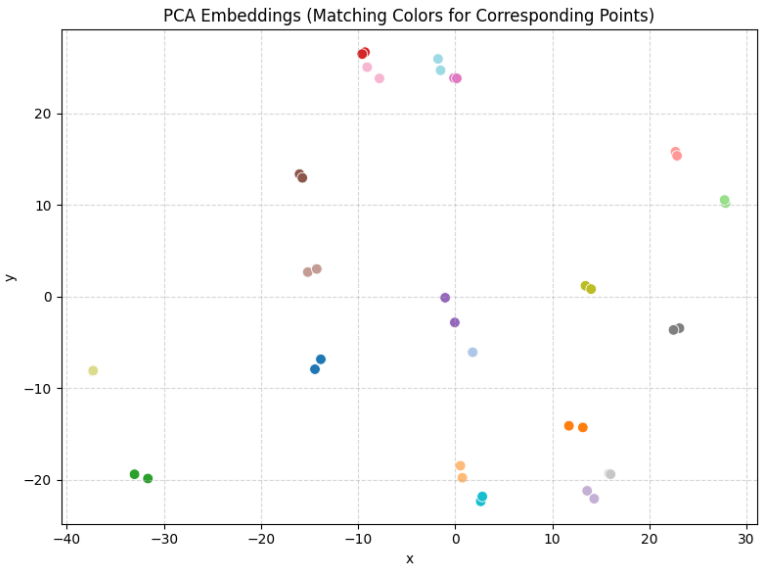


Figure 5.1: Barlow Twins PCA Embeddings - Sparse and Non-Sparse Positive Pairs

5.3 Introduction of Decoder

We next turn our attention to introducing a decoder into the model framework. As demonstrated in chapter 2, there are several existing ways to do this:

- The decoder can be trained alongside contrastive learning, blending in the two losses in a form analogous to $L_{total} = \alpha L_{contrastive} + (1 - \alpha) L_{decoder}$ [38, 39, 40].
- The decoder can be trained alongside contrastive learning in two separate loops (inner and outer). In the outer loop, the contrastive objective is optimised, and in the inner loop, the autoencoder objective is optimised [36].
- Finally, the encoder can be split into a momentum and an online encoder. With the contrastive learning objective optimised across both encoders, the decoder is used to reconstruct the embeddings of the online encoder [42].

There is also the decision of data to reconstruct, since we input augmented data in our encoder to optimise the contrastive objective. Kadeethum et al [36] opt to reconstruct the non-augmented data, whereas Yao et al [39] opt to reconstruct the augmented data.

For this chapter, we choose to adopt the first training approach, reasoning that the other two approaches may be too destructive to weights already learned by the encoder for contrastive learning.

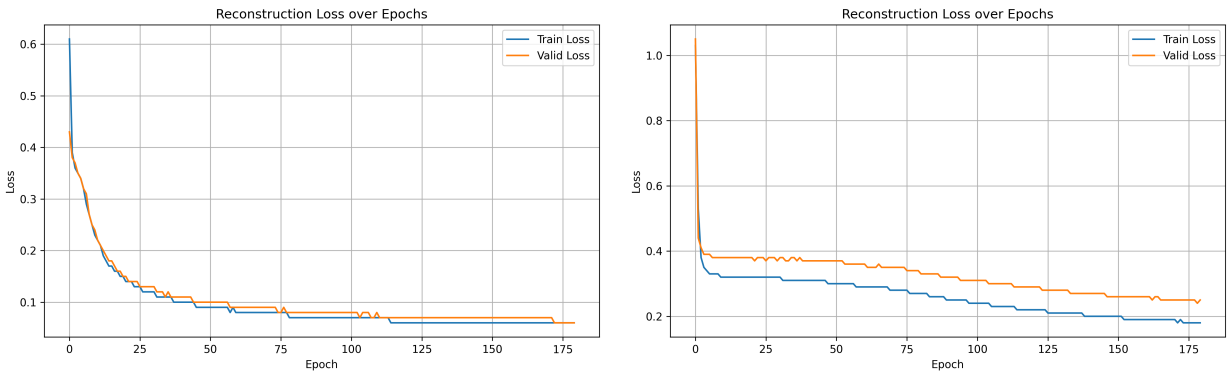
We also opt to reconstruct the non-augmented data, since this better aligns with our inference objective, where we will not be augmenting data in our final evaluation.

We also note that Quetin et al [38] first pretrain the contrastive encoder without any decoder and subsequently decay the value of α from 0.5 to 0.25 during training to enable the autoencoder objective to blend in with the contrastive objective.

Therefore, our final training pipeline will consist of the following two stages. We show a diagram of our model in figure B.1 and the same hyperparameters as in previous sections:

1. Pretraining the encoder on the contrastive objective for 100 Epochs.
2. Blending in the autoencoder objective by decaying alpha in our loss term. We start (α_{start}) at 1.0 and use an exponential decay to decrease the value: $\alpha(\text{epoch}) = \alpha_{\text{start}} \cdot e^{-k \cdot \text{epoch}}$, setting k to 0.01. We train this for 180 epochs with a learning rate of $1e^{-3}$

Figure 5.2 shows the reconstruction loss curve of the Barlow Twins, which we contrast with that of an autoencoder trained for 180 epochs with the same learning rate. We observe significant overfitting between the training and validation curves, and also note that the reconstruction loss is higher than that of the autoencoder; this issue is further explored later in this chapter.



(a) Reconstruction Losses - Autoencoder

(b) Reconstruction Losses - Barlow Twins

Figure 5.2: Reconstruction Losses - Autoencoder vs Barlow Twins

5.4 Evaluating on Forecasting Downstream Task

We next turn our attention to evaluating the trained model in a downstream task, starting with forecasting. We construct a simple sequence-to-sequence (seq2seq) encoder-decoder LSTM model in the latent space, as shown in figure B.3. We encode a set of embedded timesteps, with the decoder of the seq2seq model predicting the latent embeddings of the future timesteps. To produce the downstream embeddings, we freeze the trained encoder and move it into evaluation mode.

This provides us with an external basis for evaluating our model, using the loss in the latent space as an initial indication of model performance. Given the test data, we again further split this into 60% for training the downstream model, 20% for validation and the remaining 20% for testing. To start, for simplicity, we choose to predict the next timestep given the previous 30 timesteps. We denote the 30-timestep input as the context window and the single-timestep output as the target window. We

construct a dataset to extract these context and target windows, and do so with a stride of 1. The downstream model is trained with a batch size of 64 and a learning rate of $1e^{-2}$.

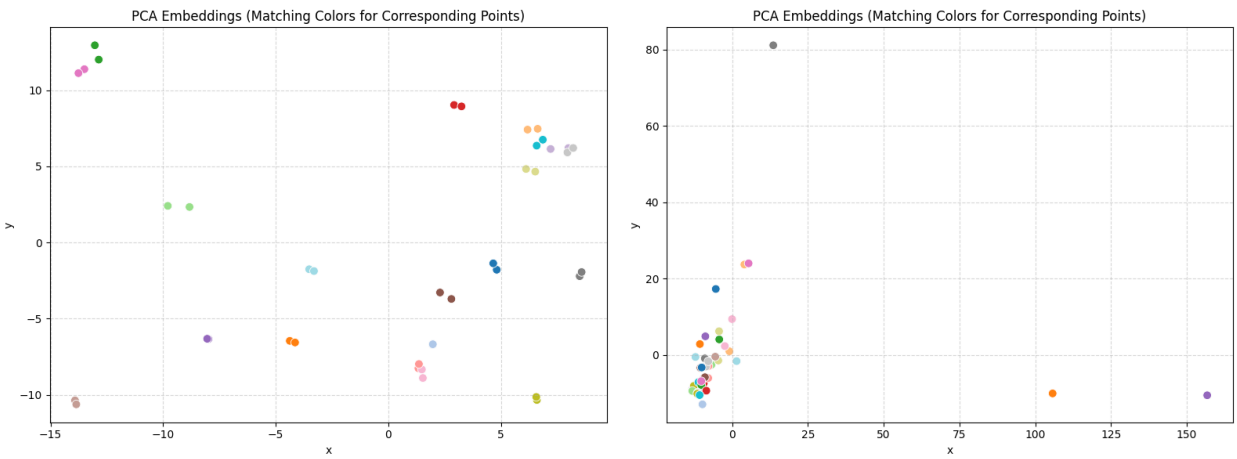
Based on the results in table 5.1, we can see that contrastive learning performs worse on the downstream task compared to the autoencoder, which is explored more in the following section.

Table 5.1: Barlow Twins vs Autencoder - Downstream Latent Forecasting

	Autoencoder	**Barlow Twins**
Downstream Performance (MSE Loss - Latent Space)	0.446	0.651

5.5 Investigation into Batch Normalisation

As shown in the results above, our contrastive learning method fails to achieve a lower latent loss on the downstream forecasting task. One interesting finding is that moving the contrastive model into evaluation mode negatively harms the produced embeddings. We can see this effect in figure 5.3, where the embeddings are dramatically distorted by switching model modes. This is further confirmed by the loss curves above in figure 5.2, which show that the move to evaluation mode produces vastly different embeddings, thereby impacting reconstruction.
We conclude that this effect is caused by the changing nature of batch normalisation (BN) layers. When moving batch normalisation layers to evaluation mode, the layer normalises the features using a running mean and variance, rather than batch statistics [55]. As such, for a given input, this changes the value of the embeddings. Since the contrastive objective was optimised for normalised embeddings using batch statistics, this produces embeddings that no longer exhibit the same qualities as in training mode. This is consistent with findings in the literature. Richemond et al. [56] show that Bootstrap Your Own Latent (BYOL), an alternative contrastive learning method, relies on batch statistics, facilitated by the batch normalisation (BN) layers, to produce high-quality embeddings. They replace these layers with group normalisation (GN) and use weight standardisation, showing that this removes the reliance on batch statistics and produces performance comparable to the original model.



(a) Barlow Twins PCA Embeddings - Train Mode

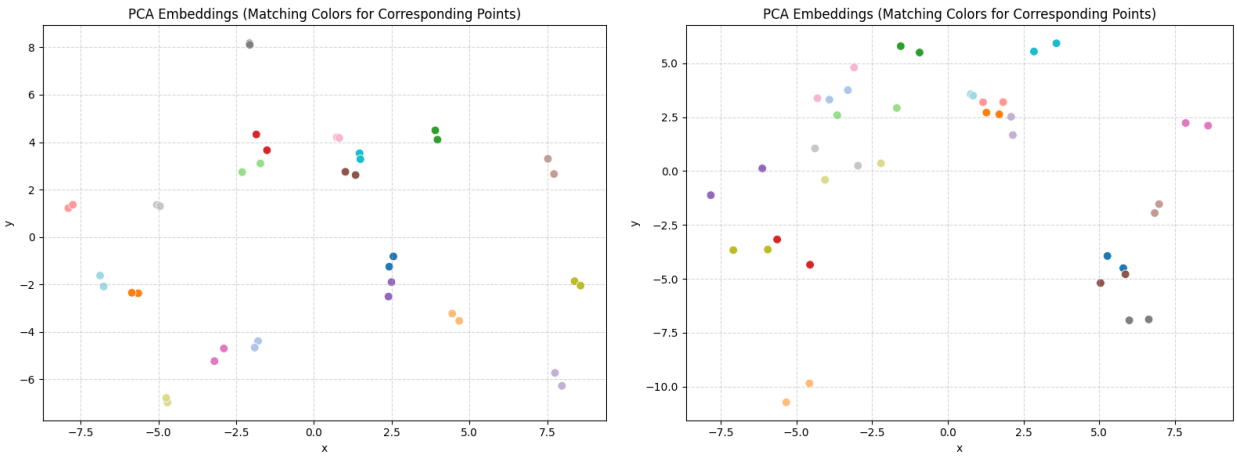
(b) Barlow Twins PCA Embeddings - Evaluation Mode

Figure 5.3: The effect on latent embeddings of moving from training to evaluation model

We explore this, replacing all batch normalisation layers, except the first layer in the ResNet-18 model, with group normalisation layers. Group normalisation layers normalise within each channel of the fea-

ture map independently of the batch, thereby removing the model's reliance on batch statistics. We theorise that a single batch normalisation layer at the start of the network will provide the network with some of the positive effects of batch normalisation, whilst having group normalisation thereafter will provide batch-independent embeddings.

Figure 5.4 confirms our findings. It shows that, while there is some movement between embeddings in train and evaluation mode, masked embeddings remain close to their full counterparts, and the general structure of the points is preserved.



(a) Barlow Twins PCA Embeddings - Train Mode **(b)** Barlow Twins PCA Embeddings - Evaluation Mode

Figure 5.4: Barlow Twins Embeddings - Group Normalisation

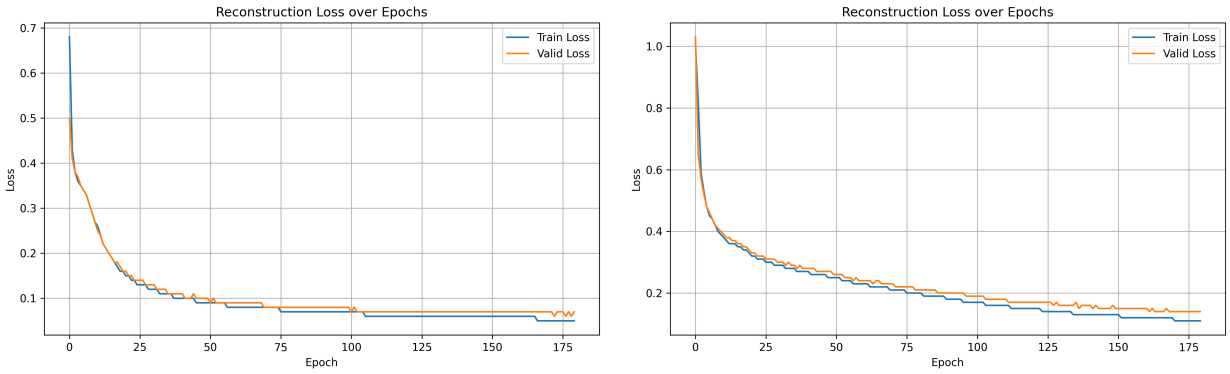
We also investigate further by comparing Barlow Twins to SIMCLR with the identical GN replacement. We train this model in the same way, using the hyperparameters found in chapter 4, and present downstream results for all models, including the autoencoder with and without GN, in table 5.2.

Table 5.2: Contrasting Downstream Performance of BN and GN

	Autoencoder - BN	**Autoencoder - GN**	**Barlow Twins - BN**	**Barlow Twins - GN**	**SIMCLR - BN**	**SIMCLR - GN**
Downstream Performance (MSE Loss - Latent Space)	0.446	**0.0341**	0.651	0.158	0.0386	**0.00524**

We can see that SIMCLR performs better on the downstream task compared to Barlow Twins, both with and without batch normalisation. One possible reason for this is that Barlow Twins explicitly enforces invariance of embeddings through its loss function. In contrast, SIMCLR implicitly enforces this by pushing negative embeddings away in the latent space. In any case, the results show the harmful impact of batch normalisation. Resulting embeddings become reliant on the features throughout the network being normalised with batch statistics. Importantly, our most performant model, SIMCLR with GN, significantly outperforms the autoencoder approach, underscoring the benefits that contrastive learning brings to our downstream task.

Figure 5.5 shows the reconstruction loss of our SIMCLR with GN model compared to the autoencoder with GN. While the losses still do not match those produced by the autoencoder, we see a significant improvement in the SIMCLR model compared to the Barlow Twins, and we choose this model to develop further in future chapters.



(a) Reconstruction Losses - Autoencoder with GN

(b) Reconstruction Losses - SIMCLR with GN

Figure 5.5: Reconstruction Losses - Autoencoder vs SIMCLR

5.6 Conclusions

To conclude, this chapter has extended our contrastive learning model to a complete end-to-end system, training a decoder on embeddings generated by contrastive learning. We further extended this to use the embeddings in the downstream task of forecasting to evaluate our solution, showing the negative impact that batch normalisation has on the embeddings and subsequent downstream performance. Through the use of group normalisation, we solved the issue and produced a SimCLR contrastive model that outperforms an autoencoder for downstream forecasting. The next chapter will incorporate concepts from current literature to improve the robustness of our solution to noisy and sparse data.

Chapter 6

Part 3 - Investigation into Robustness

Chapter 5 extended the contrastive learning model to handle sparse data and jointly blended a decoder into the training process. We investigated and demonstrated how batch normalisation with contrastive learning can harm the embeddings used in downstream tasks and showed that group normalisation provides batch size independence, leading to improved downstream performance. This chapter focuses on evaluating the model in an end-to-end fashion, using RRMSE loss with the decoded forecasted predictions as the primary metric for model evaluation. Based on these results, we propose further enhancements to the contrastive learning model, drawing on existing literature, to further boost performance.

6.1 End-to-end evaluation of the Downstream Task

Whilst the latent MSE loss metric used in chapter 5 is a good indicator of performance, it cannot be used reliably to compare and contrast models. This is because models produce different embeddings for samples, and therefore, the ground truth across models is dynamically changing. A better indication of performance is the error between the decoded prediction and the ground truth, which ensures a stable ground truth across models.

To facilitate this new evaluation, we add a training step for the contrastive learning models after the blending stage, whereby we freeze the encoder and train the decoder using masked data only for 200 epochs. This finalises our training process and enables the parameters in the decoder to settle into the fixed embeddings produced by the encoder. Our final training procedure is summarised below and used to train our contrastive learning approaches in this chapter and beyond.

1. Pretraining the encoder on the contrastive objective for 100 Epochs, using a learning rate of $1e^{-4}$, a temperature value of 0.3 and a batch size of 128.

2. Blending in the autoencoder objective by decaying alpha in our loss term. We start (α_{start}) at 1.0 and use an exponential decay to decrease the value: $\alpha(\text{epoch}) = \alpha_{\text{start}} \cdot e^{-k \cdot \text{epoch}}$, setting k to 0.01. We train this for 180 epochs with a learning rate of $1e^{-4}$ and a batch size of 128.

3. Training the downstream LSTM model for 100 epochs on our test data, using a batch size of 64 with a learning rate of $1e^{-2}$ on masked data of 70%.

4. Freezing the encoder and training the decoder on the frozen embeddings to reconstruct masked data.

6.1.1 Model Evaluation

To evaluate the models, we decode the predictions from the latent space of our downstream test dataset and compute the relative root mean square error (RRMSE). RRMSE is a unitless percentage error, allowing us to compare models easily. It is given by dividing the root mean squared error (RMSE) by the mean of the data [57]. From [58], performance is excellent if RRMSE is $< 10\%$, good if $< 20\%$, fair if $> 20\%$ and $< 30\%$ and poor if $> 30\%$.

This is shown diagrammatically in figure B.4. For simplicity, we only consider the next-step prediction as our performance indicator. Autoregressive performance for future time steps is explored in the next chapter.

In addition to the RRMSE loss, we also consider two further metrics: the Structural Similarity Index Measure (SSIM) and the Peak Signal-to-Noise Ratio (PSNR). SSIM computes the metric shown in equation 6.2 [59]. Compared to RRMSE, SSIM is a measure of the perceived quality of data, measuring similarity rather than the raw difference between data. Compared to MSE, it is more aligned with human visual perception by considering changes in structural information, luminance, and contrast. PSNR, on the other hand (equation 6.3), is more similar to RRMSE and quantifies the ratio between the maximum possible value of the data and the MSE error introduced by reconstruction. Expressed in decibels (dB), it measures the distortion caused by reconstruction [60].

We compute all metrics on the unstandardised data. This is especially important for RRMSE, as the mean of standardised test data will be close to zero, causing a substantial increase in RRMSE if used.

While having correlation, having three metrics to evaluate our methods increases the robustness of our evaluation framework. It allows us to check both the direct error and the quality of the reconstructed data.

$$\text{RRMSE} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}}{\frac{1}{n}\sum_{i=1}^{n} y_i} \times 100 \tag{6.1}$$

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{6.2}$$

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}_I^2}{\text{MSE}}\right) \tag{6.3}$$

Furthermore, we evaluate our trained model on two sets of data. First, we mask the data to 70% as used in training and evaluate the single-step forecasting. Then, we use a variable masking ratio of $[20\%, 90\%)$, which further increases the robustness of our evaluation pipeline, determining how well our models handle masked data outside of the range for which they were trained.

6.1.2 Matérn Kernel Noise

Finally, we also introduce some random noise to the data points, as a further way to test how our models perform. For this, we use Gaussian noise with a Matérn Kernel. This provides covariance that is a function of the Euclidean distance between two points; nearby points have high covariance and thus similar noise values [61]. This is crucial for geoscience data, where nearby cells have highly correlated values; thus, the noise added to those cells should be similar. For our experiments, we add

Matérn noise to $[20\%, 90\%)$ masked data using a closed form solution with $\nu = 2.5$ shown in equation 6.4 with $\ell = 1.0$ and $\sigma = 1.0$ [62][p85].

$$k(\|x - x'\|) = \sigma^2 \left(1 + \frac{\sqrt{5}\|x - x'\|}{\ell} + \frac{5\|x - x'\|^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}\|x - x'\|}{\ell}\right) \tag{6.4}$$

Figure 6.1 contrasts a data sample with added Matérn noise compared to i.i.d. Gaussian noise. We can see from the figure how the noise added with the Matérn kernel considers the proximity of the pixels, adding similar noise values to pixels close together. In contrast, i.i.d. Gaussian noise perturbs each pixel independently.

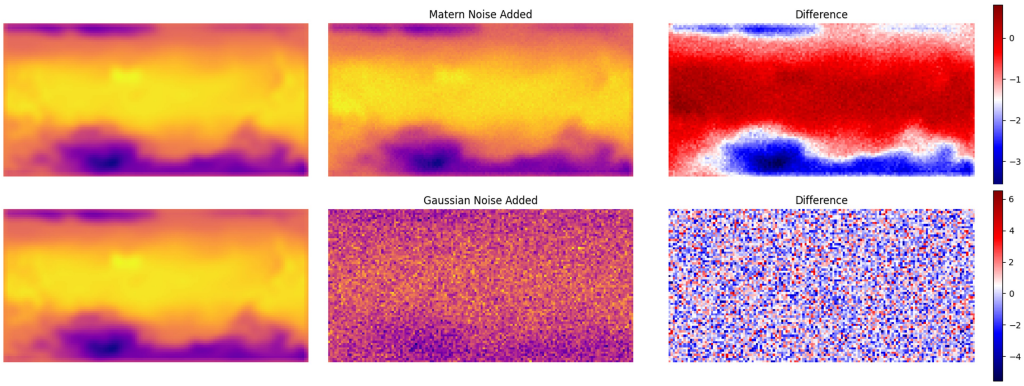


Figure 6.1: Sample with Matern Noise

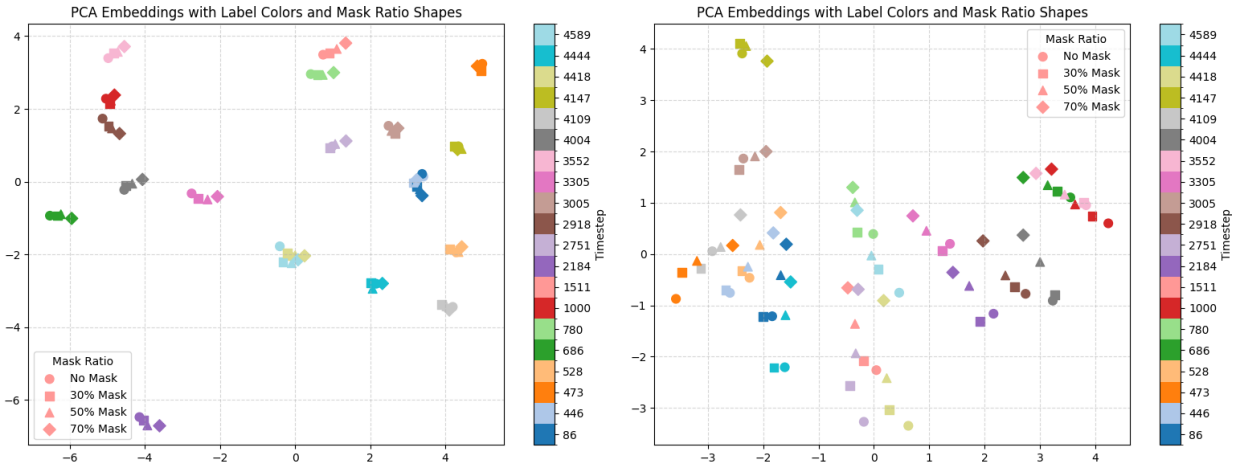
6.1.3 Initial Results

Table 6.1 shows results for SIMCLR and our comparative autoencoder. As shown, the SIMCLR model performs worse than the autoencoder. This supports our theory above, that latent loss alone is not suitable for comparing the models.

Table 6.1: Autoencoder vs SIMCLR - Downstream Decoded MSE Loss

	70% Mask			**20%-90% Mask**			**20%-90% Mask + Matern Noise**		
	RRMSE(%)	**SSIM(%)**	**PSNR**	**RRMSE(%)**	**SSIM(%)**	**PSNR**	**RRMSE (%)**	**SSIM(%)**	**PSNR**
Autoencoder - GN	6.68	78.81	26.34	6.73	78.64	26.31	8.12	76.25	25.22
SIMCLR - GN	6.90	76.37	26.12	7.36	75.43	25.81	10.57	70.36	23.69

Figure 6.2 offers some insight into this. Here, we plot various masked points from a random batch of our test data, with different ratios, against the full data point. We can see that an autoencoder produces better separation than SIMCLR, which leads to better performance, at least in the single-step prediction. In the following sections, we will explore various methods inspired by literature to enhance the contrastive learning method, thereby improving its performance.



(a) Distribution of Masked Data Points - Autoencoder **(b)** Distribution of Masked Data Points - SIMCLR

Figure 6.2: Distribution of Masked Data Points

6.2 Improving the Robustness of Contrastive Learning

6.2.1 Varied Mask Ratios

To start, one current weakness of our model is that it is only trained on masked data, with a ratio of 70%. While, in theory, aligning very sparse samples and full data samples should encourage moderately sparse samples to align as well, this is not guaranteed. A simple change that can be incorporated is to have a varying masking ratio when augmenting the second view to create the positive sample. We apply this change, choosing a random masking ratio uniformly in the range $[50\%, 90\%)$. We retrain the model, and show the resulting PCA plot in figure 6.3. We can see the positive effect this change has on embeddings, encouraging points with various masking ratios align with the full data point. As shown in table 6.2, we can see that this increases performance, decreasing overall RRMSE error, especially for varied mask data, highlighting the effectiveness of our solution. This also supports our hypothesis above, suggesting that performance discrepancies between the autoencoder and SIMCLR approach were caused by a reduction in the quality of masked embeddings compared to non-masked embeddings.

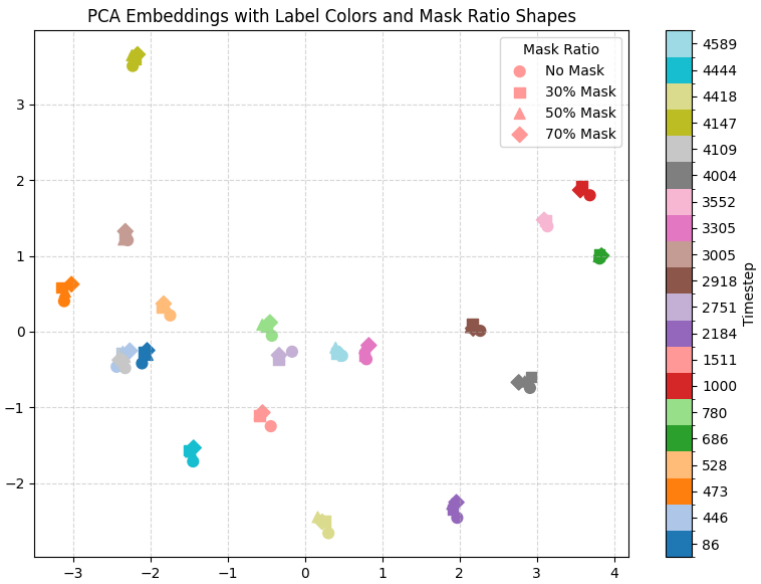


Figure 6.3: Distribution of Masked Data Points with Varied Masking Ratio

6.2.2 Temporal Structure of Data Points

We next turn our attention to the temporal nature of embeddings. Unlike image data, the points in our dataset have a temporal structure, which has a potential implicit impact on how embeddings should be laid out in embedding space. For a given sample, points closer in time would be deemed more positive than points further away, due to the similarity in the data. As mentioned in chapter 2, this has been explored in time-series contrastive learning, mainly using weighted losses and windowed approaches [16, 15]. Contrastive learning for time series data should be aware of this constraint, especially for forecasting, where we hypothesise that if points are structured temporally in latent space, it should lead to an increase in downstream performance. Therefore, we introduce two concepts to exploit this idea and improve performance.

Hard Negatives

First, we turn our attention to how samples are selected during model training. Currently, samples in a batch for SIMCLR are selected randomly. Whilst this would be adequate for image data, where all samples are equally negative, this is suboptimal for time-series data, where temporal proximity encodes meaningful structure. Therefore, we introduce explicit hard and soft negatives to the sampler during model training. Specifically, each sample (anchor) in the batch has a corresponding “hard” (within 30 timesteps of the anchor) and “soft” negative (outside 1000 timesteps of the anchor).

This approach mirrors the effect of increasing the batch size in SimCLR, which increases the number of negative samples, making the separation task more challenging and producing higher-quality embeddings. We theorise that this approach similarly increases difficulty in the temporal domain, encouraging the model to learn more nuanced temporal distinctions. By ensuring each batch contains a consistent mix of temporally structured negatives, the model learns to separate hard and soft negatives in the latent space more effectively. The model now leverages a similar set of temporally spaced samples in each batch, creating a consistent task across batches and forcing the model to learn how to separate hard and soft negatives in the latent space. In contrast, the previous sampling strategy left

the range of negative samples to chance, potentially resulting in lower-quality embeddings.

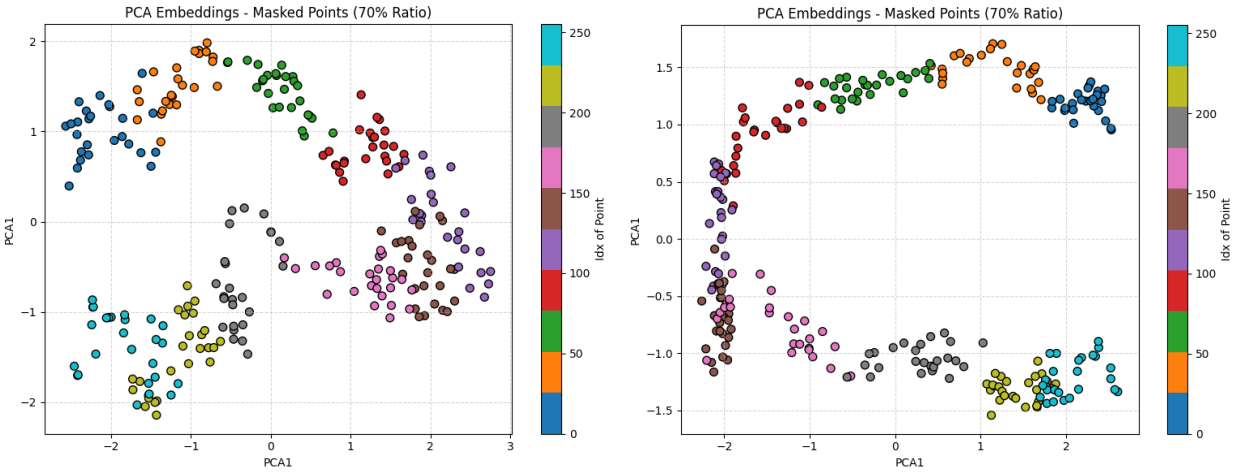
We show the result of adding this approach to the baseline SIMCLR model in table 6.2, showing increased performance over the original sampling approach and supporting our hypothesis.

Windowed Data

We next investigate the creation of the positive pairs for SIMCLR. Currently, we select an individual timestep and augment this twice, introducing sparsity in one of the augmentations. While this provides a reasonable starting point, it fails to fully leverage the temporal structure of the data, specifically that samples closer in time to a given point are inherently more similar (and thus more positive) than those farther apart. Therefore, to construct a positive pair, we should select a point close in time to the current point. This is consistent with the literature. For example, Lee et al [16] select a positive pair by sampling points at time t and time $t + T$. Bach et al [17] follow a similar approach by selecting a positive pair as $(t, t+1)$. Specifically, for the ERA5 dataset, Wang et al [28] construct a positive pair by randomly sampling a point in the range $[0, T]$ with a monotonically decreasing probability distribution.

We follow a similar approach by randomly selecting a positive sample from the range $[1, 5]$ with a monotonically exponentially decreasing probability distribution. Specifically, our probability distribution is given by $\exp(-\text{decay} \cdot t)$, where decay is set to 0.1. This is in line with [28], which uses hourly ERA5 data within a range of $[0, 180]$, corresponding to roughly 7 days. We choose a slightly smaller range and exclude 0 to ensure that we always select a point in the future, thereby fully exploiting the temporal nature of points.

Interestingly, we show that this approach does achieve the desired results, showing improved smoothness in the latent space in figure 6.4 for a batch of data in our test dataset. However, we observe that it does not lead to improved performance.



(a) Latent Space - SIMCLR, Original Approach

(b) Latent Space - SIMCLR, Windowed Approach

Figure 6.4: Smoothness of Latent Space - Windowed vs Original Approach

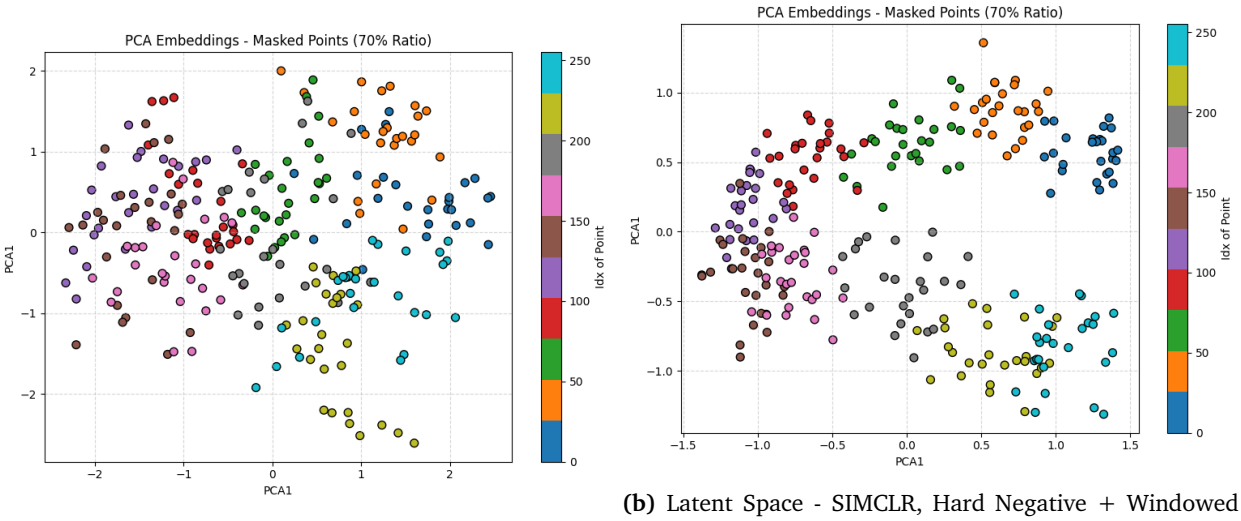
6.2.3 Combining Approaches

In the preceding section, we have introduced additions to the original SIMCLR approach, showing how each impacted performance for single-step forecasting. This next section focuses on combining

these approaches to create a unified solution that offers improvements over the autoencoder.

Windowed and Hard Negative (W + HN)

Whilst the windowed approach introduced did not increase performance for single-timestep predictions, we combine this with the hard negative approach to see if this leads to a performance increase. As shown in table 6.3, we can see that the combination of these concepts results in increased performance, outperforming the simple autoencoder approach. Figure 6.5 provides insight into why this occurs. We have demonstrated that incorporating hard negatives enhances the model's performance by producing higher-quality embeddings. However, this leads to the destruction of the temporal structure of points. Adding the windowed approach restores this, suggesting that the windowed approach offered no improvement on its own, as the points were already well-structured temporally in the latent space.



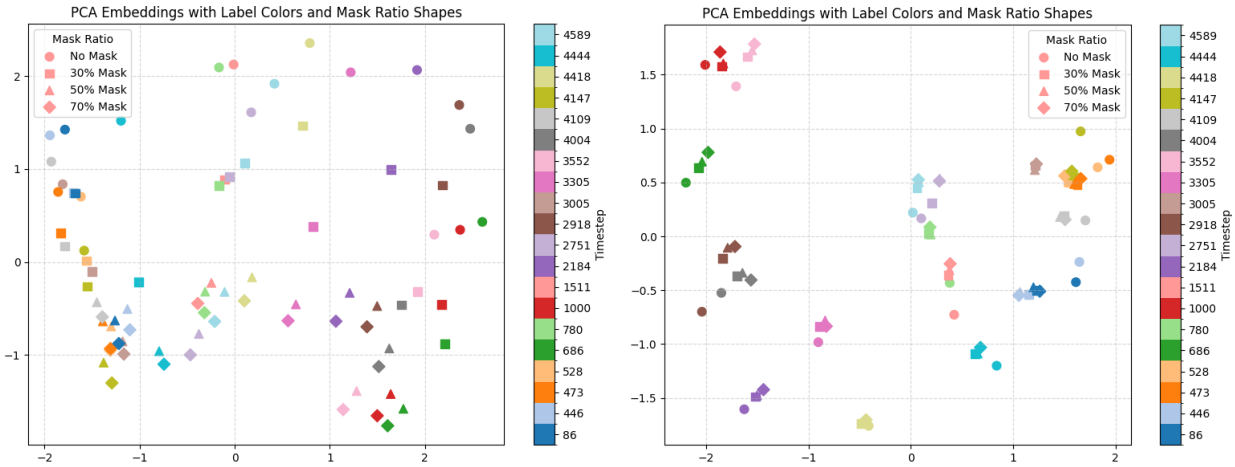
(a) Latent Space - SIMCLR, Hard Negative Approach

(b) Latent Space - SIMCLR, Hard Negative + Windowed Approach

Figure 6.5: Smoothness of Latent Space - Combining both Hard Negative + Windowed Approaches

Adding Variable Masking Ratio (W + HN + MR)

We further inspect the distribution of masked points from the model above, showing that the combination of hard negatives and the windowed positive pair approach maintains the inefficiency of masked points that we saw in figure 6.2. To amend this, we add the variable masking ratio from section 6.2.1. Figure 6.6 shows the impact of adding this on masked points in the latent space. We demonstrate similar improvements to those in section 6.2.1, resulting in increased performance, particularly for data with varied masking ratios, as shown in table 6.3.



(a) Masked Data Points - HN + W Approach

(b) Masked Data Points - HN + W + MR Approach

Figure 6.6: Masked Data Points After Adding Variable Masking Ratio

Expansion of Windowed Approach (EW + HN + MR)

Following work by Bach et al [17], we also adopt a multiple positive approach by selecting points from either side of the anchor sample. Based on the random timestep selected, we select the point at the timestep on either side of our anchor, i.e. $(t - \Delta t, t, t - \Delta t)$. This gives us 2 positive pairs $(t, t - \Delta t)$ and $(t, t + \Delta t)$. Bach et al. combine this in the loss function shown in equation 6.5. We slightly modify this, choosing to optimise the sum of two separate contrastive losses as shown in equation 6.6.

$$\ell(i,t) = -\log \frac{\exp(\text{sim}(z_{i,t}, z_{i,t-1})/\tau) + \exp(\text{sim}(z_{i,t}, z_{i,t+1})/\tau)}{\sum_{k=1}^{T} \mathbf{1}_{[k \neq t, t-1, t+1]} \exp(\text{sim}(z_{i,t}, z_{i,k})/\tau) + \sum_{l=1}^{T} \mathbf{1}_{[l \neq t, t-1]} \exp(\text{sim}(z_{i,t-1}, z_{i,l})/\tau)} \tag{6.5}$$

$$\ell(t, \Delta t) = -\log \frac{\exp(\text{sim}(z_t, z_{t-\Delta t})/\tau)}{\sum_{t=1}^{2T} \mathbf{1}_{[k \neq t]} \exp(\text{sim}(z_t, z_k)/\tau)} + \frac{\exp(\text{sim}(z_t, z_{t+\Delta t})/\tau)}{\sum_{t=1}^{2T} \mathbf{1}_{[k \neq t]} \exp(\text{sim}(z_t, z_k)/\tau)} \tag{6.6}$$

We can see from table 6.3 that expanding the windowed offers no improvement on single-step performance.

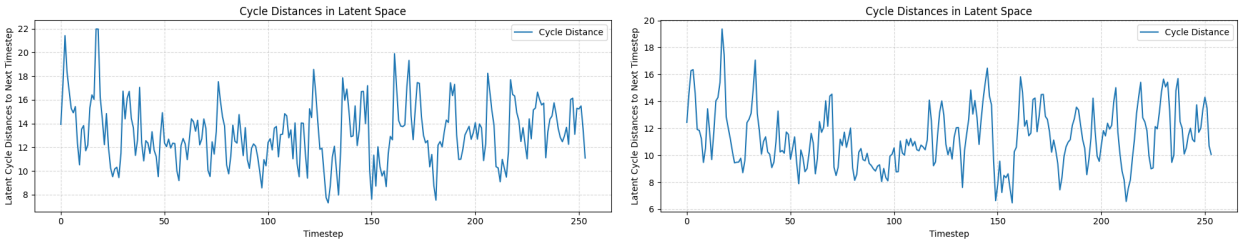
Cycle Consistency Loss (W + HN + MR + Cycle)

Finally, we refine our approach from section 6.2.3. Since we introduced a symmetric positive approach, this enables us to penalise the difference in distance between these points in the latent space. We formulate that better temporal smoothness in the latent space should boost downstream performance. Ensuring constant distance between the latent projections of points that are the same timestep apart from an anchor should lead to an improvement in temporal smoothness, resulting in enhanced downstream performance.

Inspired by the work proposed by Xie et al [48], which proposes a jerk penalty using the third-order finite difference, we adapt this idea to our positive samples by proposing a penalty to penalise the second-order central finite difference between these positive points with the anchor. Mathematically, we propose the following loss shown in equation 6.7, denoting this as a cycle consistency loss. We show the final model diagram in figure B.2.

$$\frac{1}{d}\sum_{i=1}^{d}\left[\left(Z^i_{t+\Delta}-Z^i_t\right)-\left(Z^i_t-Z^i_{t-\Delta}\right)\right]^2=\frac{1}{d}\sum_{i=1}^{d}\left[Z^i_{t+\Delta}-2Z^i_t+Z^i_{t-\Delta}\right]^2 \tag{6.7}$$

We can see from table 6.3 that the inclusion of a cycle consistency loss leads to increased performance, notably an increase in SSIM, which highlights improved quality. Moreover, we plot the mean squared error of the second-order difference between adjacent embeddings ($||z_{t+1}-2z_t+z_{t-1}||^2$) for a batch of test data and compare the results in figure 6.7. We can see that the addition of this cycle loss achieves our goal of improving temporal smoothness between latent embeddings.



(a) Squared Euclidean Distances - No Cycle Loss **(b)** Squared Euclidean Distances - With Cycle Loss

Figure 6.7: Masked Data Points After Adding Variable Masking Ratio

6.3 Results

Table 6.2: Results of Improvements, Contrasted to Autoencoder Approach

	70% Mask			20%-90% Mask			20%-90% Mask + Matern Noise		
	RRMSE(%)	SSIM(%)	PSNR	RRMSE(%)	SSIM(%)	PSNR	RRMSE (%)	SSIM(%)	PSNR
Autoencoder - GN	**6.68**	**78.91**	**26.36**	**6.75**	**78.55**	**26.29**	**8.15**	**76.25**	**25.30**
SIMCLR - GN	6.90	76.37	26.12	7.36	75.43	25.81	10.57	70.36	23.69
SIMCLR - Robust Mask Ratio	6.77	76.92	26.24	6.78	76.70	26.20	8.81	73.35	24.78
SIMCLR - Hard Negative	6.53	78.42	26.43	7.90	75.00	25.51	10.46	70.36	23.98
SIMCLR - Windowed Approach	7.11	75.93	26.06	8.62	75.02	25.55	11.25	68.70	23.52

Table 6.3: Results of Combined Improvements, Contrasted to Autoencoder Approach

	70% Mask			20%-90% Mask			20%-90% Mask + Matern Noise		
	RRMSE(%)	SSIM(%)	PSNR	RRMSE(%)	SSIM(%)	PSNR	RRMSE (%)	SSIM(%)	PSNR
Autoencoder - GN	**6.68**	**78.91**	**26.36**	**6.75**	**78.55**	**26.29**	**8.15**	**76.25**	**25.30**
Autoencoder - GN + Sampling	6.75	78.54	26.33	6.76	78.30	26.28	7.83	76.88	25.59
SIMCLR - W + HN	6.57	78.87	26.54	7.50	77.43	26.05	10.71	70.00	23.75
SIMCLR - W + HN + MR	6.47	78.99	26.59	6.48	78.70	26.56	9.15	72.88	24.67
SIMCLR - EW + HN + MR	6.52	78.98	26.55	6.51	78.71	26.50	9.30	72.32	24.62
SIMCLR - EW + HN + MR + Cycle	**6.48**	**79.64**	**26.63**	**6.48**	**79.45**	**26.59**	**9.06**	**73.62**	**24.70**

To summarise, our results show that a baseline contrastive learning model does not offer significant improvements over an autoencoder approach, at least for single-timestep forecasting. However, the addition of a carefully selected sampling scheme and cycle consistency loss enables improvements that exceed the autoencoder's performance, especially in terms of perceived quality, as shown by the SSIM. We qualitatively show the enhancement in quality in figures C.2 and C.3, highlighting that the SimCLR model produces data with smaller absolute errors and greater detail.

An interesting observation is that incorporating Matern noise into our data results in the autoencoder outperforming SimCLR. Currently, data fed into the encoder for contrastive learning solely optimises a loss in the projection space. Contrastive learning imposes an inductive bias; two augmented views of the same image are treated as positive examples. Without an explicit reconstruction objective, the encoder will only preserve features needed for the contrastive loss, ignoring noise and focusing on preserving only the shared information between views [63, 64]. In the autoencoder objective, every pixel detail will be encoded, as reconstruction is required, potentially including noise and other irrelevant details. This produces representations that are less linearly separable compared to contrastive learning [65].

Therefore, as the autoencoder preserves more detail in the latent space, it is more robust to strong perturbations like the addition of Matern noise. It is learning how to take noisy, corrupted inputs and reconstruct them. In contrast, contrastive learning learns to separate representations in the latent space, removing any details unnecessary for this purpose. We examine this concept in greater depth in chapter 7 and propose a solution to address it.

6.4 Conclusions

This chapter investigated the decoded forecasted results in a robust evaluation framework. We first highlighted the initial strong performance of the autoencoder approach. However, we demonstrated that modifications to the contrastive learning method, focusing on improved temporal alignment of points and handling varying masking ratios, yielded stronger embeddings that resulted in enhanced downstream forecasting performance. The next chapter focuses on improving the decoder to enhance results further, and more importantly, investigates autoregressive performance, predicting values across a range of future time points based on a single context window. We also investigate how our models perform on additional downstream tasks of conditional latent diffusion and latent classification.

Chapter 7

Part 4 - Additional Downstream Tasks and Improvements to the Decoder

This chapter first investigates improvements to the decoder, examining subtle ways to enhance its performance. As in all previous chapters, we compare these improvements to those of an autoencoder to demonstrate the benefits of contrastive learning. We then expand our evaluation framework to examine how our contrastive learning approach performs in autoregressive forecasting. Finally, we implement additional downstream tasks to explore how the model performs with conditional latent diffusion and latent classification.

7.1 Improvements to the Decoder

7.1.1 Specific Reconstruction Losses

To further increase performance, we plot the individual modal error in a histogram to understand which modes the model struggles with the most. We show the results for this in figure C.1. We notice the model significantly struggles to reconstruct both the horizontal and vertical wind modes. To improve this, we utilise our MSE reconstruction loss when blending in the autoencoder and expand it, as shown in Equation 7.1. Specifically, we are now computing the reconstruction loss in a mode-independent manner, which will enable us to weight the losses of each mode, encouraging the model to place more emphasis on the wind modes.

$$\text{Loss}_{\text{mse}} = \sum_{c=0}^{C-1} \mathcal{L}_{\text{mse}} \left(\hat{X}_c, X_c \right) \tag{7.1}$$

We present the performance increase of this change in table 7.1, without any weights, along with a minor modification to the decoder, which involves increasing the number of decoder blocks by one. This already offers an increase in performance, as it provides an implicitly higher weighting for MSE loss in the overall loss function, leading to improved decoding. As an aside, we also notice overfitting occurring in the downstream model, so we apply a small L2 regularisation (weight decay) coefficient of $1e^{-5}$. Each SimCLR model in table 7.1 undergoes these changes iteratively to demonstrate the benefit of each modification. The first SimCLR model applies weight decay to the downstream task, whereas the second applies decoder tweaks and weight decay. The following models in the table operate similarly, continually making compounded changes iteratively.

Weighted Losses

We next weigh the losses of the horizontal and vertical wind modes by modifying the MSE loss as such: $\text{Loss}_{\text{mse}} = \sum_{c=0}^{C-1} W_{mode} \cdot \mathcal{L}_{\text{mse}}\left(\hat{X}_c, X_c\right)$. Where W_{mode} is set to 2 for wind modes and 1 for all other modes. Surprisingly, this offers no increase in performance. We theorise that this is a limitation of the singular encoder for all modes. Despite the increased weighting of the losses, the decoder has to work with a shared representation for all modes, which limits its ability to reconstruct. Regardless, we retain the improved MSE loss in our training flow, as the implicit increase in weighting of this loss leads to performance improvement.

7.1.2 Comparison to Autoencoder

As with previous approaches, we compare our approach to a simple autoencoder. In this case, since we observed no performance gains, we train an autoencoder without the hard negative sampler, but use the modified reconstruction loss described in section 7.1.1. We use the same weight decay value of $1e^{-5}$ for the downstream model and compare contrastive learning models with all changes.

We note some inefficiencies in the training of the autoencoder, notably overfitting, so we apply L1 and L2 regularisation coefficients of $1e^{-6}$ to the training process. Unexpectedly, we see that the autoencoder with this regularisation outperforms our best SIMCLR model. However, our current evaluation framework does not fully explore the capabilities of our models; we expand on this further below, looking at autoregressive performance.

Table 7.1

	70% Mask			20%-90% Mask			20%-90% Mask + Matern Noise		
	RRMSE (%)	**SSIM(%)**	**PSNR**	**RRMSE (%)**	**SSIM(%)**	**PSNR**	**RRMSE (%)**	**SSIM(%)**	**PSNR**
Autoencoder - GN	6.31	80.24	26.67	6.33	80.08	26.66	6.97	78.32	25.95
Autoencoder - GN, L1 + L2 Decay	5.85	81.92	27.13	5.87	81.71	27.10	7.24	80.29	26.31
SIMCLR - Just Weight Decay	6.33	78.98	26.66	6.33	78.73	26.63	8.28	75.21	25.54
SIMCLR - Decoder Tweaks	5.98	81.20	27.07	5.99	80.97	27.03	8.46	76.21	25.59
SIMCLR - New MSE Loss	5.85	81.81	27.18	5.85	81.53	27.13	7.70	77.91	25.94
SIMCLR - Weighted Wind Losses	5.90	81.69	27.14	5.92	81.44	27.10	7.47	78.13	26.11

7.2 Autoregressive Performance

We now focus on evaluating autoregressive performance to provide a more comprehensive assessment of our models. We use the same evaluation protocol as before, extending the prediction horizon to 100 timesteps in an autoregressive manner, where each prediction is fed into the LSTM to generate the next one.

Table 7.2 presents the autoregressive performance of our SIMCLR model in comparison to the best autoencoder from table 7.1. We present the results for autoregressive timesteps of T = 25, T = 50, T = 75, and T = 100 (for full results of RRMSE, PSNR and SSIM, see appendix C.3). For completeness, we also evaluate the autoencoder trained with the same dataset sampling as introduced in chapter 6. Finally, to obtain the most informative results, we also evaluate the models for shorter context windows and with varying strides. Introducing strides means that there is less overlap between successive context windows, reducing the number of training examples. To counter this, for strides greater than 1, we train three downstream models using three different seeds and average the results. This enables more stable testing and allows us to compare models with more confidence, summarising these results

in tables 7.3 and 7.4.

Table 7.2

	Context Window 30, Stride 1			
	RRMSE (%) T=25	**RRMSE (%) T=50**	**RRMSE (%) T=75**	**RRMSE (%) T=100**
Autoencoder - L1 + L2 Decay	20.19	21.11	22.30	22.17
Autoencoder - L1 + L2 Decay + Sampling	11.09	11.06	11.09	11,1
SIMCLR - New MSE Loss	18.29	19.36	20.19	20.17

Table 7.3

	Context Window 5, Stride 5			
	RRMSE (%) T=25	**RRMSE (%) T=50**	**RRMSE (%) T=75**	**RRMSE (%) T=100**
Autoencoder - L1 + L2 Decay	24.69	24.48	24.48	24.75
Autoencoder - L1 + L2 Decay + Sampling	20.38	21.29	22.13	22.23
SIMCLR - New MSE Loss	22.11	22.82	22.76	22.89

Table 7.4

	Context Window 5, Stride 10			
	RRMSE (%) T=25	**RRMSE (%) T=50**	**RRMSE (%) T=75**	**RRMSE (%) T=100**
Autoencoder - L1 + L2 Decay	21.56	21.67	22.15	22.35
Autoencoder - L1 + L2 Decay + Sampling	28.35	29.87	29.71	29.71
SIMCLR - New MSE Loss	23.94	24.17	24.24	24.42

The results are revealing and highlight several contributions of this work. Firstly, we note the impact of introducing our hard-negative approach from chapter 6. This substantially improves autoregressive performance for all context windows with a stride of 1. A masked autoencoder is trained to reconstruct masked data; as such, it encourages masked data points to be close to full data points in the latent space, thereby forcing negative samples to be pushed away in the latent space, even without an explicit contrastive term. By introducing our sampling method, we add additional structure to the distribution of samples the autoencoder encounters, providing implicit regularisation, such that the autoencoder will separate both samples close and far away from anchor points.

Whilst the results show that contrastive learning can outperform an autoencoder in some conditions, it has, to date, failed to yield significant improvements in downstream forecasting performance. We address this limitation below and show conclusively the benefits that contrastive learning offers.

7.3 Further Improvements to Contrastive Learning

7.3.1 Decoding SIMCLR Embeddings

We detailed in section 6.3 how the contrastive learning and autoencoder objective functions implicitly form different representations in the latent space due to their distinct inductive biases. Although we reconstruct equivalent masked data samples through our blended training approach, without reconstructing the SimCLR embeddings, we create a duelling objective, where the autoencoder loss attempts to preserve information for reconstruction and the SimCLR loss does not. Fundamentally, autoencoder latents are designed to be generative, whereas contrastive latents are designed to be discriminative. By introducing an additional reconstruction loss for the SimCLR embeddings in the fine-tuning stage,

we hypothesise the encoder will retain richer information in its projections, thereby improving downstream forecasting performance. We also increase the number of epochs for the fine-tuning stage from 180 to 250, allowing the autoencoder to blend more into the contrastive learning approach, anticipating that this will further enhance the model's ability to decode embeddings.

We also aim to address the deficit explored in section 6.3, where we note that due to the lack of reconstruction, the model's forecasting performance was hindered by the addition of Matern noise. For the same reasons as above, the additional reconstruction term will encourage the inputs to be more robust to structured noise perturbations.

Tables 7.5, 7.6, 7.7 show the autoregressive results of adding these changes. We observe a substantial performance increase for all experiments, particularly for strided data. This confirms the first part of our theory, showcasing the efficacy of contrastive learning for autoregressive forecasting when strided data is introduced.

Table 7.8 shows that the addition of the reconstruction loss does lead to a smaller performance change when Matern noise is added to inputs, verifying the second part of our theory. However, this does not improve single-step forecasting when compared to previous models. We observe that the additional reconstruction loss presents a trade-off between autoregressive and single-step forecasting. However, due to the substantial performance increase and wider usability of a model with enhanced autoregressive performance, we continue to use this additional reconstruction loss.

Table 7.5

	Context Window 30, Stride 1			
	RRMSE (%) T=25	**RRMSE (%) T=50**	**RRMSE (%) T=75**	**RRMSE (%) T=100**
Autoencoder - L1 + L2 Decay + Sampling	11.09	11.06	11.09	11,1
SIMCLR - New MSE Loss	18.29	19.36	20.19	20.17
SIMCLR + Decoding Embeddings	**11.6**	**11.71**	**11.74**	**11.77**

Table 7.6

	Context Window 5, Stride 5			
	RRMSE (%) T=25	**RRMSE (%) T=50**	**RRMSE (%) T=75**	**RRMSE (%) T=100**
Autoencoder - L1 + L2 Decay + Sampling	20.38	21.29	22.13	22.23
SIMCLR - New MSE Loss	22.11	22.82	22.76	22.89
SIMCLR + Decoding Embeddings	**11.50**	**11.60**	**11.58**	**11.58**

Table 7.7

	Context Window 5, Stride 10			
	RRMSE (%) T=25	**RRMSE (%) T=50**	**RRMSE (%) T=75**	**RRMSE (%) T=100**
Autoencoder - L1 + L2 Decay + Sampling	28.35	29.87	29.71	29.71
SIMCLR - New MSE Loss	23.94	24.17	24.24	24.42
SIMCLR + Decoding Embeddings	**17.81**	**18.44**	**18.05**	**18.23**

Table 7.8

	70% Mask			**20%-90% Mask**			**20%-90% Mask + Matern Noise**		
	RRMSE(%)	**SSIM(%)**	**PSNR**	**RRMSE(%)**	**SSIM(%)**	**PSNR**	**RRMSE (%)**	**SSIM(%)**	**PSNR**
Autoencoder	6.31	80.24	26.67	6.33	80.08	26.66	6.97	78.32	25.95
Autoencoder - L1 + L2 Decay	5.85	81.92	27.13	5.87	81.71	27.10	7.24	80.29	26.31
SIMCLR - New MSE Loss	5.85	81.81	27.18	5.85	81.53	27.13	7.70	77.91	25.94
SIMCLR + Decoding Embeddings	7.37	76.38	26.07	7.35	76.16	26.06	8.00	74.63	25.61

7.4 Conditional Latent Diffusion Model

As explained in chapter 2, latent diffusion models operate in the latent space, generating new latent codes, whilst conditional diffusion models can be conditioned on an input to guide their denoising process, producing data more akin to the condition. We construct a conditional latent diffusion model as an additional downstream task for evaluation. We follow the work of Zhuang et al [45] and Rombach et al [44], constructing a U-Net model in the latent space, as shown in Figure B.5. In a similar vein to [45], we condition each block on a 70% masked data point in cross-attention blocks.

The original work utilised a spatial latent code, allowing the conditional vector to function as multiple tokens in the cross-attention block. However, we found that retaining the flattened 1000D vector and constructing a linear model yielded better performance. As such, the condition and the noisy input are combined into a single token of dimension 1000 for the cross-attention block. Alternatively, we could have used each dimension as a token, resulting in 1000 tokens; however, since the complexity of the attention block is quadratic, this would have significantly increased computation.

We train a conditional latent model on the encoded representations produced by our autoencoder and our best SimCLR model, both with the same data sampling process. We use the same split of training, validation and testing data as the forecasting model.

To evaluate the models, we take a random batch of test data to condition and measure the RRMSE error of the generated data, as well as the realism and latent density scores. For thoroughness, we generate a data sample 10 times and take the mean of the generated data points, showing the standard deviation in the results.

Table 7.9 presents the results, showing that the inclusion of contrastive learning leads to improved RRMSE, realism, and latent density scores. Additionally, contrastive learning produces more stable generation, with a lower standard deviation across the 10 generated batches. We also compare a conditioned generated sample in figure 7.1, highlighting the qualitative difference in generation between the autoencoder and the SIMCLR model.

As we mentioned above, contrastive learning focuses on creating embeddings that preserve the shared information between views, ignoring the augmentations applied to create these views. This leads to some features being ignored in the latent space, for example, noise or jitter. This produces a latent space that highlights the semantic structure of the data. Small perturbations do not move the latent code far from the latent code of the clean data. Furthermore, it has been shown that contrastive learning produces latent codes that are aligned and uniform on a hypersphere [66]. Uniformity means that latent codes are evenly distributed, so adding noise will not move the latent vector far from the training distribution of encoded latent vectors. Alignment means that adding noise does not move the latent vector far from its clean counterpart. In addition, the introduction of contrastive loss ensures greater separation between negative samples, meaning that perturbing a latent vector with noise will not move the vector into the realm of a vector of a different data sample.

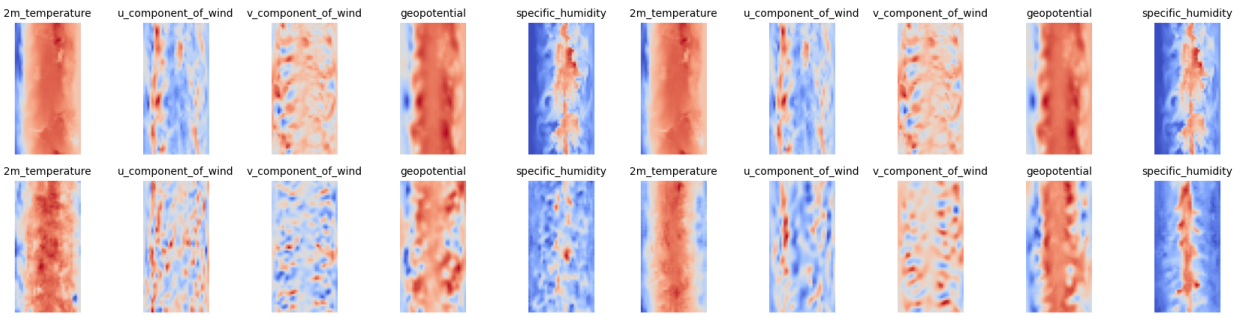
Due to these reasons, the noisy latent codes will be closer to clean samples than in the autoencoder case, meaning our model creates a latent representation that is more robust for latent diffusion. This is further supported by Park et al [67] and Nam et al [68], who apply a contrastive learning loss to diffusion and latent diffusion models, respectively. These losses maximise the mutual information between the input and output image, producing better generative results.

Note that we do not consider Matern noise as detailed in chapter 6 as a "small perturbation". Adding structured noise adds additional patterns to the data, potentially moving it visually closer to another sample. Gaussian noise, on the other hand, degrades the data, without creating a new competing one.

These findings demonstrate that combining contrastive and autoencoder objectives yields a better-structured latent space, which excels in both generative and predictive tasks. This further supports our claim that contrastive learning enhances the representational quality of the latent space compared to a simple autoencoder approach.

Table 7.9

	RRMSE (%)	SSIM (%)	PSNR	Latent Density Score	Realism Score	Std Dev - Generation
Autoencoder - L1 + L2 Decay + Sampling	37.89	19.44	12.01	0.35	0.097	3.01
SIMCLR + Decoding Embeddings	30.34	35.37	15.27	0.38	0.13	2.91



(a) Generated Sample - Autoencoder **(b)** Generated Sample - SIMCLR

Figure 7.1: Generated Sample from Latent Conditional Diffusion Model

7.5 Latent Classification Model

Finally, we implement a classification model in the latent space to perform the inverse problem of season identification. In this task, given a masked data sample, the model will classify it into a season: 'winter (0)', 'spring (1)', 'summer(2)', 'autumn (3)'. To achieve this, we implement a small linear model with a classification head in the latent space and input test data, masked with a variable ratio of [20%, 90%). We train this model for 100 epochs, choosing a learning rate of $1e^{-2}$ for the SimCLR model and $1e^{-3}$ for the autoencoder.

Figure 7.2 contrasts the training and validation curves for the latent model trained on the autoencoder embeddings vs the SimCLR embeddings. We previously mentioned how the contrastive learning objective creates more discriminative embeddings compared to an autoencoder. The results in figure 7.2 confirm this, showing the benefits of contrastive learning explicitly separating samples. We observe a smaller validation loss, as well as less overfitting and more stable training, which highlights the increased linear separability of the latent space. This is better suited for classification tasks than the autoencoder's objective.

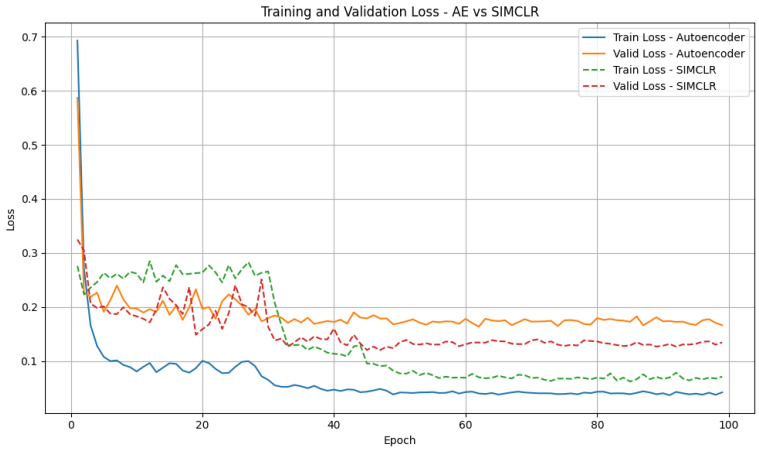


Figure 7.2: Classification Train Curves - Autoencoder vs SIMCLR

7.6 Conclusions

This chapter investigated improvements to the decoder and demonstrated the effectiveness of an optimised autoencoder approach for single-step forecasting predictions. We built upon our evaluation framework for autoregressive performance, showing that improving our data sampling approach yields substantial benefits, even with an autoencoder, due to the implicit regularisation it provides. We then examined how performance changes with strided context windows, showing that contrastive learning excels, significantly boosting forecasting performance. Finally, we extended our model to two additional downstream tasks: conditional latent generation and latent classification, revealing how contrastive learning creates a more structured latent space for both tasks compared to an autoencoder. In the following chapter, we conduct an in-depth analysis of the latent space, examining how improvements in its smoothness are correlated with downstream forecasting performance.

Chapter 8

Part 5 - Analysis of the Latent Space

8.1 Introduction

This chapter conducts a thorough analysis of the latent space, examining how smoothness correlates with downstream performance. We demonstrate how focusing on temporal smoothness, cycle distances and jerk leads to a direct improvement in strided downstream performance.

8.2 Analysis Setup

As we mentioned in chapter 2, there are several ways to measure the smoothness of the latent space. To perform a comprehensive evaluation, we adopt three metrics to measure latent smoothness.

1. Based on work by [46], we determine the temporal smoothness of our latent space by computing the squared Euclidean distance between adjacent points in the latent space: $\Delta z_t = z_t - z_{t-1}$. This provides a first-order smoothness measure.

2. Similar to chapter 6, we evaluate the cycle distance of points by computing the second-order central finite difference for embeddings: $z_{t-1} - 2z_t + z_{t+1}$.

3. As established in chapter 2, we train a Neural ODE in latent space to provide a method to compute the velocity of embeddings. We then take the output of the neural ODE and compute the second derivative, which provides the jerk, a third-order smoothness measure.

For all metrics, we evaluate our model by computing the metrics for a batch of test embeddings, using a batch size of 256. This enables us to plot the metrics and determine the overall trend of the data, as well as overall metrics such as the mean and variance. For the neural ODE, we train it for 10 epochs with a learning rate of 10^{-2} using the Adam optimiser on all embeddings from our test dataset. We then use the same batch of test data to evaluate acceleration and jerk.

8.3 Initial Analysis

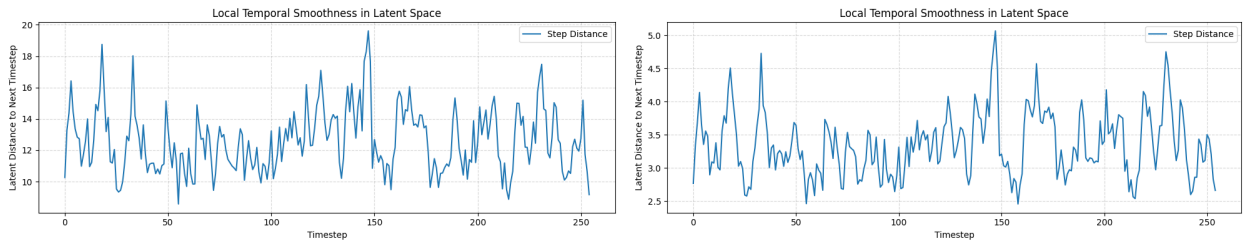
8.3.1 Autoencoder

We begin with our analysis with an autoencoder. Figures 8.1, 8.2 and 8.3 show the difference in temporal distances, cycle distances and jerk for the autoencoder with and without the improved sampling method introduced in chapter 6. The figures highlight the improved temporal and cycle distances

between the two models; however, we see slightly worse jerk.

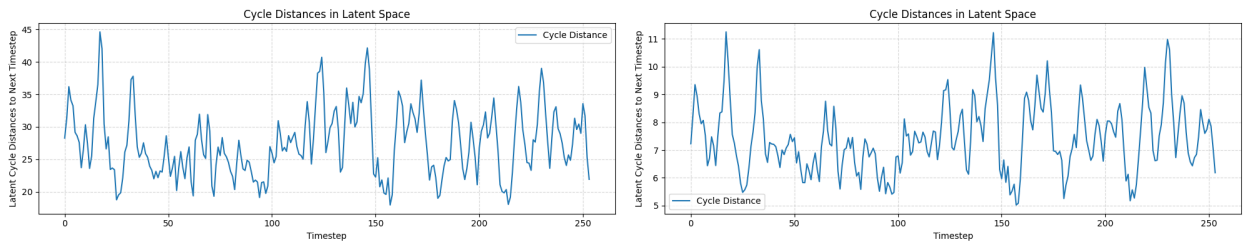
Tables 7.3 and 7.4 in chapter 7 show that the addition of the sampling approach results in a substantial improvement in autoregressive results for a stride of 1, slight improvement for a stride of 5, but worse results for a stride of 10. This leads us to our first hypothesis: that first and second-order smoothness measures are better indicators of autoregressive performance for smaller strides. Whereas, jerk, a third-order measure, better indicates performance for larger strides.

Moreover, this highlights the importance of having multiple metrics to measure the latent space. When we introduce stride, we skip over more points, so the emphasis is on the overall trend of acceleration of the points. We focus on how the latent space consistently moves over S steps. When we forecast with strides, what matters is if the velocity stays constant over the stride jump. Temporal smoothness measures speed-uniformity, while jerk shows how the change in velocity changes over time. Introducing stride ignores more of the jitter between individual timesteps, and the downstream model relies more on the overall movement of the points. A smaller jerk means more constant velocity and enables simpler linear extrapolation when forecasting, thereby improving performance.



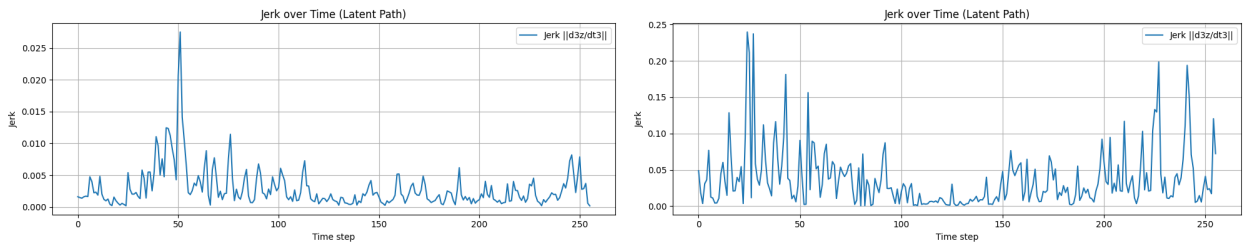
(a) Temporal Smoothness - Without Sampling Process **(b)** Temporal Smoothness - With Sampling Process

Figure 8.1: Temporal Smoothness of Autoencoder With and Without Sampling Process



(a) Cycle Distances - Without Sampling Process **(b)** Cycle Distances - With Sampling Process

Figure 8.2: Cycle Distances of Autoencoder With and Without Sampling Process

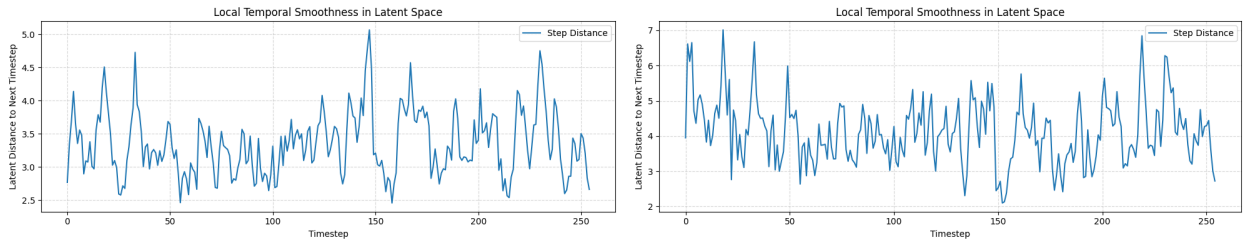


(a) Jerk - Without Sampling Process **(b)** Jerk - With Sampling Process

Figure 8.3: Jerk of Autoencoder With and Without Sampling Process

8.3.2 SimCLR

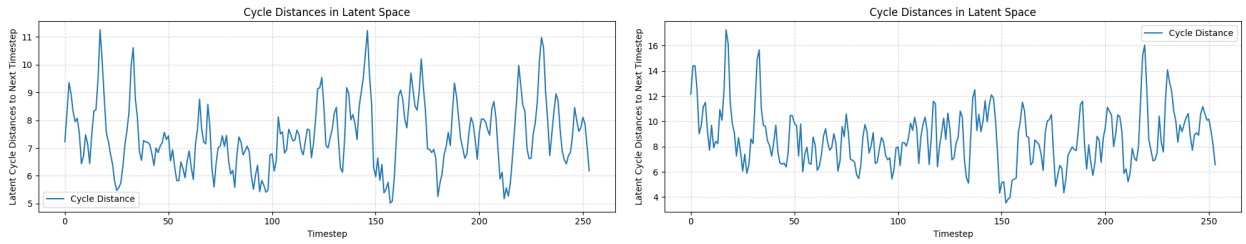
Figures 8.4, 8.5 and 8.6 show the plots for temporal distance, cycle distance and jerk for the final SIMCLR model in chapter 7 compared to the autoencoder with the sampling process. Adding contrastive learning results in improved jerk, with a slight increase in temporal and cycle distances. This is sound as SIMCLR actively pushes negative embeddings apart, resulting in worse temporal and cycle distances between points. We demonstrated in chapter 7 how this resulted in improved forecasting outcomes when stride was introduced, highlighting the significance of the jerk measure in forecasting with stride and validating our hypothesis above.



(a) Temporal Smoothness - Autoencoder With Sampling

(b) Temporal Smoothness - SIMCLR

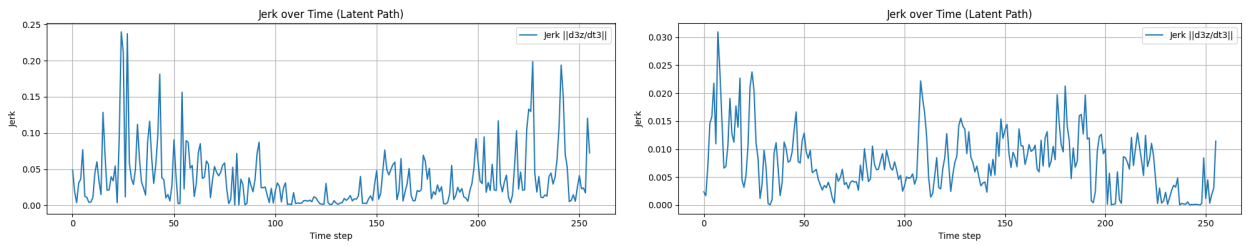
Figure 8.4: Temporal Smoothness - Autoencoder vs SIMCLR



(a) Cycle Distances - Autoencoder With Sampling

(b) Cycle Distances - SIMCLR

Figure 8.5: Cycle Distances - Autoencoder vs SIMCLR



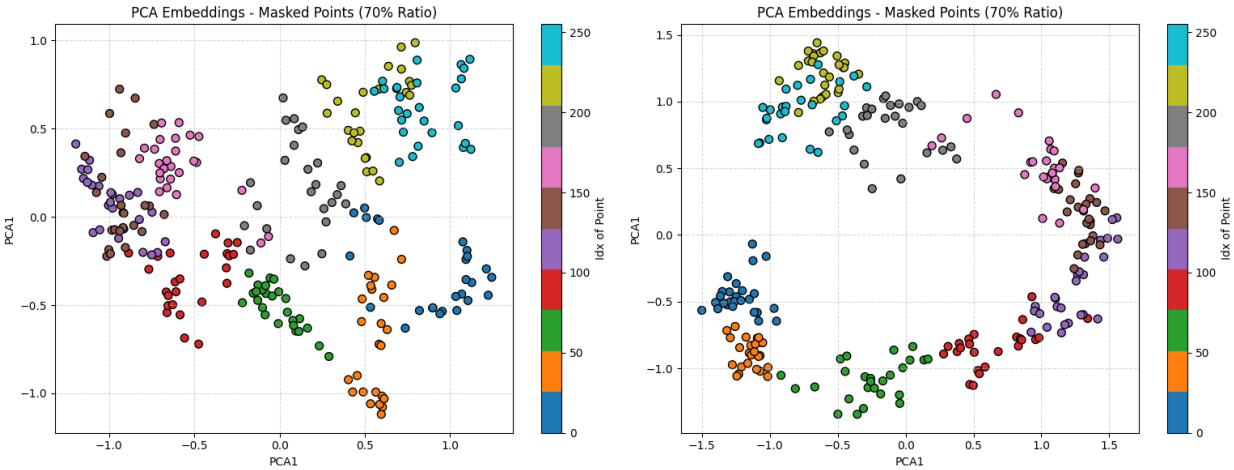
(a) Jerk - Autoencoder With Sampling Process

(b) Jerk - SIMCLR

Figure 8.6: Jerk - Autoencoder vs SIMCLR

Figure 8.7 further enhances our analysis by showing a PCA plot of our embeddings. We can see the improved temporal smoothness that SIMCLR adds. Furthermore, figure 8.8 shows the latent trajectories for 70% masked context windows of size 5, using a stride of 5. Here we show the average embedding for the context window and the direction of the next point to be forecasted. We can see that these latent trajectories support our findings. Contrastive learning generates more structured latent trajectories, with reduced overlap for strided data, resulting in improved downstream performance.

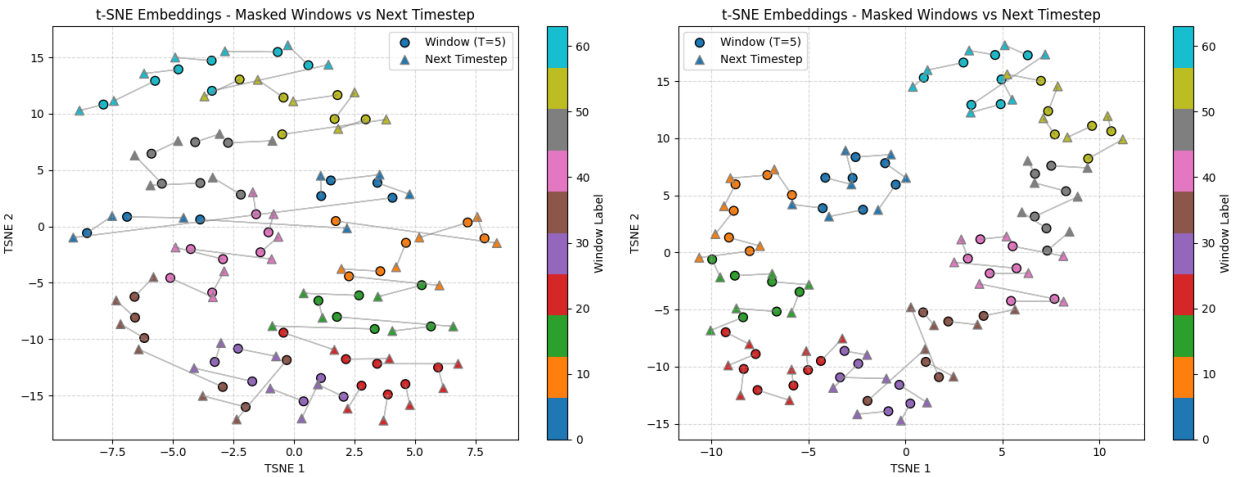
For completeness, we also include plots for a context window of 30 with a stride of 1 in figure C.19 and for a context window of 5 with a stride of 10 in figure C.20. These plots corroborate our findings, showing that SimCLR produces more structured context windows compared to the autoencoder as the stride increases.



(a) Autoencoder PCA - With Sampling Process

(b) SIMCLR PCA

Figure 8.7: PCA of Latent Points Autoencoder vs SIMCLR



(a) Latent Trajectories - Autoencoder

(b) Latent Trajectories - SIMCLR

Figure 8.8: Latent Trajectories – CW 5, Stride 5 Autoencoder vs SIMCLR

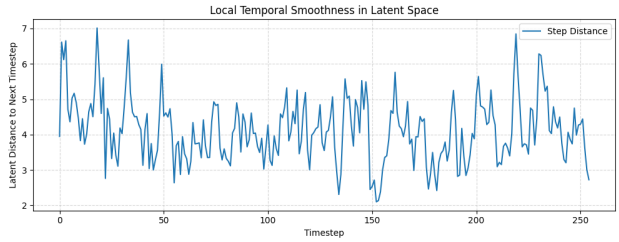
8.4 Improvements

Based on this analysis, we trialled additional improvements to our contrastive learning model. Performance results for these methods are shown in tables 8.1 and 8.2.

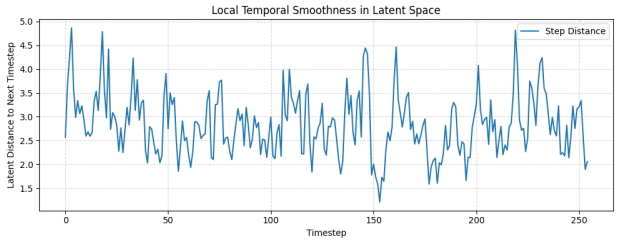
8.4.1 Batch Size

Until present, all models have been trained with a batch size of 128. Figure 2.3 from chapter 2 illustrates the impact of the batch size on SimCLR, highlighting the performance increase achieved by

using a larger batch size. We therefore trialled using a batch size of 256, illustrating the changes in smoothness in figures 8.9, 8.10, and 8.11.

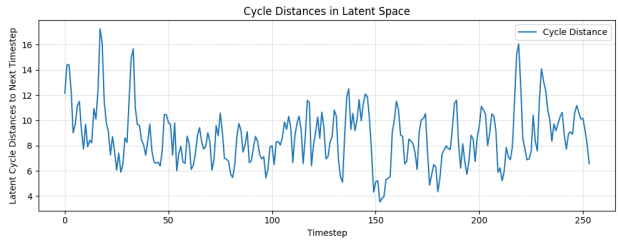


(a) Temporal Smoothness - SIMCLR

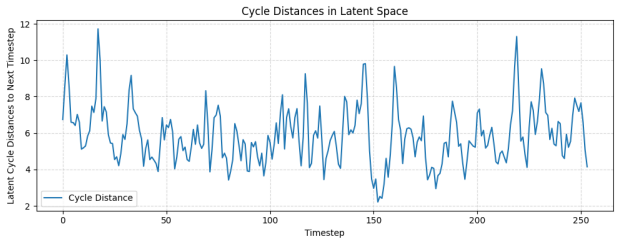


(b) Temporal Smoothness - SIMCLR - 256 Batch Size

Figure 8.9: Temporal Smoothness - 256 Batch Size

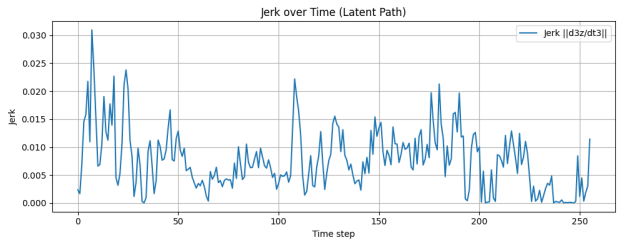


(a) Cycle Distances - SIMCLR

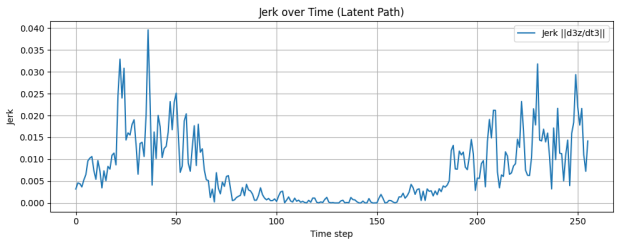


(b) Cycle Distances - SIMCLR - 256 Batch Size

Figure 8.10: Cycle Distances - 256 Batch Size



(a) Jerk - SIMCLR



(b) Jerk - SIMCLR - 256 Batch Size

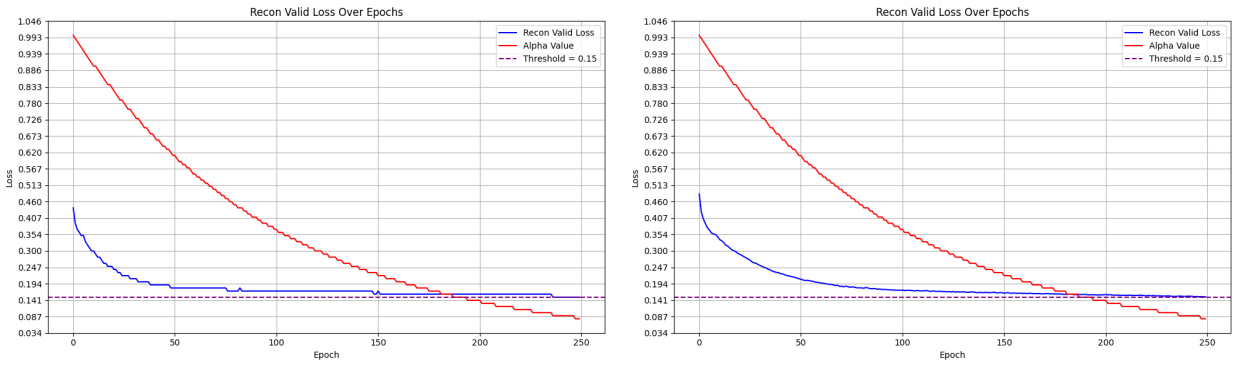
Figure 8.11: Jerk - 256 Batch Size

We observe that these changes lead to improvements in temporal and cycle distances, albeit at the expense of increased jerk. Tables 8.1 and 8.2 show that the change in batch size has minimal forecasting impact, slightly increasing performance for a stride of 5, but decreasing for a stride of 10. However, it confirms our hypothesis in section 8.3.1 that first and second-order smoothness metrics are better measures of forecasting for smaller strides. At the same time, jerk indicates forecasting performance for larger strides. Fundamentally, we demonstrate a multi-objective trade-off in the latent space; improved cycle and temporal smoothness can compromise jerk, which in turn affects the model's ability to forecast with higher strides.

8.4.2 Restricting the Decay of Alpha

For further analysis, we plot the reconstruction loss of the reconstruction objective in the fine-tuning stage against the number of epochs, for both 256 and 128 batch sizes, as shown in figure 8.12. We

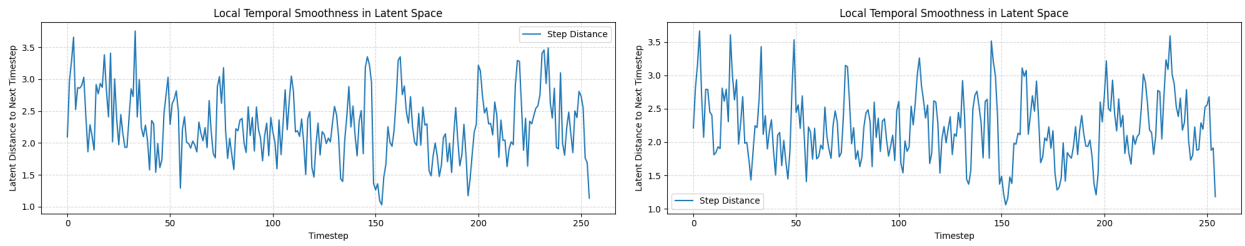
can see here that for a batch size of 128, the reconstruction loss plateaus at roughly epoch 70, when α reaches 0.5. As the autoencoder objective is blended in, the encoder focuses more on producing embeddings that are better suited for reconstruction, rather than creating a more structured latent space, as the SimCLR objective does. We find similar values for epoch 160 and alpha 0.2 when using a batch size of 256. We therefore restrict the decay of alpha, stopping it from decaying further from the values found. We hypothesise that this will better preserve the smoothness of the latent space, whilst maintaining similar reconstruction capabilities.



(a) Reconstruction Loss vs Epochs - 128 Batch Size **(b)** Reconstruction Loss vs Epochs - 256 Batch Size

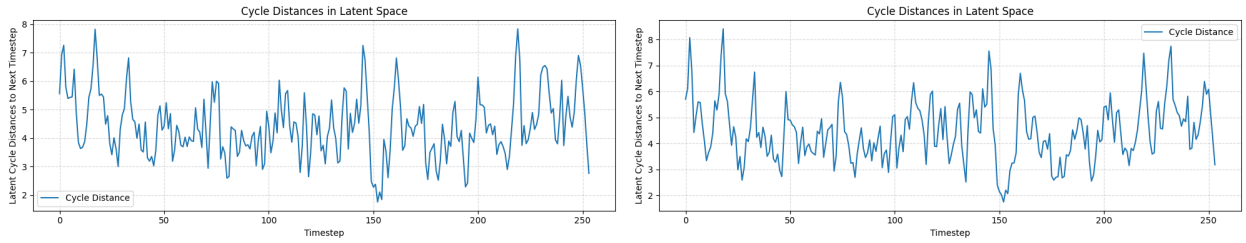
Figure 8.12: Reconstruction Loss vs Epochs

Figures 8.13, 8.14, and 8.15 show how these changes impact temporal smoothness, cycle distance and jerk. We find that for both batch sizes, this change decreases both temporal and cycle distances; however, we only see a reduction in jerk when this is applied to a batch size of 128. Nevertheless, from tables 8.1 and 8.2, we see that the most significant performance increase occurs when this change is applied to a batch size of 256. We reason that even though jerk results may suggest otherwise, the increase in quality of representations brought about by the increased batch size ultimately yields greater performance increases.



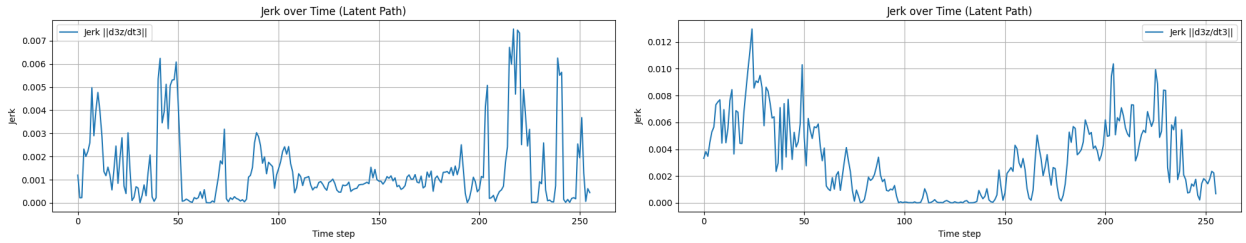
(a) Restricted Alpha - 128 Batch Size **(b)** Restricted Alpha - 256 Batch Size

Figure 8.13: Temporal Smoothness, with Restricted Alpha



(a) Restricted Alpha - 128 Batch Size **(b)** Restricted Alpha - 256 Batch Size

Figure 8.14: Cycle Distances, with Restricted Alpha



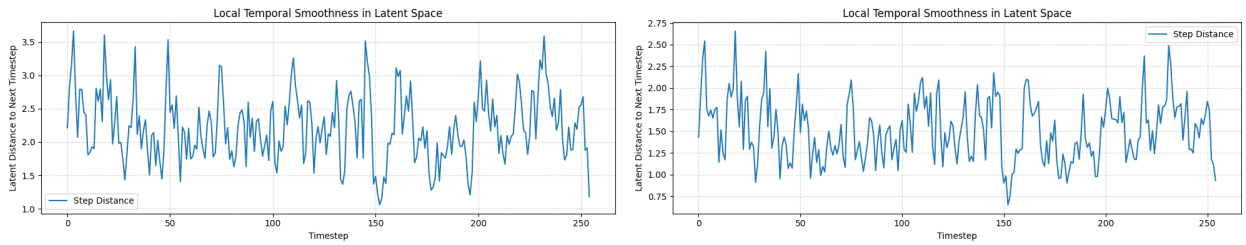
(a) Restricted Alpha - 128 Batch Size **(b)** Restricted Alpha - 256 Batch Size

Figure 8.15: Jerk, with Restricted Alpha

8.4.3 Tweaks to Cycle Consistency Loss

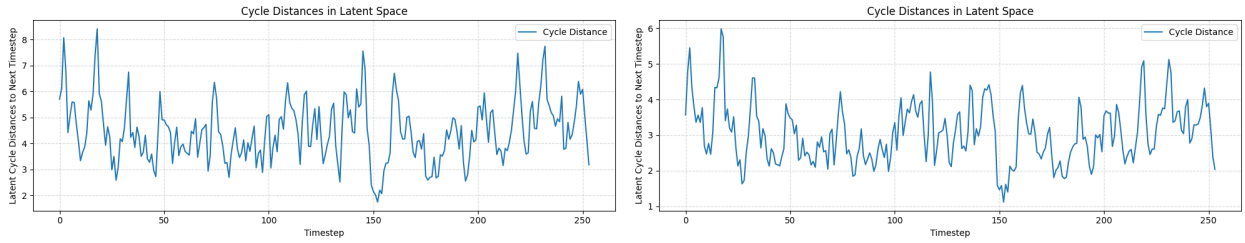
An indicator of smoothness is the central second-order finite difference between embeddings. In chapter 6, we added a cycle consistency loss to the projections. However, as the embeddings are used in the downstream task, applying the cycle consistency loss directly to the embeddings is likely to encourage smoother representations and yield greater performance. We apply this change to the model with a batch size of 256 and restricted alpha.

Figures 8.16, 8.17 and 8.18 show the impact of this change. We observe that these two changes consistently enhance smoothness across all metrics, resulting in a substantial improvement in downstream performance for a stride of 10, as shown in table 8.2.



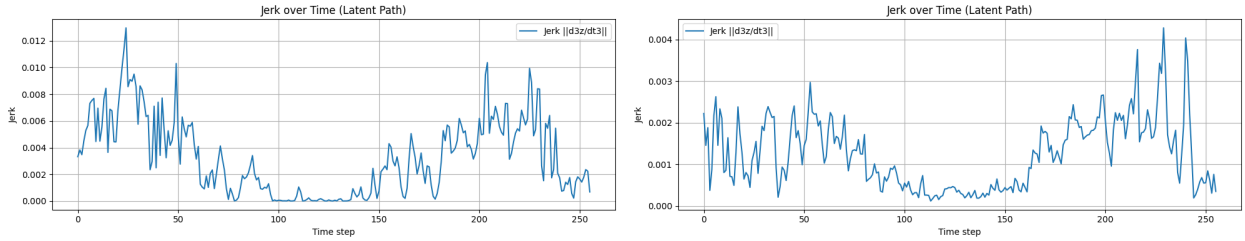
(a) SIMCLR 256 Batch Size **(b)** SIMCLR 256 Batch Size + Cycle Change

Figure 8.16: Temporal Smoothness, with changed Cycle Loss and Alpha Constraint



(a) SIMCLR 256 Batch Size **(b)** SIMCLR 256 Batch Size + Cycle Change

Figure 8.17: Cycle Distances, with changed Cycle Loss and Alpha Constraint



(a) SIMCLR 256 Batch Size **(b)** SIMCLR 256 Batch Size + Cycle Change

Figure 8.18: Jerk, with changed Cycle Loss and Alpha Constraint

8.4.4 Results

Ultimately, the results below demonstrate that combining an increased batch size, restricting alpha decay, and modifying the cycle consistency loss yields an improvement in autoregressive performance, particularly with a stride of 10. Interestingly, the changes on their own do not lead to a significant performance increase; however, due to the compounded additional smoothness they add, this results in a marked performance improvement. For completeness, we compare our final result to an autoencoder with the increased batch size and the hard negative sampling approach. We find that increasing the batch size harms the autoencoder, indicating that the optimal batch size for the autoencoder has already been identified in the previous chapter.

Table 8.1

	Context Window 5, Stride 5			
	RRMSE (%) T=25	RRMSE (%) T=50	RRMSE (%) T=75	RRMSE (%) T=100
SIMCLR	**11.50**	**11.60**	**11.58**	**11.58**
SIMCLR - 256 Batch Size	11.42	11.53	11.57	11.60
SIMCLR - 128 Batch Size, Cap Alpha	11.36	11.42	11.45	11.49
SIMCLR - 256 Batch Size, Cap Alpha	11.57	11.64	11.68	11.71
SIMCLR - 256 Batch Size, Cap Alpha, Improved Cycle Loss	**11.45**	**11.49**	**11.52**	**11.56**
Autoencoder - 256 Batch Size, With Sampling	25.94	26.19	25.96	26.07

Table 8.2

	Context Window 5, Stride 10			
	RRMSE (%) T=25	RRMSE (%) T=50	RRMSE (%) T=75	RRMSE (%) T=100
SIMCLR	**17.81**	**18.44**	**18.05**	**18.23**
SIMCLR - 256 Batch Size	19.35	20.28	20.11	20.42
SIMCLR - 128 Batch Size, Cap Alpha	18.04	18.77	18.76	18.74
SIMCLR - 256 Batch Size, Cap Alpha	16.29	16.62	16.81	17.24
SIMCLR - 256 Batch Size, Cap Alpha, Improved Cycle Loss	**13.25**	**13.52**	**13.65**	**13.85**
Autoencoder - 256 Batch Size, With Sampling	24.22	25.10	25.33	25.98

To present a comprehensive picture, we evaluate our final model on the additional downstream tasks introduced in chapter 7 (table 8.3). Notably, changes introduced in this chapter provide a trade-off between forecasting performance and conditional latent diffusion. Nevertheless, we have shown throughout this chapter and chapter 7 that contrastive learning offers significant improvements across multiple downstream tasks. This experiment highlights that finer tuning based on analysis for specific tasks can yield even greater performance.

Table 8.3

	Conditional Latent Diffusion			Latent Classification
	RRMSE (%)	**Latent Density Score**	**Realism Score**	**Test Loss**
SIMCLR	30.34	0.38	0.13	0.1343
SIMCLR - 256 Batch Size, Cap Alpha, Improved Cycle Loss	59.97	0.25	0.067	0.1346
Autoencoder - 128 Batch Size, With Sampling	37.89	0.35	0.097	0.1665

8.5 Conclusions

This chapter performed an analysis of the latent space of our models, demonstrating how improvements to downstream forecasting arise from an increase in smoothness within the latent space. We proposed three metrics, each computing a higher-order property of the latent trajectory position. We demonstrated how a focus on first and second-order smoothness measures enhances forecasting results for smaller strides, whereas a focus on third-order smoothness improves forecasting for greater strides. As such, we introduced three additional improvements to our contrastive learning method, which are combined to provide an increase in smoothness and an improvement in strided forecasting performance. In the final chapter, we will investigate multi-modal contrastive learning approaches, comparing how late-fusion methods increase performance compared to early-fusion methods.

Chapter 9

Part 6 - Multi-Modal Fusion

Chapter 10

Conclusions and Future Work

Bibliography

[1] IBMClimate Data Store. *What is self-supervised learning?* [Online]. Available from: https://www.ibm.com/think/topics/self-supervised-learning;. [Accessed: 2025-04-29]. pages 2

[2] Gui J, Chen T, Zhang J, Cao Q, Sun Z, Luo H, et al. A Survey on Self-Supervised Learning: Algorithms, Applications, and Future Trends. *IEEE Trans Pattern Anal Mach Intell*. 2024;46(12):9052–9071. Available from: https://doi.org/10.1109/TPAMI.2024.3415112. pages 2

[3] Finextra. *Self-supervised Learning The future of Artificial Intelligence* [Online]. Available from: https://www.finextra.com/blogposting/26343/self-supervised-learning-the-future-of-artificial-intelligence;. [Accessed: 2025-04-29]. pages 2

[4] Jaiswal A, Babu AR, Zadeh MZ, Banerjee D, Makedon F. A Survey on Contrastive Self-Supervised Learning. *Technologies*. 2021;9(1). Available from: https://www.mdpi.com/2227-7080/9/1/2. pages 2

[5] Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations. In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org; 2020. . pages 2, 5, 23

[6] Yue Z, Wang Y, Duan J, Yang T, Huang C, Tong Y, et al. TS2Vec: Towards Universal Representation of Time Series. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022 Jun;36(8):8980-7. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/20881. pages 2

[7] Dexter. *Leveraging 500TB of weather forecasts: A paradigm shift to array-native infrastructure* [Online]. Available from: https://dexterenergy.ai/news/leveraging-500tb-of-weather-forecasts-a-paradigm-shift-to-array-native-infrastructure/;. [Accessed: 2025-04-29]. pages 2

[8] Price I, Sanchez-Gonzalez A, Alet F, Andersson TR, El-Kadi A, Masters D, et al. Probabilistic weather forecasting with machine learning. *Nature*. 2025;637(8044):84-90. Available from: https://www.nature.com/articles/s41586-024-08252-9. pages 2

[9] Bi K, Xie L, Zhang H, Chen X, Gu X, Tian Q. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*. 2023;619(7970):533-8. pages 2, 3

[10] WeatherBench 2. *WeatherBench 2 Data Guide* [Online]. Available from: https://weatherbench2.readthedocs.io/en/latest/data-guide.html;. [Accessed: 2025-04-28]. pages 3, 22

[11] Li S, Tang H. Multimodal Alignment and Fusion: A Survey arXiv [Preprint]; 2024 Version 1. Available from: `https://arxiv.org/abs/2411.17040`. pages 3, 10

[12] Agabin A, Prochaska JX, Cornillon PC, Buckingham CE. Mitigating Masked Pixels in a Climate-Critical Ocean Dataset. *Remote Sensing*. 2024;16(13). Available from: `https://www.mdpi.com/2072-4292/16/13/2439`. pages 3, 14, 21

[13] Khosla P, Teterwak P, Wang C, Sarna A, Tian Y, Isola P, et al. Supervised contrastive learning. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Red Hook, NY, USA: Curran Associates Inc.; 2020. . pages 6, 23

[14] Zbontar J, Jing L, Misra I, LeCun Y, Deny S. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In: Meila M, Zhang T, editors. *Proceedings of the 38th International Conference on Machine Learning*. vol. 139 of Proceedings of Machine Learning Research. PMLR; 2021. p. 12310-20. Available from: `https://proceedings.mlr.press/v139/zbontar21a.html`. pages 6, 23, 24

[15] Zhao W, Fan L. Time-series representation learning via Time-Frequency Fusion Contrasting. *Frontiers in Artificial Intelligence*. 2024;Volume 7 - 2024. Available from: `https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1414352`. pages 8, 38

[16] Lee S, Park T, Lee K. Soft contrastive learning for time series arXiv [Preprint]; 2024 Version 3. Available from: `https://arxiv.org/abs/2312.16424`. pages 9, 38, 39

[17] Shamba AK, Bach K, Taylor G. Dynamic Contrastive Learning for Time Series Representation arXiv [Preprint]; 2024 Version 1. Available from: `https://arxiv.org/abs/2410.15416`. pages 9, 39, 41

[18] Tang Q, Liang J, Zhu F. A comparative review on multi-modal sensors fusion based on deep learning. *Signal Processing*. 2023;213:109165. Available from: `https://www.sciencedirect.com/science/article/pii/S0165168423002396`. pages 10

[19] Ge H, Wang L, Pan H, Liu Y, Li C, Lv D, et al. Cross Attention-Based Multi-Scale Convolutional Fusion Network for Hyperspectral and LiDAR Joint Classification. *Remote Sensing*. 2024;16(21). Available from: `https://www.mdpi.com/2072-4292/16/21/4073`. pages 10

[20] Dufumier B, Castillo-Navarro J, Tuia D, Thiran JP. What to align in multimodal contrastive learning? In: *International Conference on Learning Representations*; 2025. Available from: `https://arxiv.org/abs/2409.07402`. pages 10, 11

[21] Guo H, Xu X, Wu H, Liu B, Xia J, Cheng Y, et al. Multi-scale and multi-modal contrastive learning network for biomedical time series. *Biomedical Signal Processing and Control*. 2025;106:107697. Available from: `https://www.sciencedirect.com/science/article/pii/S1746809425002083`. pages 11

[22] Bahaduri B, Ming Z, Feng F, Mokraou A. Multimodal Transformer Using Cross-Channel attention for Object Detection in Remote Sensing Images arXiv [Preprint]; 2024 Version 3. Available from: `https://arxiv.org/abs/2310.13876`. pages 11

[23] Saioni M, Giannone C. Multimodal Attention Is All You Need. In: Dell'Orletta F, Lenci A, Montemagni S, Sprugnoli R, editors. *Proceedings of the 10th Italian Conference on Computational Linguistics (CLiC-it 2024)*. Pisa, Italy: CEUR Workshop Proceedings; 2024. p. 873-9. Available from: `https://aclanthology.org/2024.clicit-1.94/`. pages 11

[24] Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, et al. Learning Transferable Visual Models From Natural Language Supervision. In: Meila M, Zhang T, editors. *Proceedings of the 38th International Conference on Machine Learning*. vol. 139 of Proceedings of Machine Learning Research. PMLR; 2021. p. 8748-63. Available from: `https://proceedings.mlr.press/v139/radford21a.html`. pages 12

[25] Cepeda VV, Nayak GK, Shah M. GeoCLIP: clip-inspired alignment between locations and images for effective worldwide geo-localization. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS '23. Red Hook, NY, USA: Curran Associates Inc.; 2023. . pages 12

[26] Jia C, Yang Y, Xia Y, Chen YT, Parekh Z, Pham H, et al. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. In: Meila M, Zhang T, editors. *Proceedings of the 38th International Conference on Machine Learning*. vol. 139 of Proceedings of Machine Learning Research. PMLR; 2021. p. 4904-16. Available from: `https://proceedings.mlr.press/v139/jia21b.html`. pages 12

[27] Foster D, Friston K. Generative deep learning : teaching machines to paint, write, compose, and play. Second edition. ed. Sebastopol, CA: O'Reilly Media, Incorporated; 2023 - 2023. pages 12, 17

[28] Wang L, Li Q, Lv Q. Self-Supervised Classification of Weather Systems Based on Spatiotemporal Contrastive Learning. *Geophysical Research Letters*. 2022;49(15):e2022GL099131. E2022GL099131 2022GL099131. Available from: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2022GL099131`. pages 12, 20, 22, 23, 24, 28, 39

[29] Wang C, Yang N, Li X. Advancing forecasting capabilities: A contrastive learning model for forecasting tropical cyclone rapid intensification. *Proceedings of the National Academy of Sciences*. 2025;122(4):e2415501122. Available from: `https://www.pnas.org/doi/abs/10.1073/pnas.2415501122`. pages 12, 20, 22

[30] Ballard T. Contrastive Learning for Climate Model Bias Correction and Super-Resolution. In: *AAAI 2022 Fall Symposium: The Role of AI in Responding to Climate Challenges*; 2022. Available from: `https://www.climatechange.ai/papers/aaaifss2022/10`. pages 13

[31] Shi Z, Zheng H, Dong J. Spatiotemporal self-supervised predictive learning for atmospheric variable prediction via multi-group multi-attention. *Knowledge-Based Systems*. 2024;300:112090. Available from: `https://www.sciencedirect.com/science/article/pii/S095070512400724X`. pages 13

[32] Han T, Chen Z, Guo S, Xu W, Bai L. CRA5: Extreme Compression of ERA5 for Portable Global Climate and Weather Research via an Efficient Variational Transformer arXiv [preprint]; 2024 Version 2. Available from: `https://arxiv.org/abs/2405.03376`. pages 13, 20

[33] Goh E, Yepremyan AR, Wang J, Wilson B. MAESSTRO: Masked Autoencoders for Sea Surface Temperature Reconstruction under Occlusion. *EGUsphere*. 2023;2023:1-20. Available from: `https://egusphere.copernicus.org/preprints/2023/egusphere-2023-1385/`. pages 14, 21

[34] He K, Chen X, Xie S, Li Y, Dollár P, Girshick R. Masked Autoencoders Are Scalable Vision Learners. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2022. p. 15979-88. pages 14

[35] Vandal TJ, Duffy K, McDuff D, Nachmany Y, Hartshorn C. Global atmospheric data assimilation with multi-modal masked autoencoders arXiv [preprint]; 2024 Version 1. Available from: `https://arxiv.org/abs/2407.11696`. pages 14, 21

[36] Kadeethum T, Ballarin F, O'Malley D, Choi Y, Bouklas N, Yoon H. Reduced order modeling for flow and transport problems with Barlow Twins self-supervised learning. *Sci Rep*. 2022 Nov;12(1):20654. pages 15, 17, 29

[37] Kadeethum T, Silva VLS, Salinas P, Pain CC, Yoon H. Boosting Barlow Twins Reduced Order Modeling for Machine Learning-Based Surrogate Models in Multiphase Flow Problems. *Water Resources Research*. 2024;60(10). Available from: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2023WR035778`. pages 16

[38] Quetin S, Ghosh T, Maleki F. Should we pre-train a decoder in contrastive learning for dense prediction tasks? arXiv [preprint]; 2025 Version 1. Available from: `https://arxiv.org/abs/2503.17526`. pages 16, 29, 30

[39] Dippel J, Vogler S, Höhne J. Towards Fine-grained Visual Representations by Combining Contrastive Learning with Image Reconstruction and Attention-weighted Pooling. *CoRR*. 2021. Available from: `https://arxiv.org/abs/2104.04323`. pages 16, 17, 29

[40] Yao Y, Desai N, Palaniswami M. Masked Contrastive Representation Learning arXiv [preprint]; 2022 Version 1. Available from: `https://arxiv.org/abs/2211.06012`. pages 16, 17, 29

[41] He K, Fan H, Wu Y, Xie S, Girshick R. Momentum Contrast for Unsupervised Visual Representation Learning. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2020. p. 9726-35. pages 17

[42] Huang Z, Jin X, Lu C, Hou Q, Cheng MM, Fu D, et al. Contrastive Masked Autoencoders are Stronger Vision Learners. *IEEE Trans Pattern Anal Mach Intell*. 2024 Apr;46(4):2506–2517. Available from: `https://doi.org/10.1109/TPAMI.2023.3336525`. pages 17, 29

[43] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Red Hook, NY, USA: Curran Associates Inc.; 2020. . pages 17

[44] Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-Resolution Image Synthesis With Latent Diffusion Models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2022. p. 10684-95. pages 18, 48

[45] Zhuang Y, Cheng S, Duraisamy K. Spatially-aware diffusion models with cross-attention for global field reconstruction with sparse observations. *Computer Methods in Applied Mechanics and Engineering*. 2025;435:117623. Available from: `https://www.sciencedirect.com/science/article/pii/S0045782524008776`. pages 18, 48

[46] Tzaban R, Mokady R, Gal R, Bermano A, Cohen-Or D. Stitch it in Time: GAN-Based Facial Editing of Real Videos. In: *SIGGRAPH Asia 2022 Conference Papers*. SA '22. New York, NY, USA: Association for Computing Machinery; 2022. Available from: `https://doi.org/10.1145/3550469.3555382`. pages 19, 51

[47] Kim G, Shim H, Kim H, Choi Y, Kim J, Yang E. Diffusion Video Autoencoders: Toward Temporally Consistent Face Video Editing via Disentangled Video Encoding. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2023. p. 6091-100. pages 19

[48] Xie X, Mowlavi S, Benosman M. Smooth and Sparse Latent Dynamics in Operator Learning with Jerk Regularization arXiv [Preprint]; 2024 Version 1. Available from: `https://arxiv.org/abs/2402.15636`. pages 19, 41

[49] Chen RTQ, Rubanova Y, Bettencourt J, Duvenaud D. Neural ordinary differential equations. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Red Hook, NY, USA: Curran Associates Inc.; 2018. p. 6572–6583. pages 19

[50] Xu J, Le H, Samaras D. Assessing Sample Quality via the Latent Space of Generative Models. In: Leonardis A, Ricci E, Roth S, Russakovsky O, Sattler T, Varol G, editors. *Computer Vision – ECCV 2024*. Cham: Springer Nature Switzerland; 2025. p. 449-64. pages 20

[51] Kynkäänniemi T, Karras T, Laine S, Lehtinen J, Aila T. In: Improved precision and recall metric for assessing generative models. Red Hook, NY, USA: Curran Associates Inc.; 2019. . pages 20

[52] Climate Data Store. *ERA5 hourly data on single levels from 1940 to present* [Online]. Available from: `https://cds.climate.copernicus.eu/datasets/reanalysis-era5-single-levels?tab=overview`;. [Accessed: 2025-04-28]. pages 22

[53] Kingma D, Ba J. *Adam: A Method for Stochastic Optimization*. International Conference on Learning Representations. 2014 12. pages 24, 26

[54] Shrestha S, Zaramella M, Callegari M, Greifeneder F, Borga M. Evaluation of the ERA5 reanalysis as a reference dataset for fine-scale hydrological modelling over alpine basins. In: *EGU General Assembly Conference Abstracts*. EGU General Assembly Conference Abstracts; 2021. p. EGU21-14265. pages 28

[55] PyTorch. *BatchNorm2d* [Online]. Available from: `https://docs.pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html`;. [Accessed: 2025-06-03]. pages 31

[56] Richemond PH, Grill JB, Altché F, Tallec C, Strub F, Brock A, et al.. BYOL works even without batch statistics arXiv [Preprint]; 2020 Version 1. Available from: `https://arxiv.org/abs/2010.10241`. pages 31

[57] Despotovic M, Nedic V, Despotovic D, Cvetanovic S. Evaluation of empirical models for predicting monthly mean horizontal diffuse solar radiation. *Renewable and Sustainable Energy Reviews*. 2016;56:246-60. Available from: `https://www.sciencedirect.com/science/article/pii/S1364032115013258`. pages 35

[58] Li MF, Tang XP, Wu W, Liu HB. General models for estimating daily global solar radiation for different solar radiation zones in mainland China. *Energy Conversion and Management*. 2013;70:139-48. Available from: `https://www.sciencedirect.com/science/article/pii/S0196890413001118`. pages 35

[59] Wikipedia. *Structural similarity index measure* [Online]. Available from: `https://en.wikipedia.org/wiki/Structural_similarity_index_measure`;. [Accessed: 2025-06-04]. pages 35

[60] Wikipedia. *Peak signal-to-noise ratio* [Online]. Available from: `https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio`;. [Accessed: 2025-06-04]. pages 35

[61] Wikipedia. *Matérn covariance function* [Online]. Available from: `https://en.wikipedia.org/wiki/Matrn_covariance_function`;. [Accessed: 2025-06-06]. pages 35

[62] Rasmussen CE, Williams CKI. Gaussian Processes for Machine Learning. Cambridge, Mass.: MIT Press; 2006. pages 36

[63] Mishra S, Robinson J, Chang H, Jacobs D, Sarna A, Maschinot A, et al.. A simple, efficient and scalable contrastive masked autoencoder for learning visual representations arXiv [Preprint]; 2022 Version 1. Available from: `https://arxiv.org/abs/2210.16870`. pages 43

[64] Tian Y, Sun C, Poole B, Krishnan D, Schmid C, Isola P. What makes for good views for contrastive learning? In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Red Hook, NY, USA: Curran Associates Inc.; 2020. . pages 43

[65] Widad Zizouan, Theo Michel. *Summary Report : A Simple Framework for Contrastive Learning of Visual Representations* [Online]. Available from: `https://hal.science/hal-04175012/file/Summary_Report___A_Simple_Framework_for_Contrastive_Learning_of_Visual_Representations.pdf;`. [Accessed: 2025-06-10]. pages 43

[66] Wang T, Isola P. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In: III HD, Singh A, editors. *Proceedings of the 37th International Conference on Machine Learning*. vol. 119 of Proceedings of Machine Learning Research. PMLR; 2020. p. 9929-39. Available from: `https://proceedings.mlr.press/v119/wang20k.html`. pages 48

[67] Park T, Efros AA, Zhang R, Zhu JY. Contrastive Learning for Unpaired Image-to-Image Translation. In: Vedaldi A, Bischof H, Brox T, Frahm JM, editors. *Computer Vision – ECCV 2020*. Cham: Springer International Publishing; 2020. p. 319-45. pages 48

[68] Nam H, Kwon G, Park GY, Ye JC. Contrastive Denoising Score for Text-guided Latent Diffusion Image Editing. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2024. p. 9192-201. pages 48

Chapter 11

Declarations

Appendix A

Dataset and Augmentation Examples

A.1 Example ERA5 Data Sample

This section shows an example ERA5 data sample.

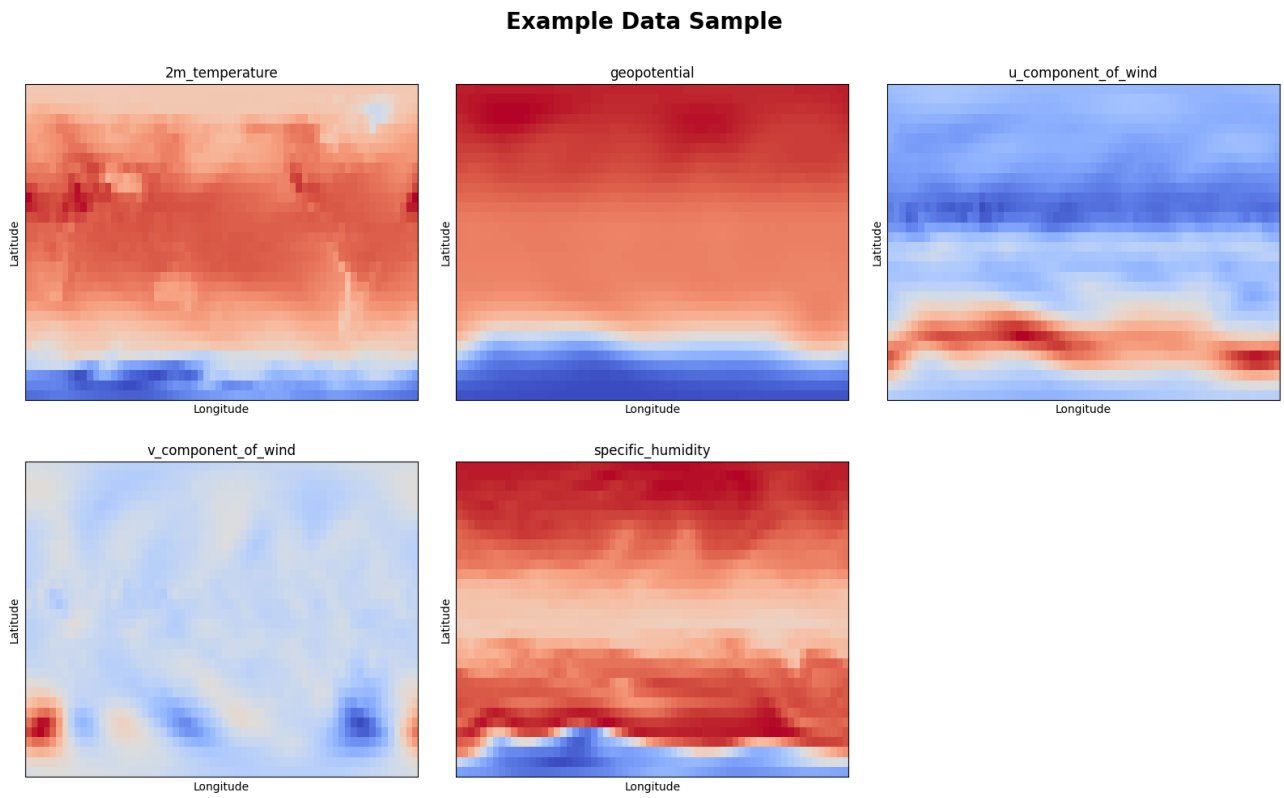


Figure A.1: Example ERA5 64x32 Data Samples

A.2 Sparse Positive Pair

This section visualizes how a sparse positive pair is constructed as referenced in chapter 5.

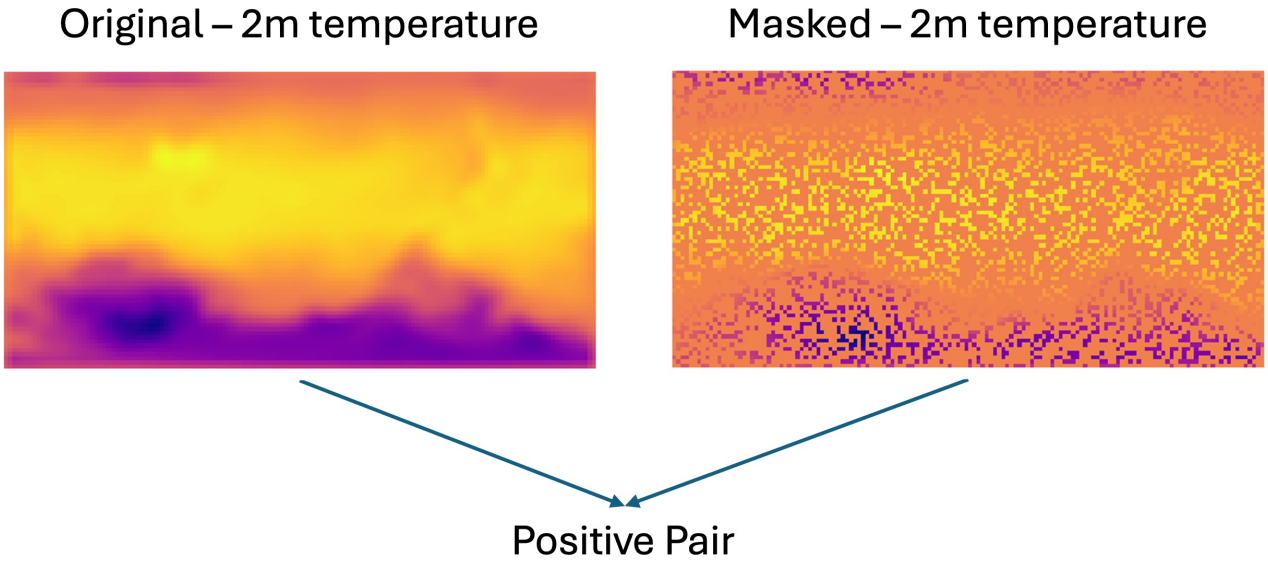


Figure A.2: Sparse Positive Pair

Appendix B

Core Model Architectures

B.1 Hybrid Encoder-Decoder Architecture - Chapter 5

This section shows the model architecture used in chapter 5.

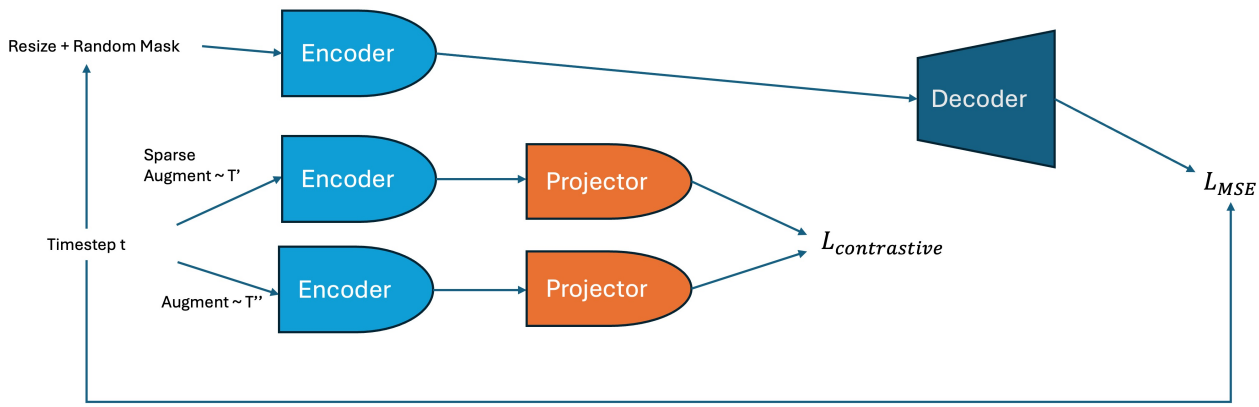


Figure B.1: Model Diagram - Chapter 5

B.2 Final Robust Model Architecture with Cycle Loss - Chapter 6

This section shows the model architecture used in chapter 6.

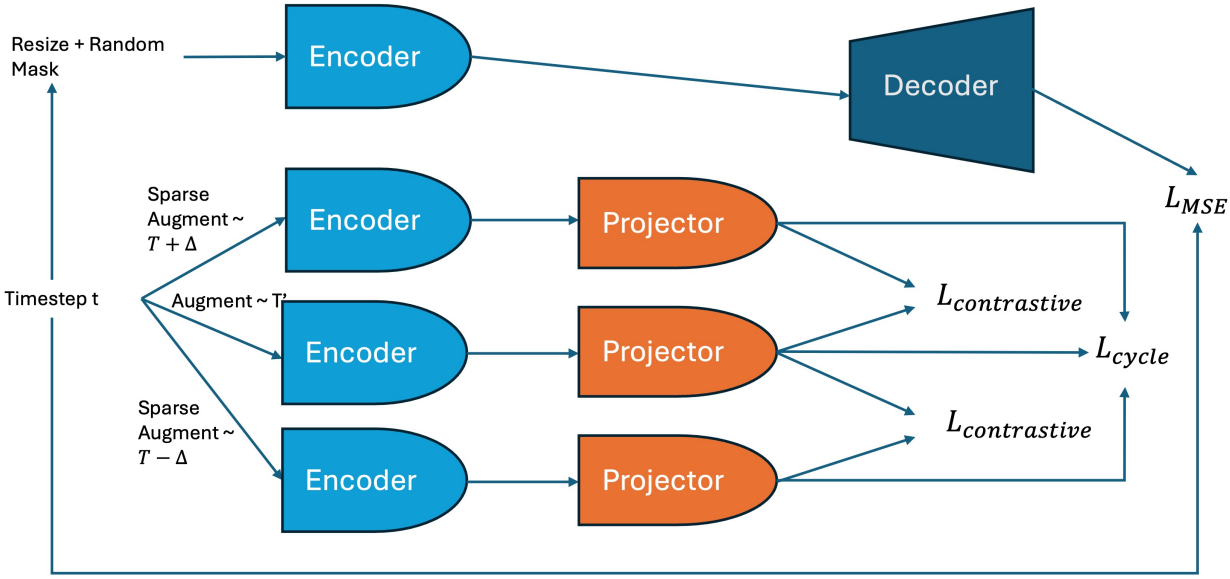


Figure B.2: Model Diagram - Chapter 6

B.3 Downstream LSTM Forecasting Model

This section shows the downstream model used in chapters 6 and 7.

B.3.1 Latent Space

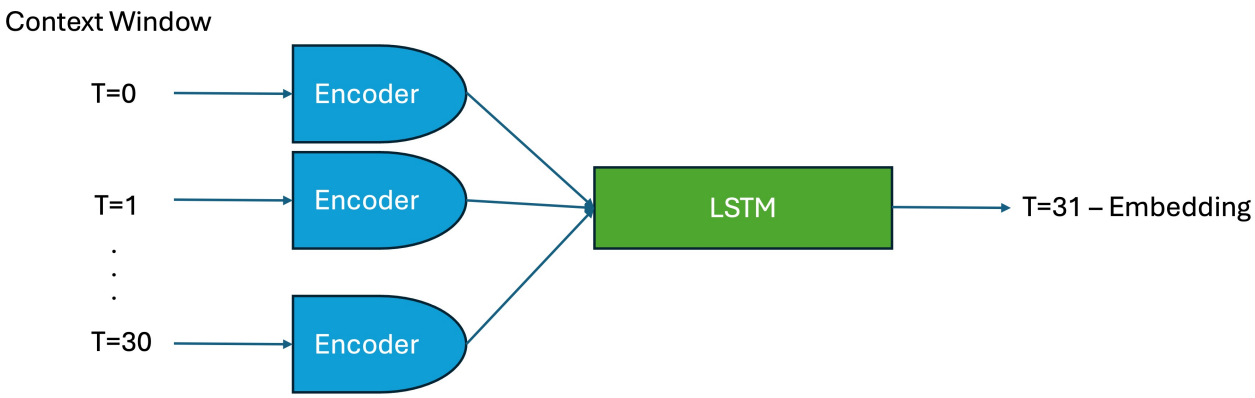


Figure B.3: Downstream Model - In Latent Space

B.3.2 Decoded to Full Space

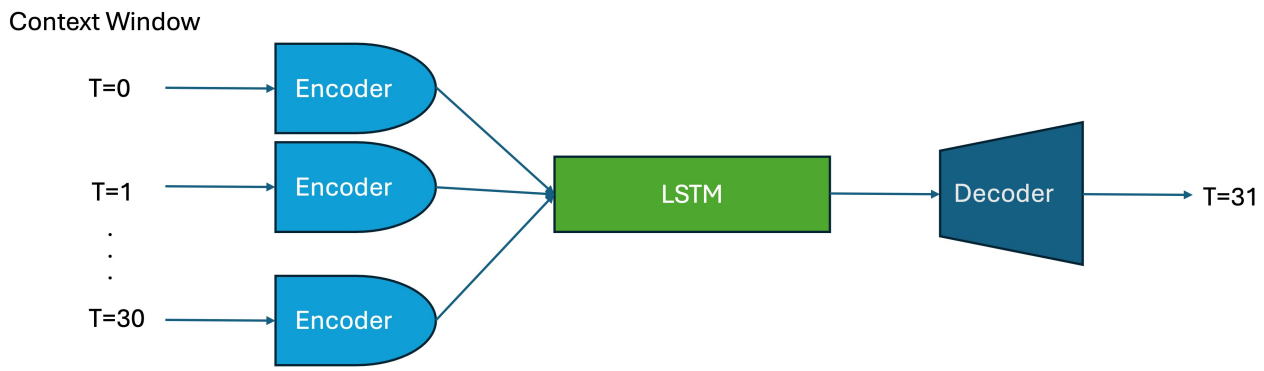


Figure B.4: Downstream Model - With Decoder

B.4 Conditional Latent Diffusion U-Net Architecture

This section shows the architecture of the conditional latent diffusion model used in chapter 7.

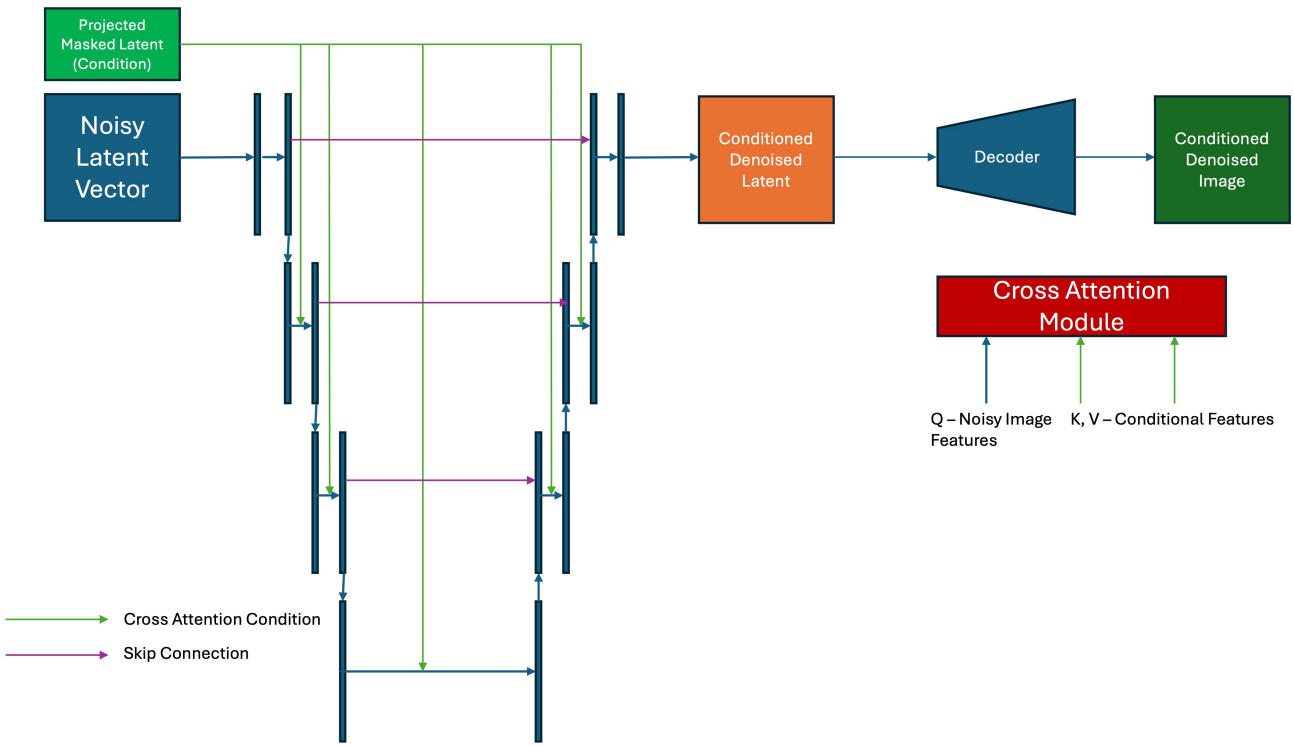


Figure B.5: Conditional Latent Diffusion U-Net Architecture

Appendix C

Supplementary Results

C.1 Histogram of Per-Channel Reconstruction Errors

This section contains the histogram showing RRMSE errors per mode, as referenced in chapter 7.

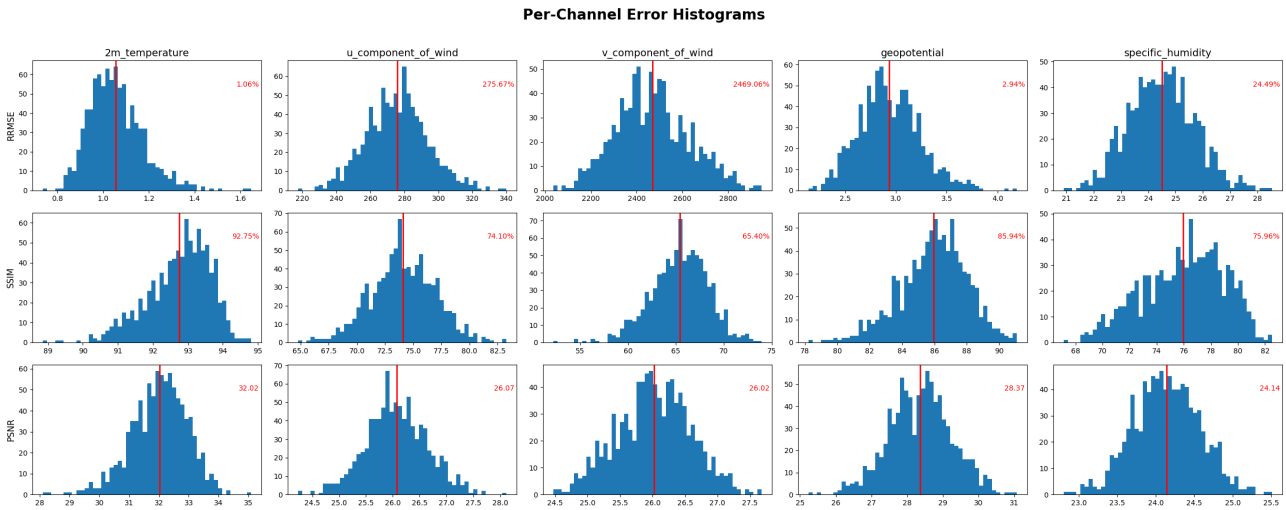


Figure C.1: Histogram of Errors per Mode

C.2 Absolute Errors

This section shows the absolute errors and corresponding heatmaps for the same random data sample, referenced in chapter 6.

C.2.1 Absolute Errors - Autoencoder

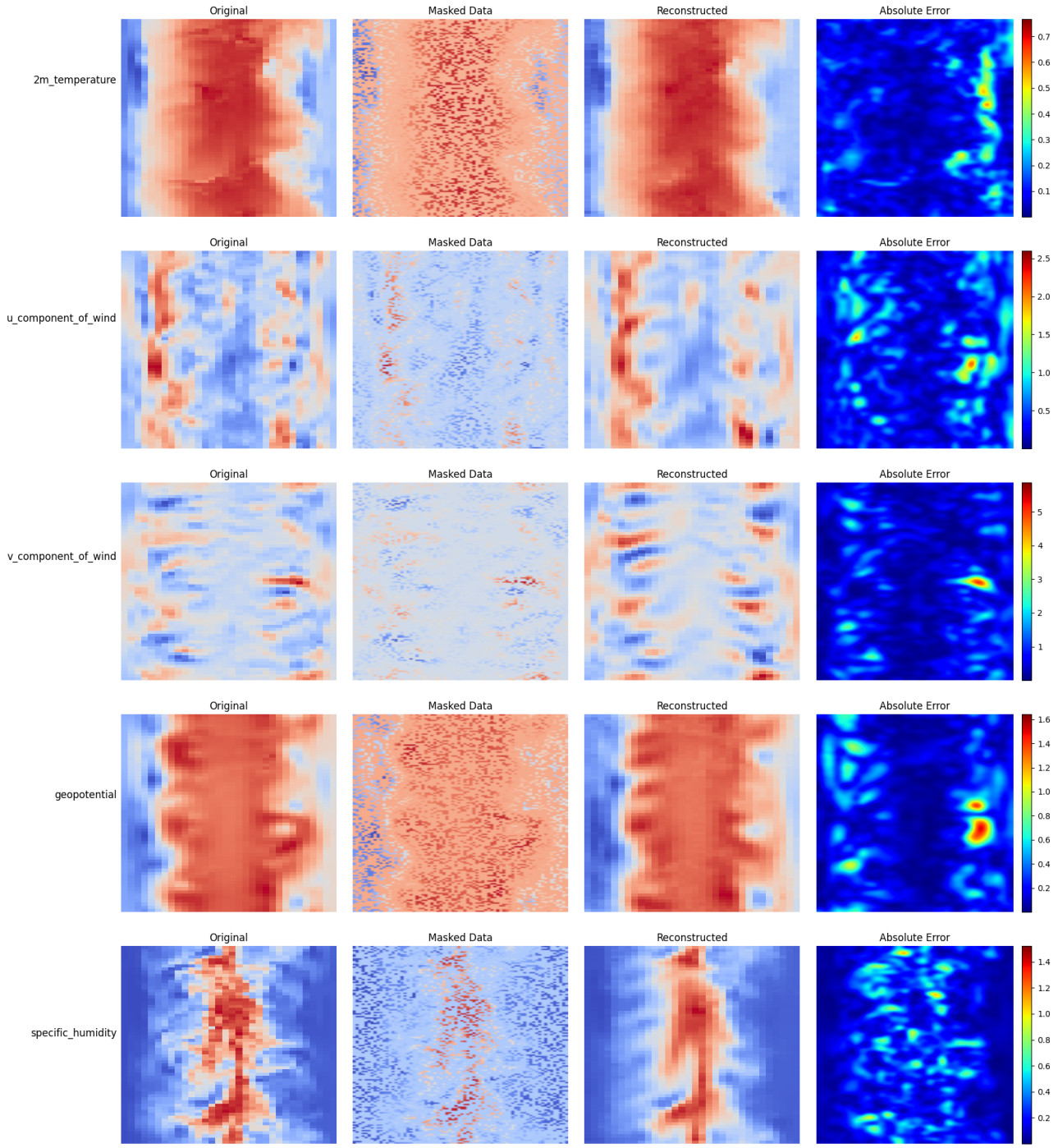


Figure C.2: Heatmap of Absolute Errors for a Random Data Sample - Autoencoder

C.2.2 Absolute Errors - SimCLR

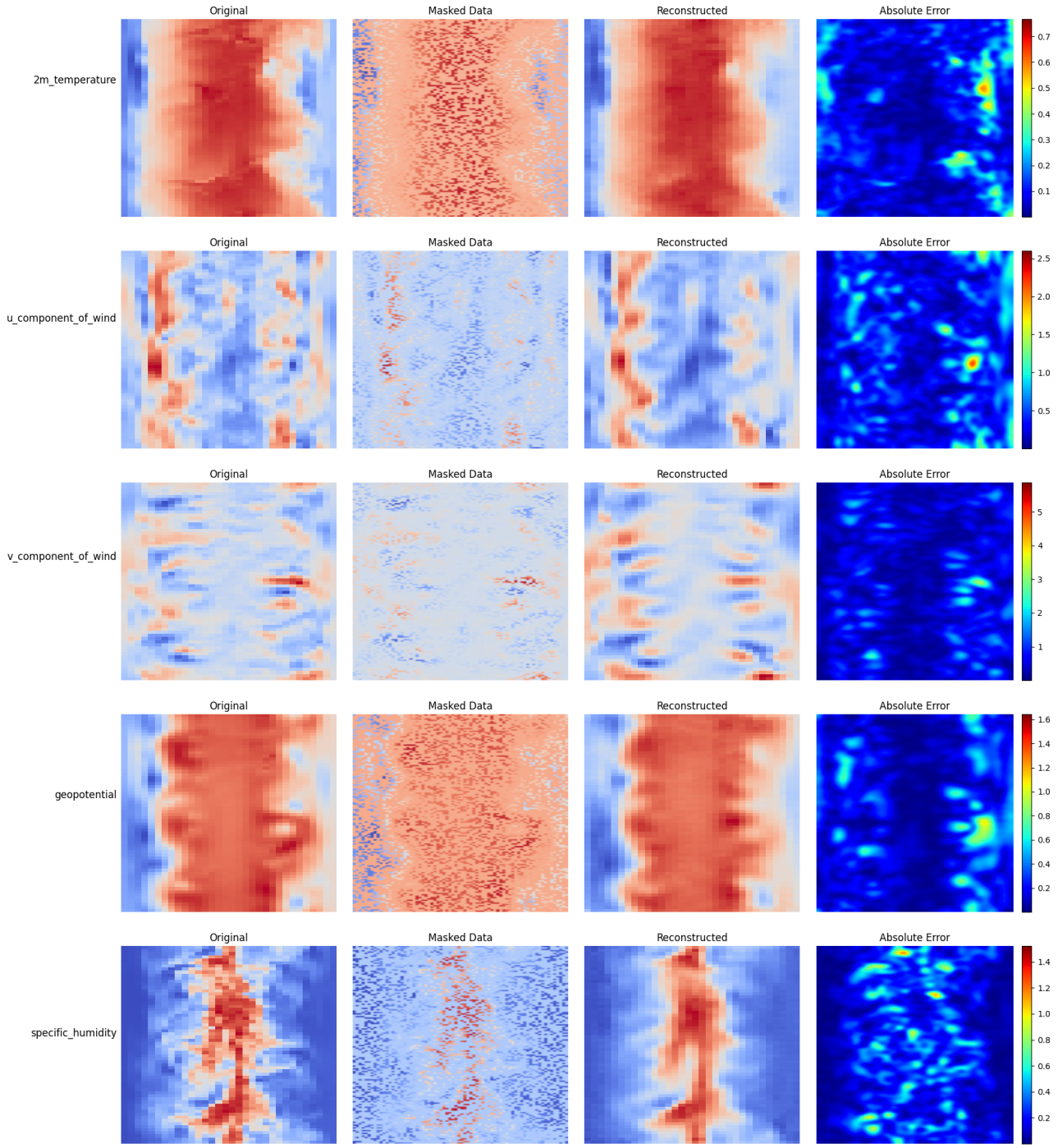


Figure C.3: Heatmap of Absolute Errors for a Random Data Sample - SimCLR

C.3 Autoregressive Results

This section contains the full autoregressive forecasting results, corresponding to the summary tables in chapters 7 and 8.

C.3.1 Autoencoder - L1 + L2 Decay

Context Window 30, Stride 1

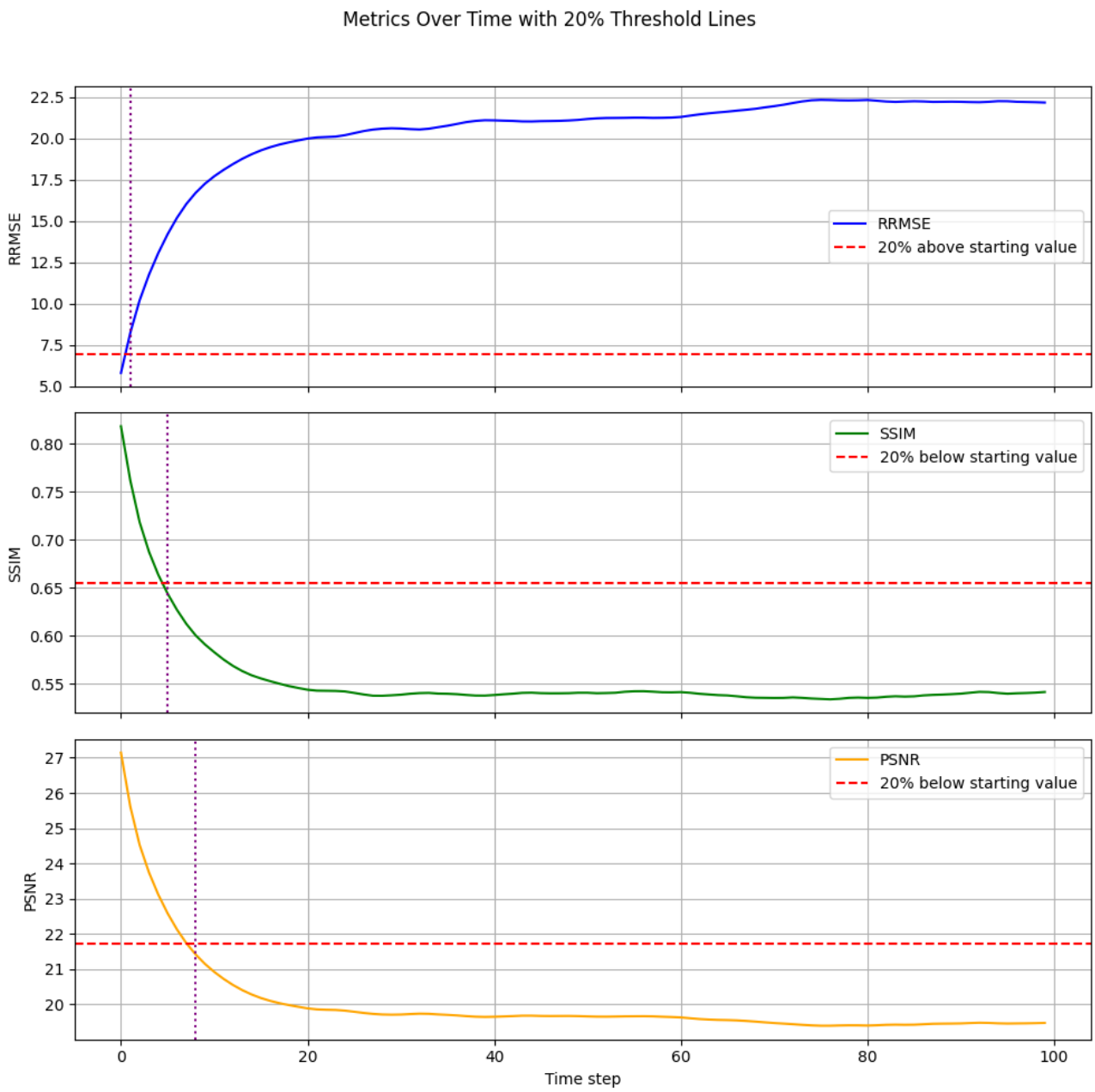


Figure C.4: Autoregressive Results - Context Window 30, Stride 1

Context Window 5, Stride 5

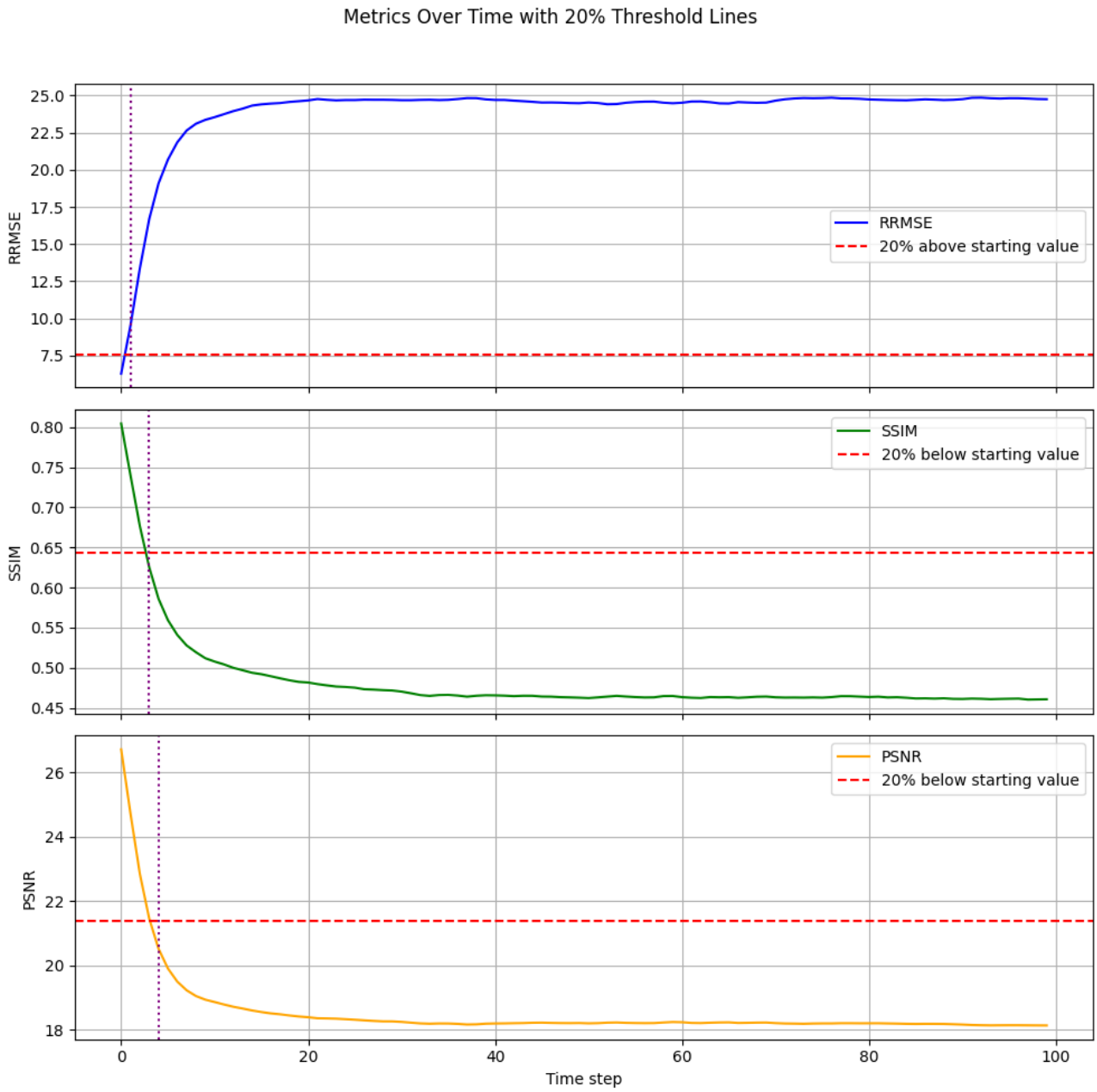


Figure C.5: Autoregressive Results - Context Window 5, Stride 5

Context Window 5, Stride 10

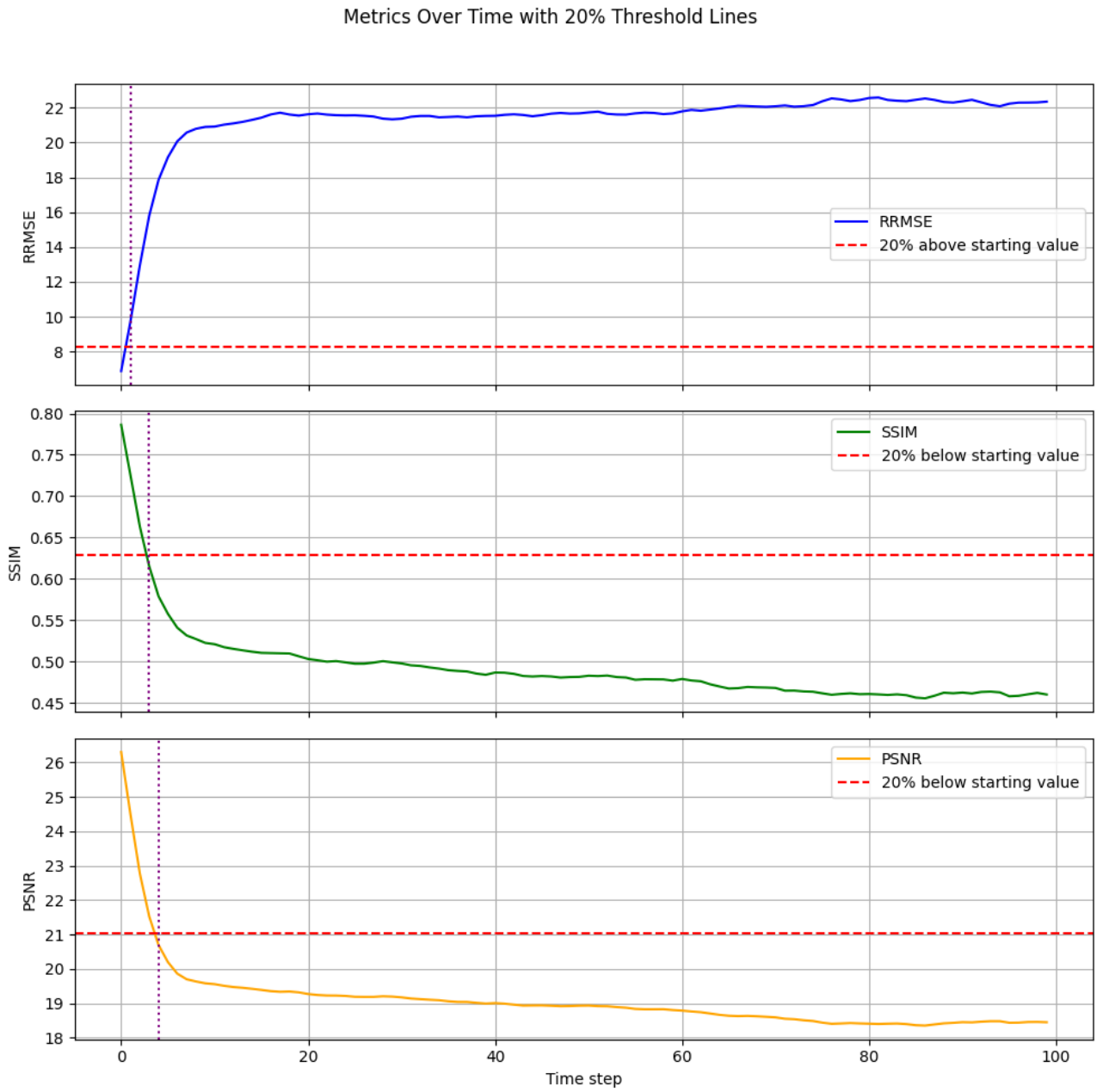


Figure C.6: Autoregressive Results - Context Window 5, Stride 10

C.3.2 Autoencoder - L1 + L2 Decay + Sampling

Context Window 30, Stride 1

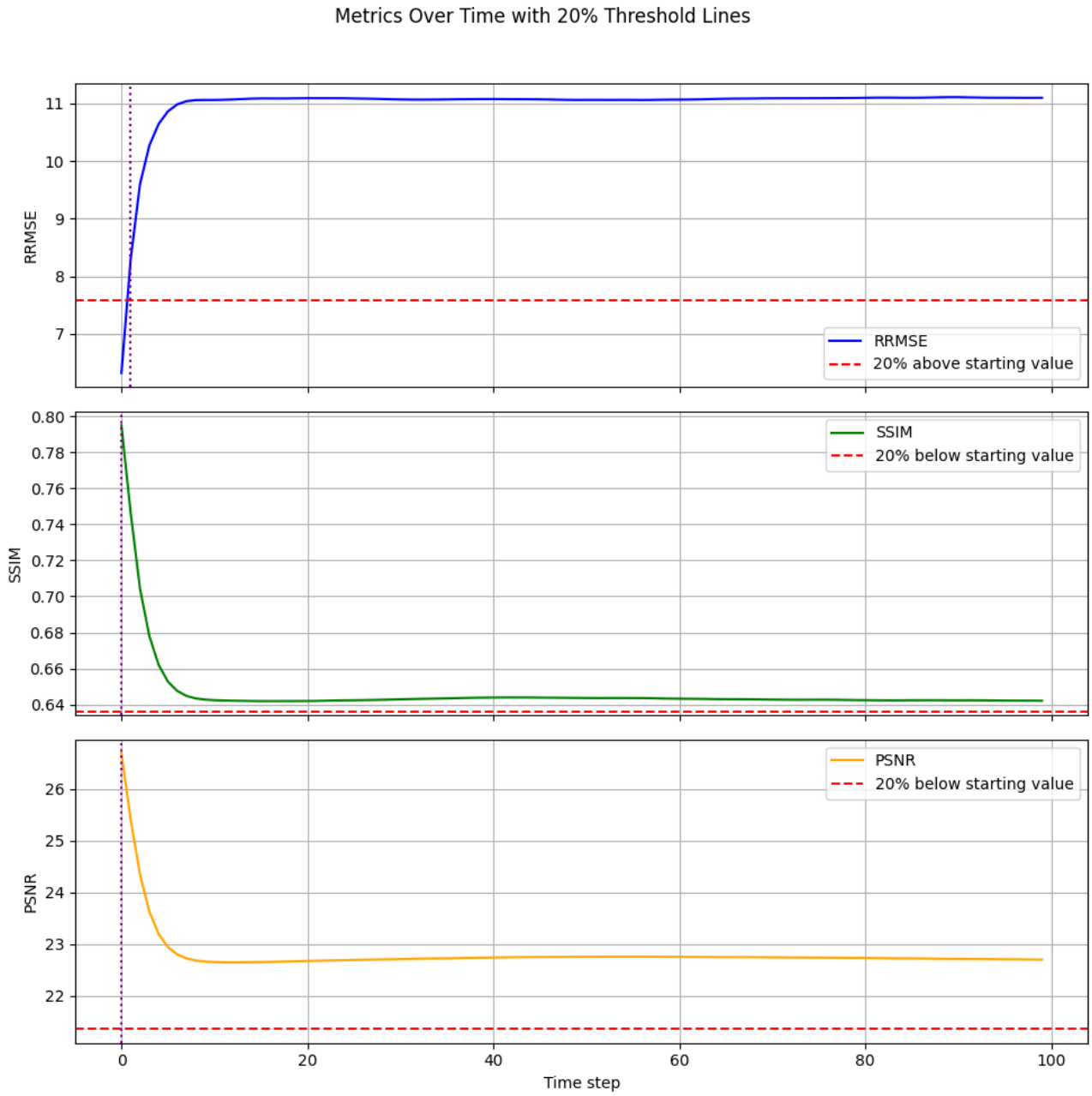


Figure C.7: Autoregressive Results - Context Window 30, Stride 1

Context Window 5, Stride 5

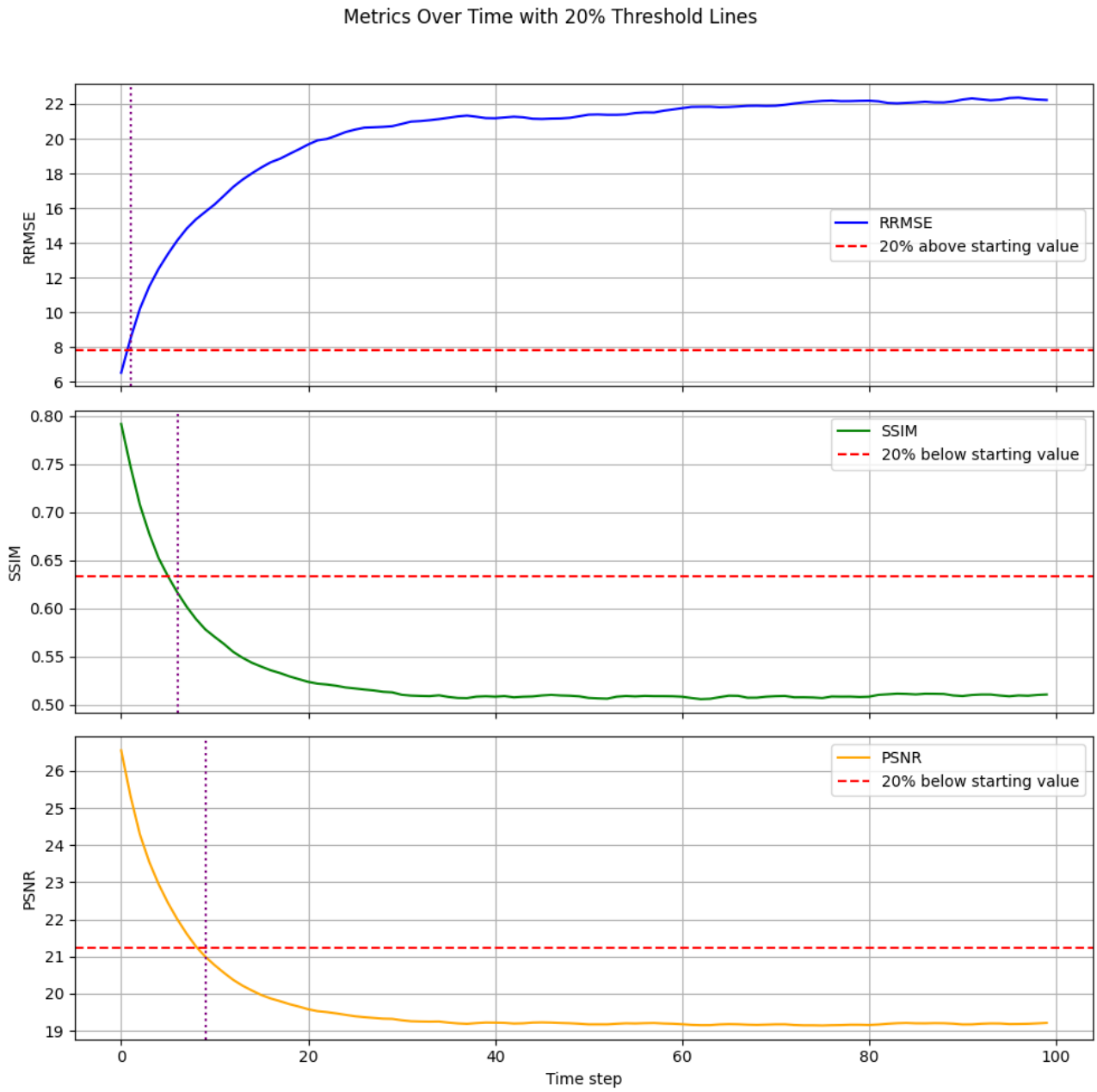


Figure C.8: Autoregressive Results - Context Window 5, Stride 5

Context Window 5, Stride 10

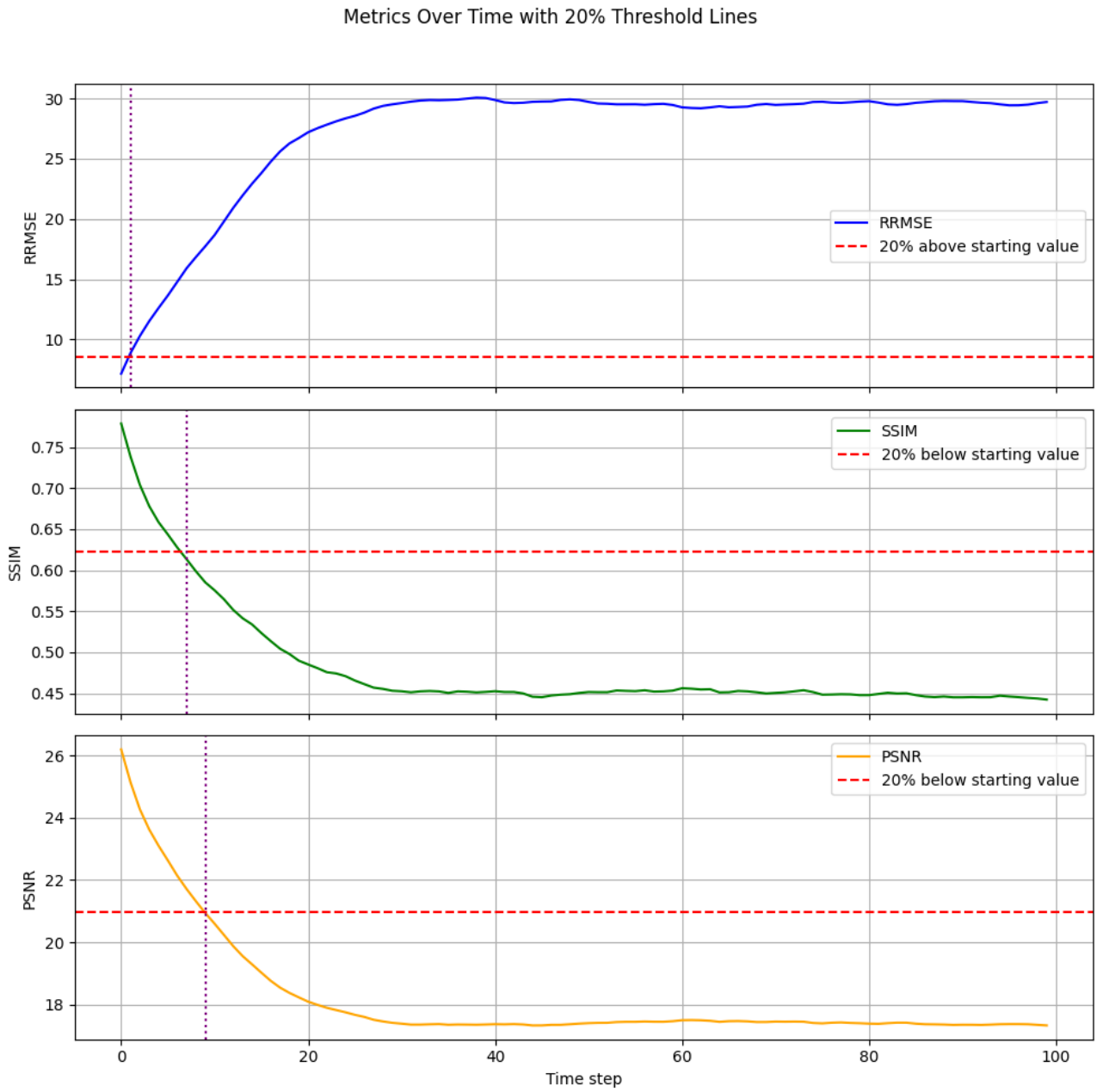


Figure C.9: Autoregressive Results - Context Window 5, Stride 10

C.3.3 SIMCLR - New MSE Loss

Context Window 30, Stride 1

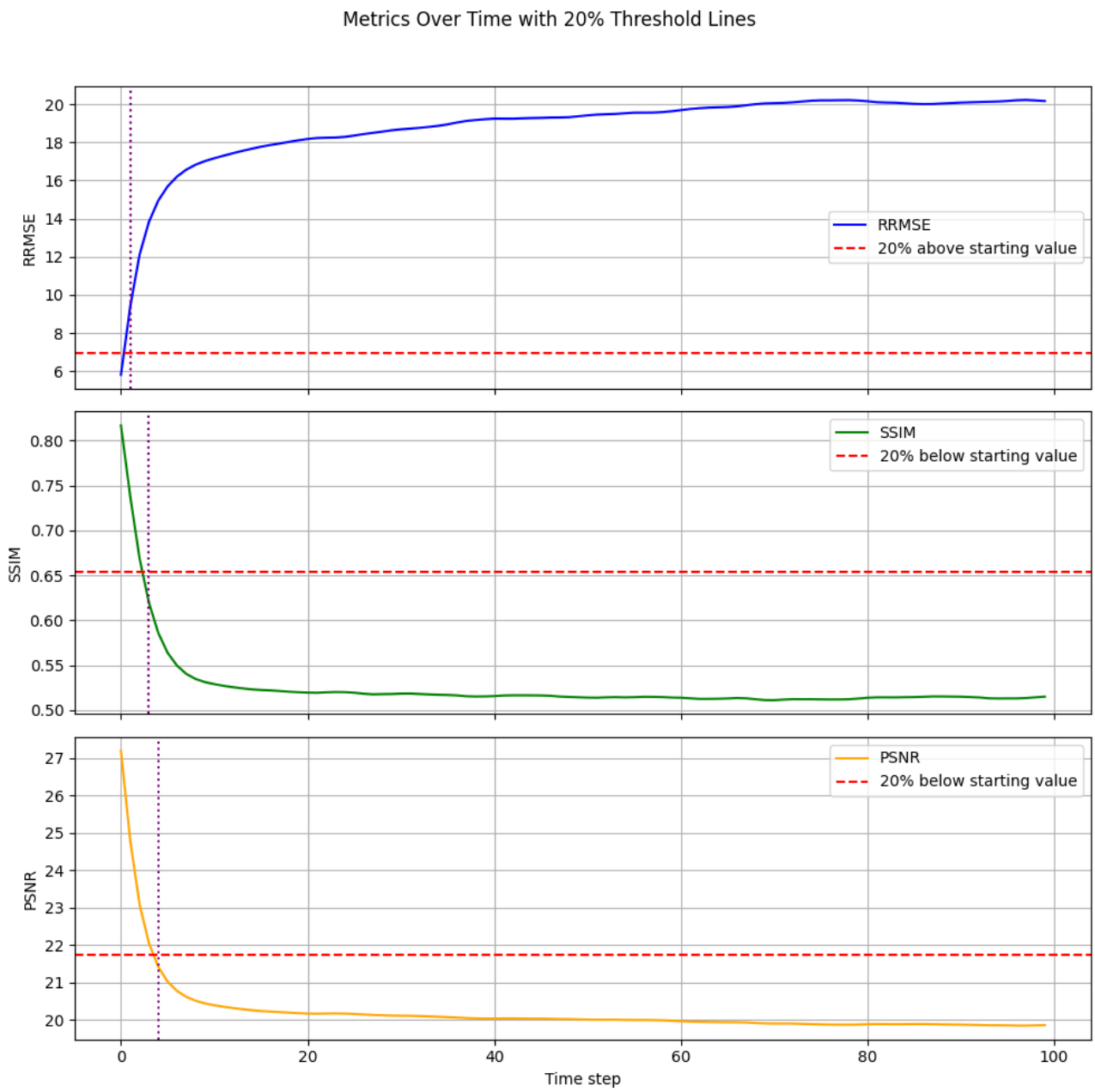


Figure C.10: Autoregressive Results - Context Window 30, Stride 1

Context Window 5, Stride 5

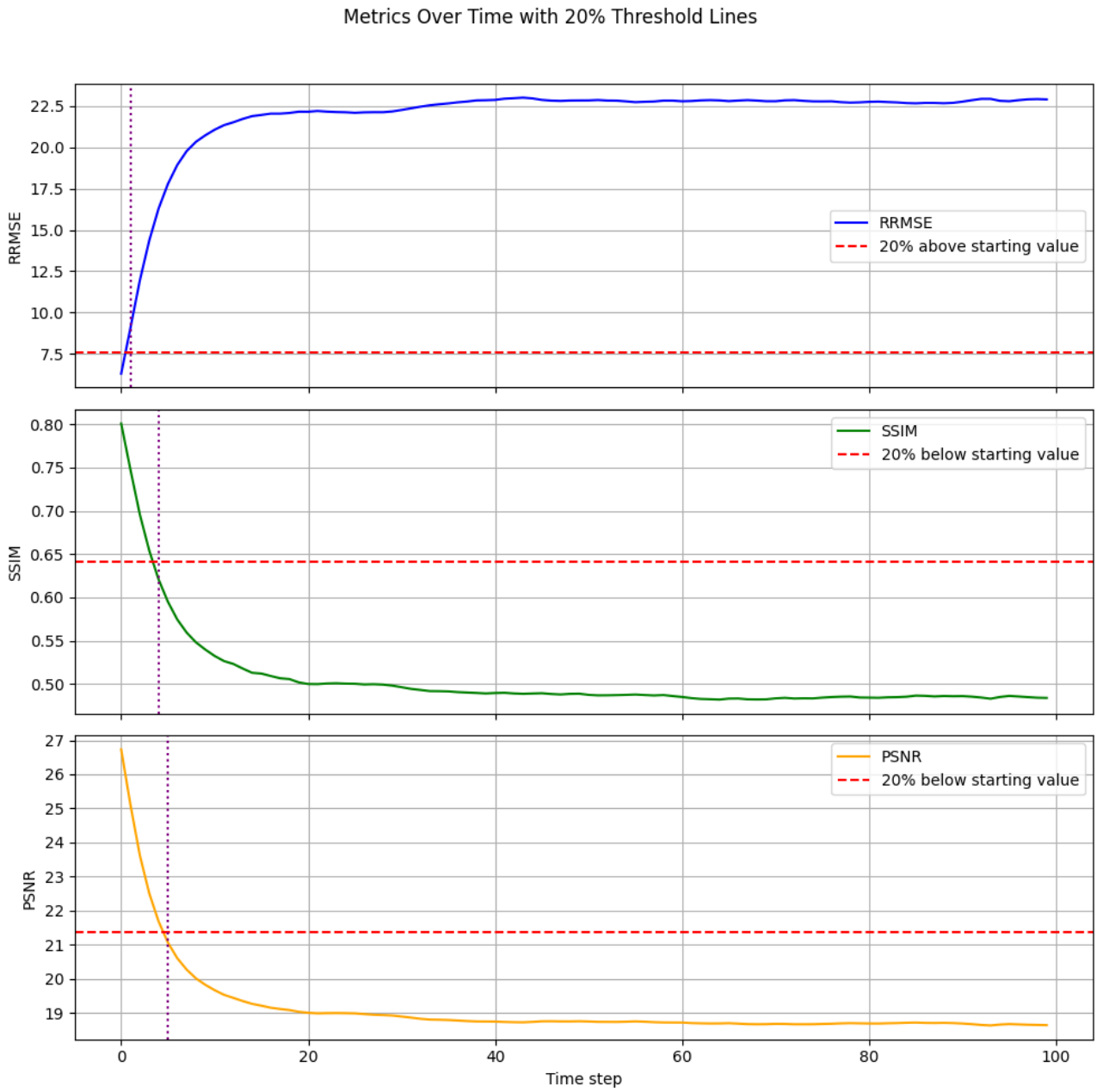


Figure C.11: Autoregressive Results - Context Window 5, Stride 5

Context Window 5, Stride 10

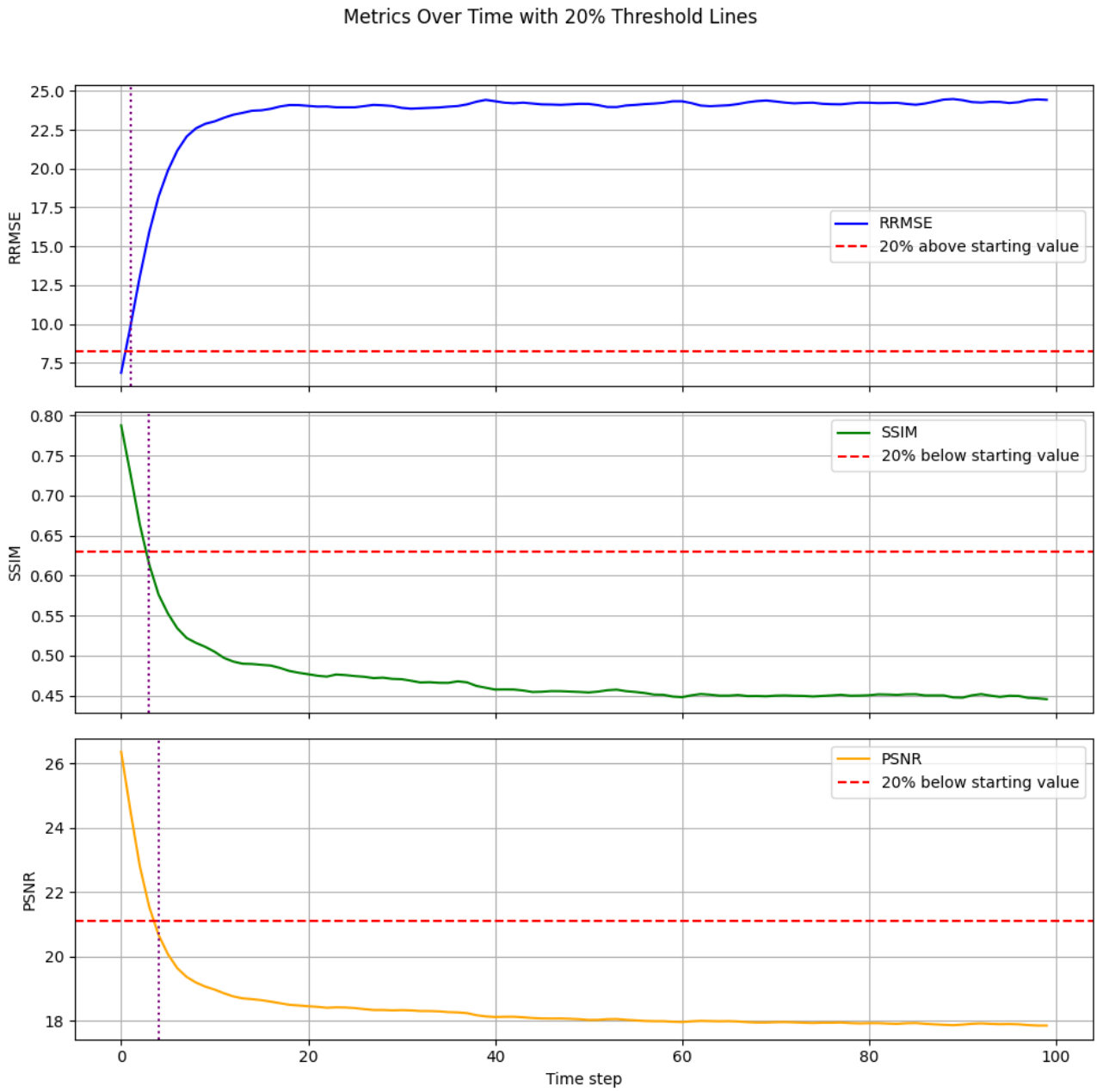


Figure C.12: Autoregressive Results - Context Window 5, Stride 10

C.3.4 SIMCLR - Decoding Embeddings

Context Window 30, Stride 1

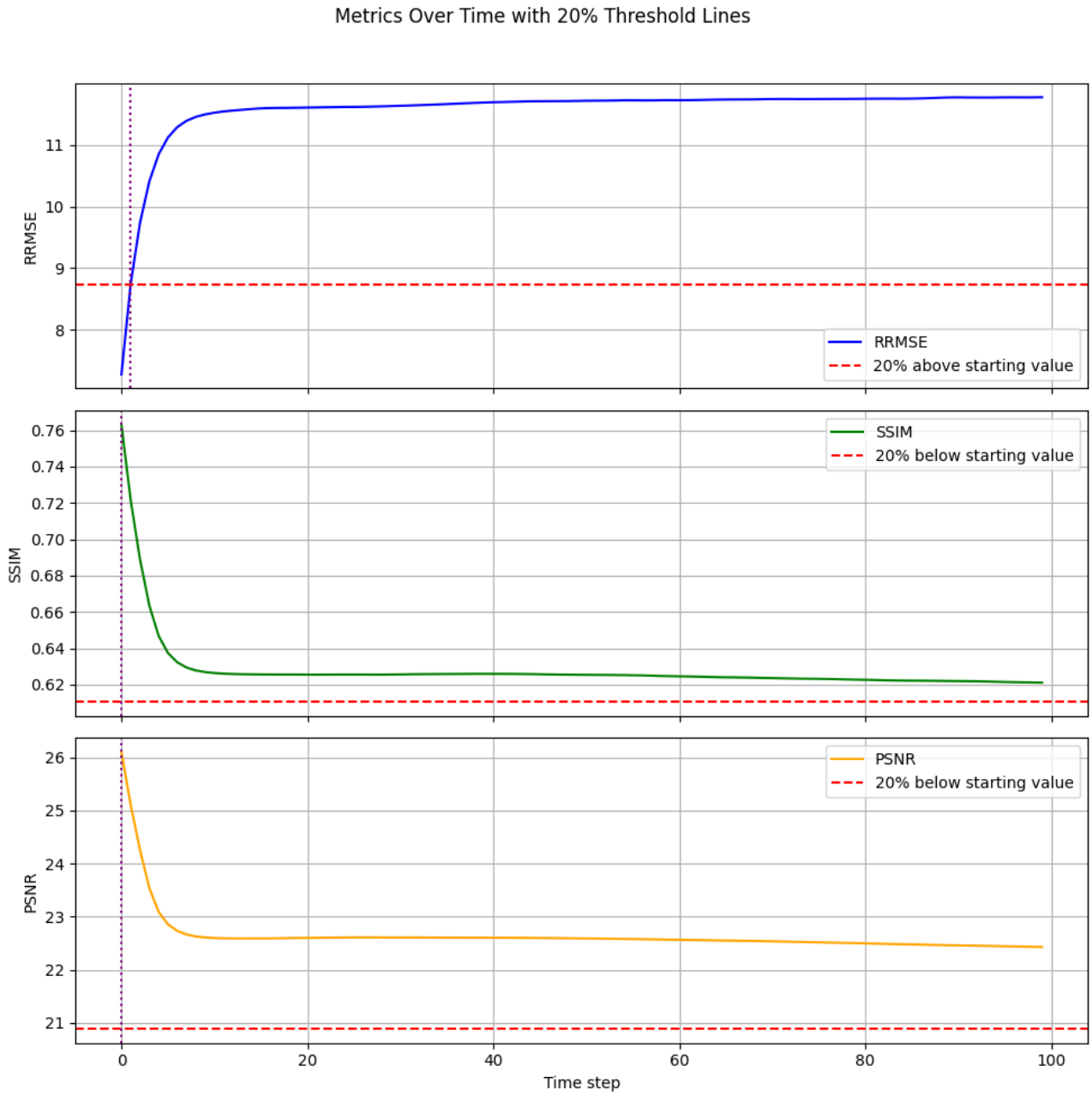


Figure C.13: Autoregressive Results - Context Window 30, Stride 1

Context Window 5, Stride 5

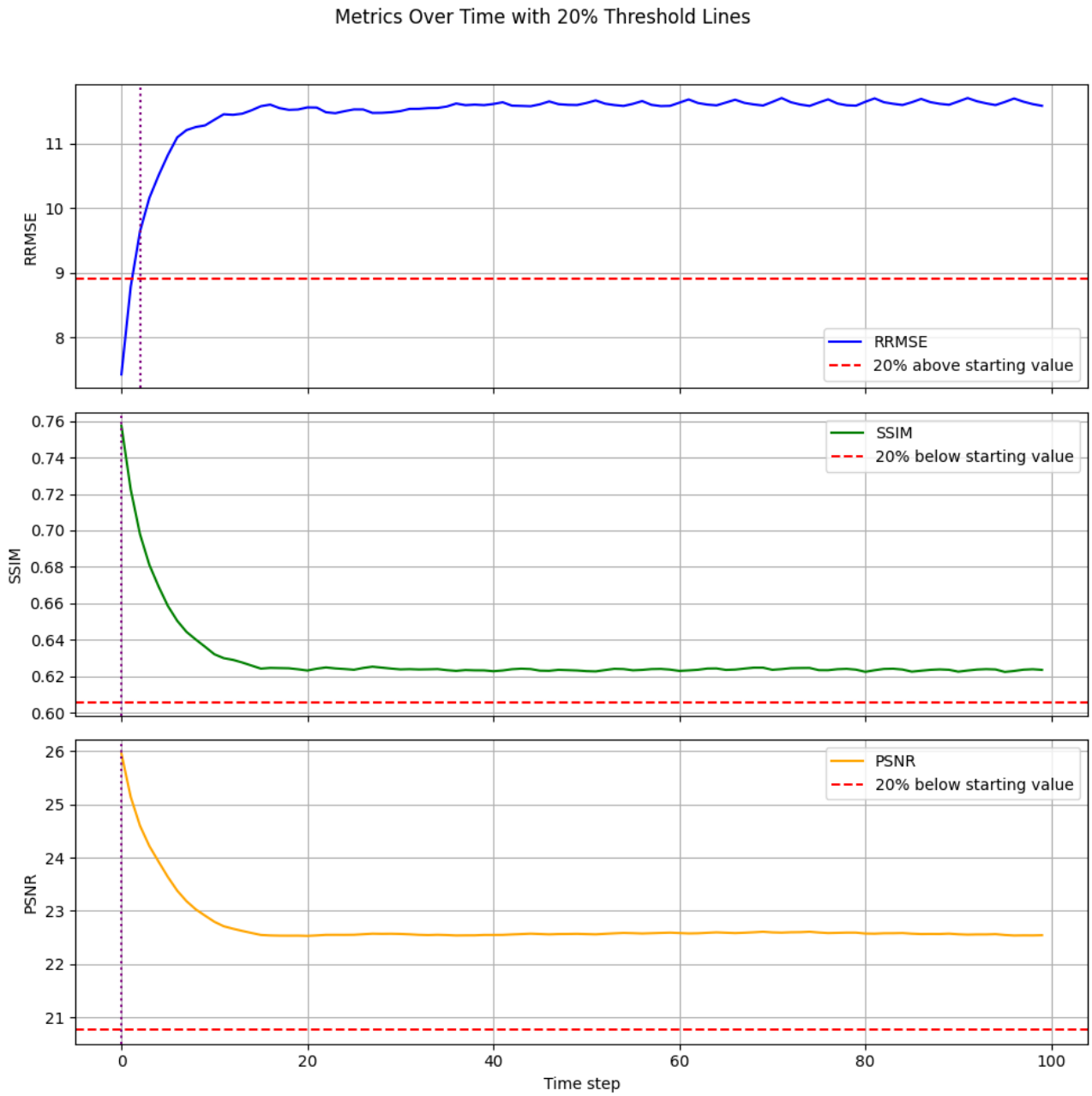


Figure C.14: Autoregressive Results - Context Window 5, Stride 5

Context Window 5, Stride 10

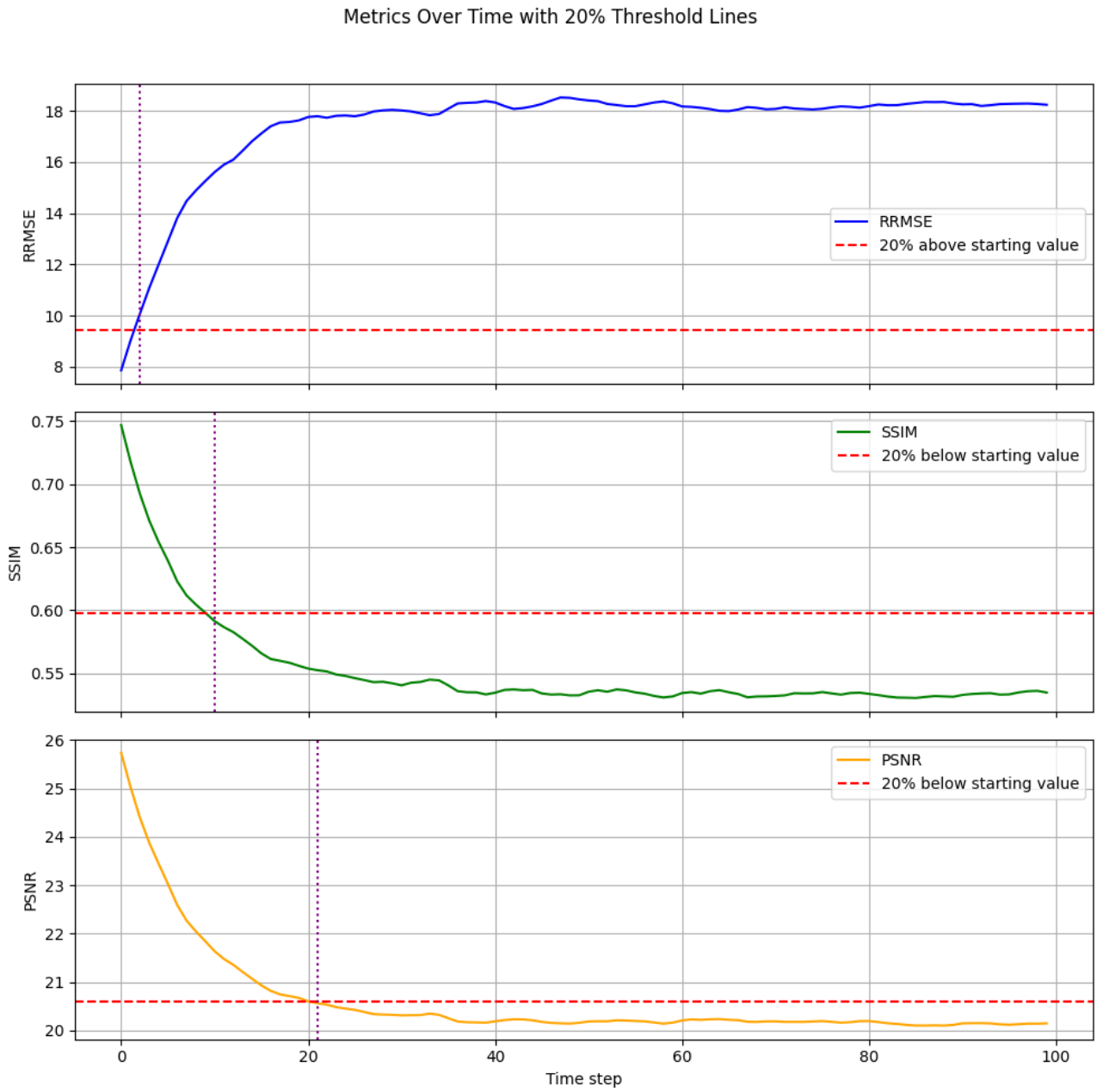


Figure C.15: Autoregressive Results - Context Window 5, Stride 10

C.3.5 SIMCLR - 256 Batch Size, Cap Alpha, Improved Cycle Loss

Context Window 30, Stride 1

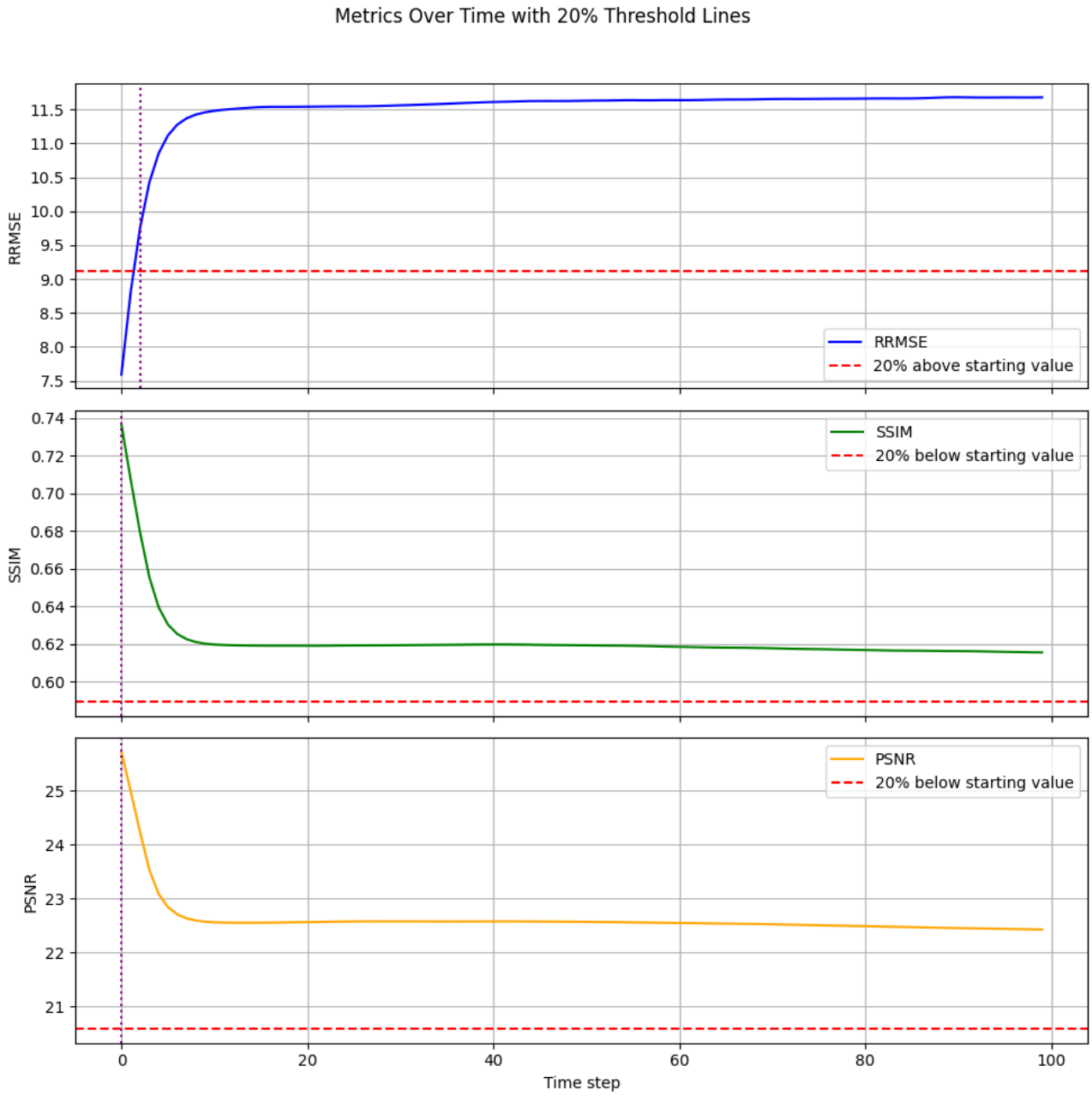


Figure C.16: Autoregressive Results - Context Window 30, Stride 1

Context Window 5, Stride 5

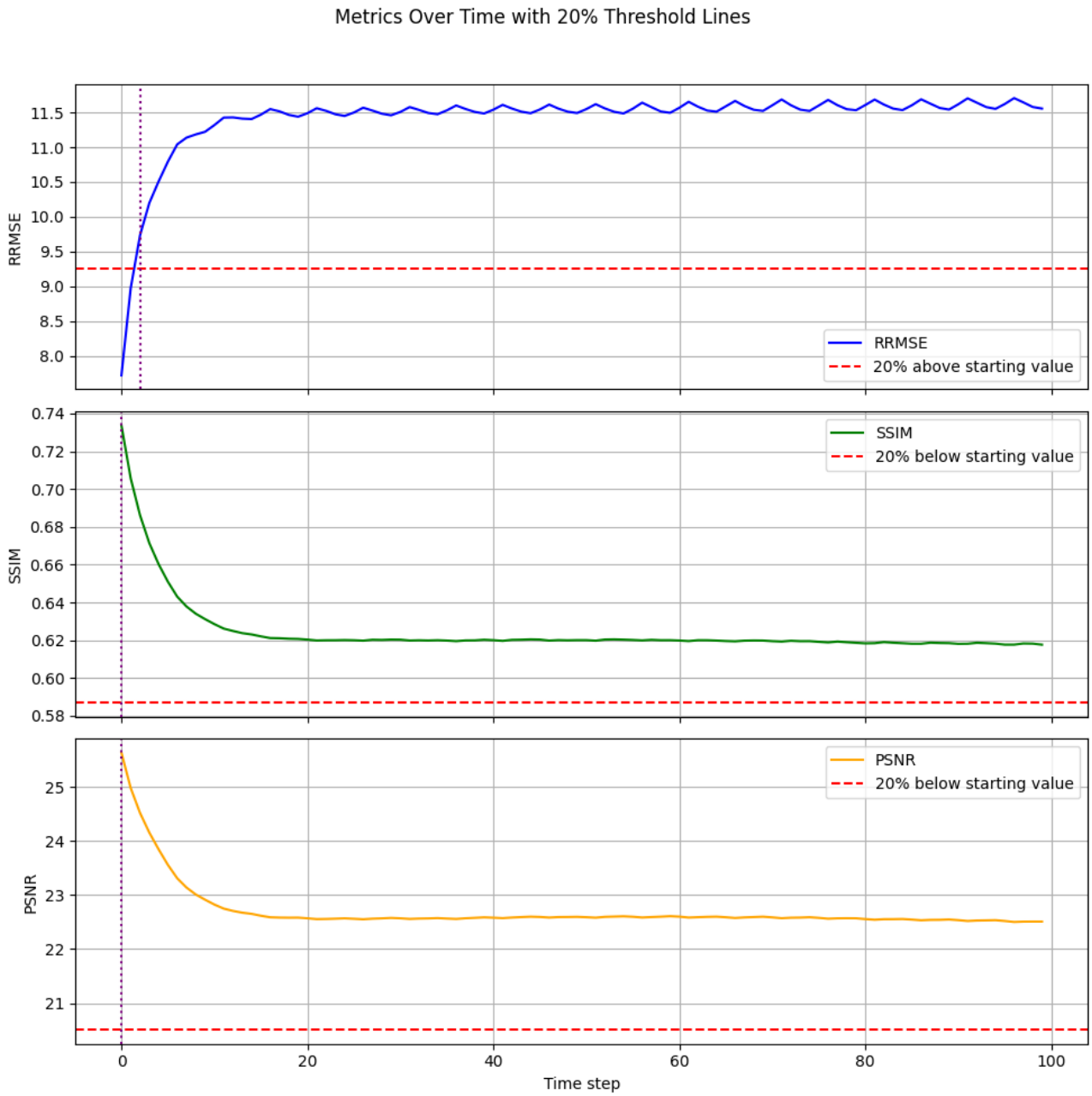


Figure C.17: Autoregressive Results - Context Window 5, Stride 5

Context Window 5, Stride 10

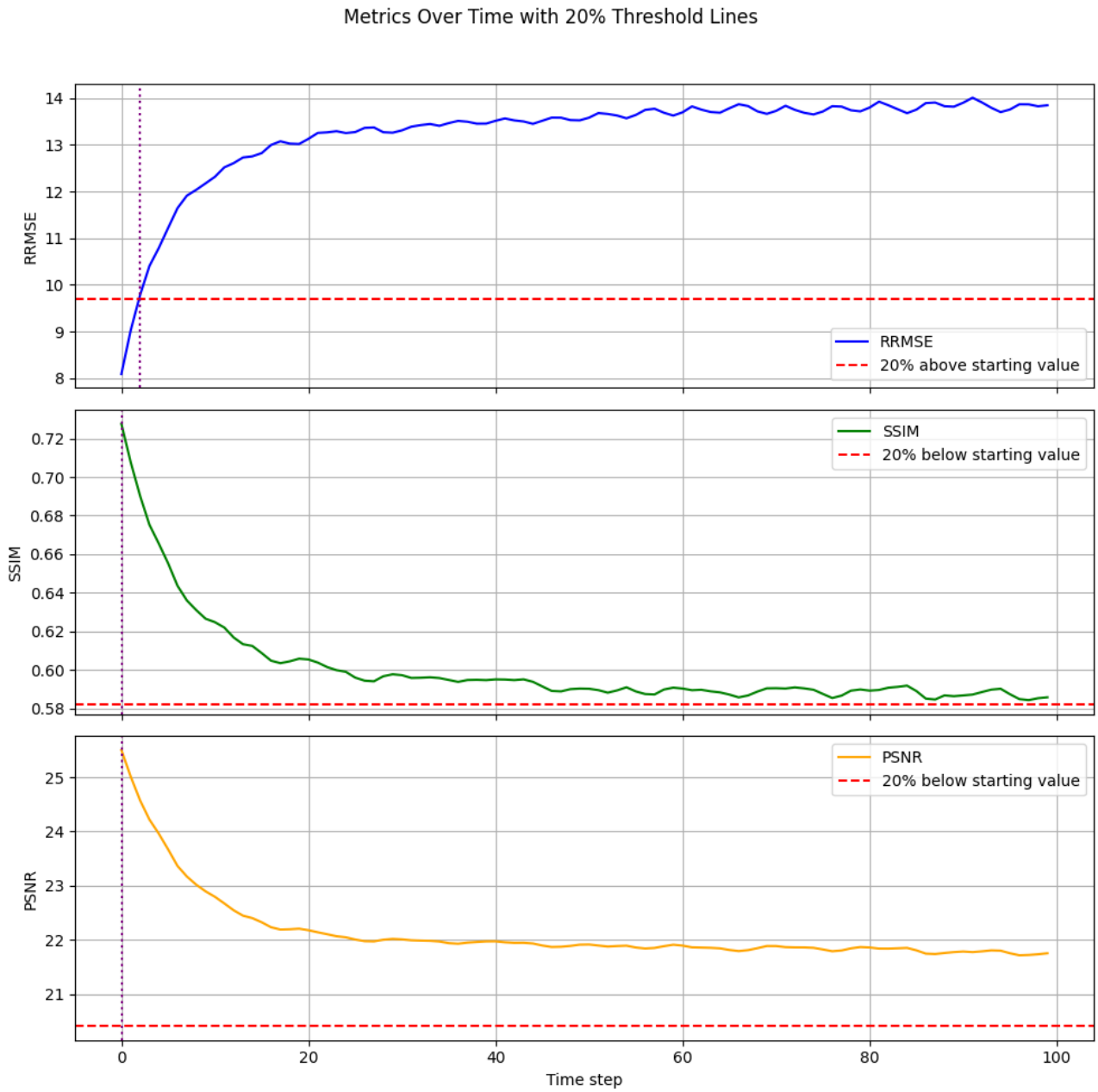
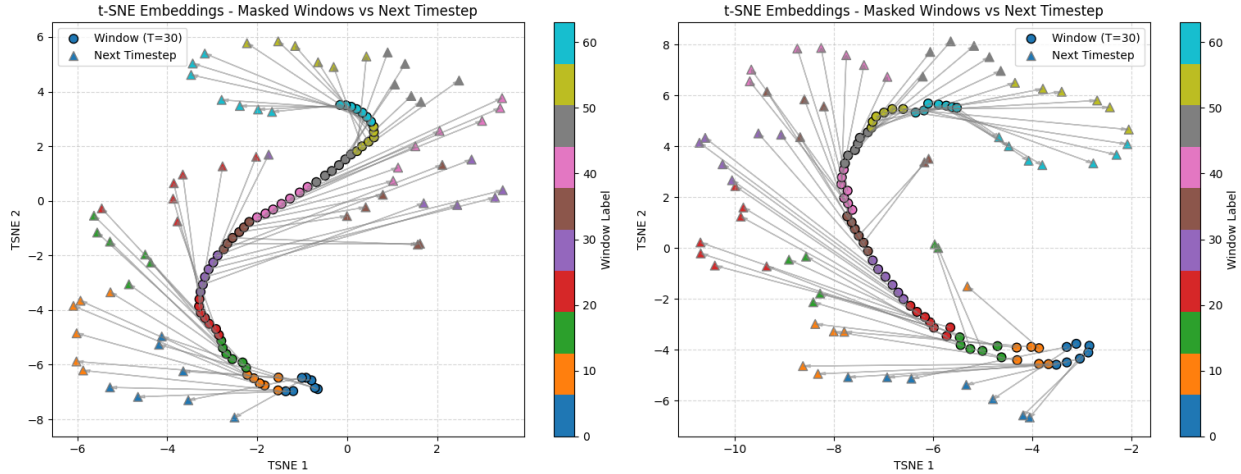


Figure C.18: Autoregressive Results - Context Window 5, Stride 10

C.4 Latent Analysis of Context Windows

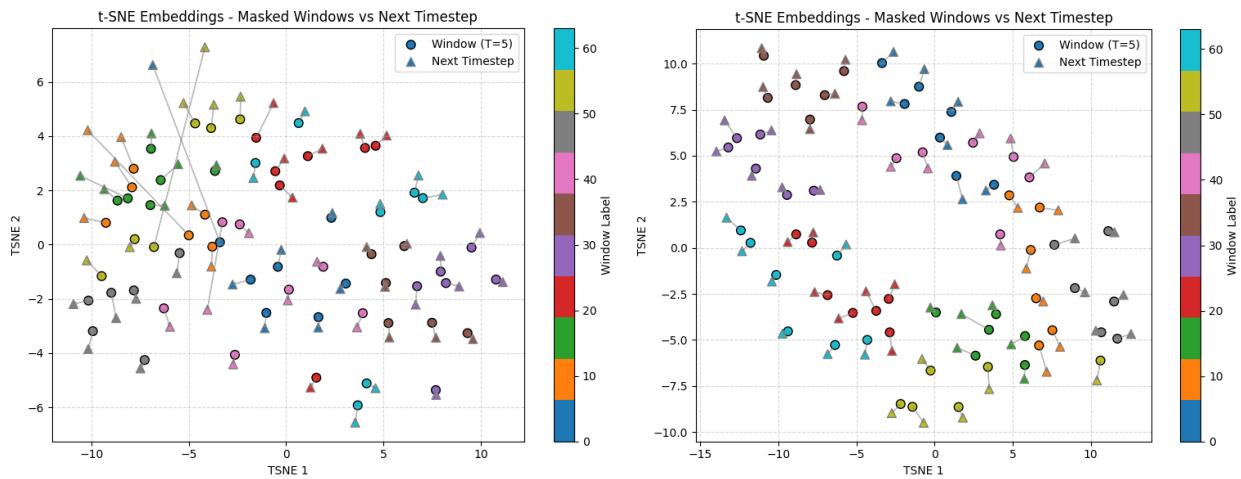
C.4.1 Context Window 30, Stride 1



(a) Latent Trajectories - Autoencoder **(b)** Latent Trajectories - SIMCLR

Figure C.19: Latent Trajectories – CW 30, Stride 1 Autoencoder vs SIMCLR

C.4.2 Context Window 5, Stride 10



(a) Latent Trajectories - Autoencoder **(b)** Latent Trajectories - SIMCLR

Figure C.20: Latent Trajectories – CW 5, Stride 10 Autoencoder vs SIMCLR

Appendix D

Multi-Modal Fusion Architectures

This chapter shows the model architectures for methods explored in chapter 9.

D.1 Average Pooling Fusion Model

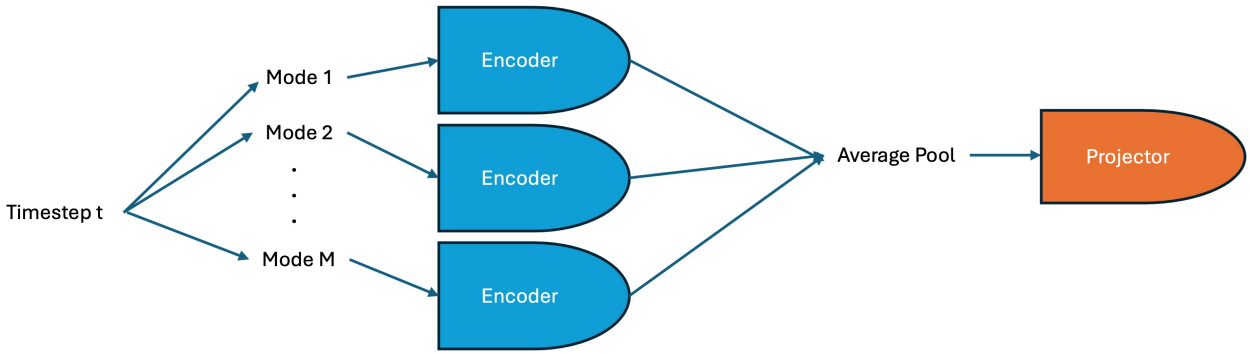


Figure D.1: Average Pooling Fusion Model

D.2 Self-Attention Fusion Model

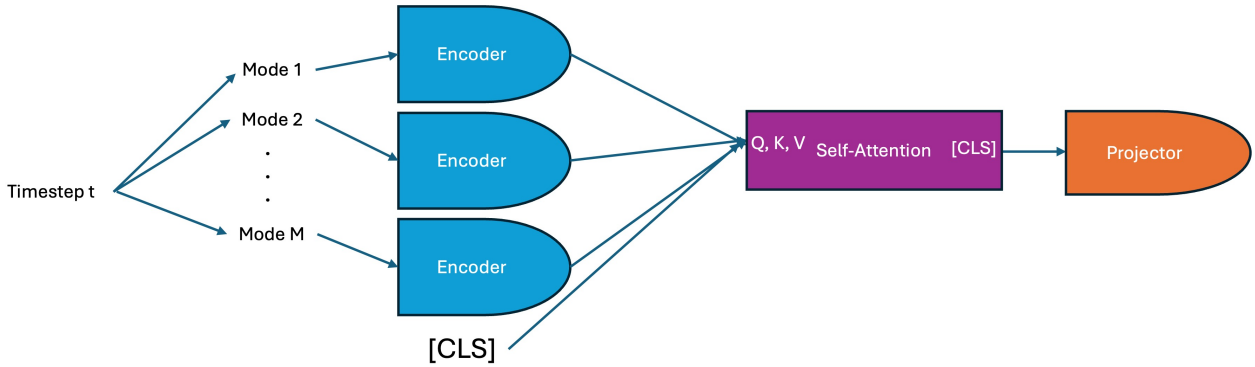


Figure D.2: Self-Attention Fusion Model

D.3 Graph Neural Network Fusion Model

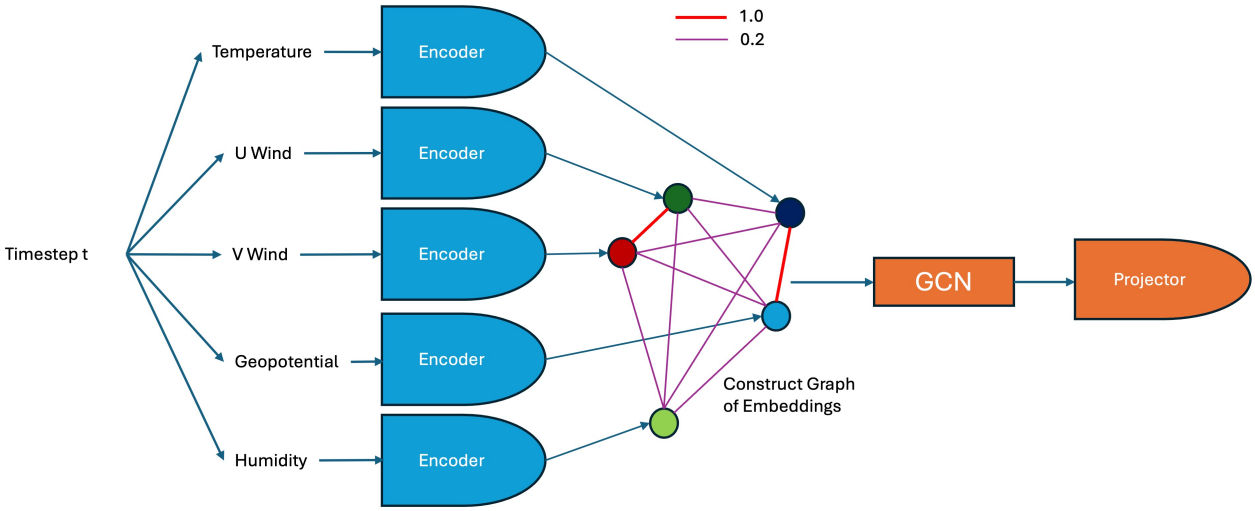


Figure D.3: Graph Neural Network Fusion Model