Design Process or Methodology Overview:

This research applies Variational Autoencoder (VAE) for addressing the challenges of dimensionality reduction problem as well as reducing the size of multichannel maps by gathering data from CAMELS project. The data was therefore collected using a publicly available set of repositories of data on CAMELS entitled CMD (CAMELS Multifield Data). He had been able to get gas mass density (Mgas), gas radial velocity (Vgas) and magnetic field strength (B) three-channel maps with a redshift of z=0.00. To get small latent encodings that preserve valuable cosmological features to solve problems such as reconstruction, the VAE was trained on the Cosmological Variation (CV) and Latin Hypercube (LH) subsets of training and generalization analysis, respectively. It contained nonlinear potential of inherently capturing variability in simulation which was validated by comparing to a Principal Component Analysis (PCA) baseline.

Preliminary Design or Design (Model) Specification:

The study suggests dimensionality reduction system that neglects crucial spatial and structural data to formulate small latent analysis of high-dimensional multi-channel information. In order to check its performance a nonlinear Variational AutoEncoder (VAE) was used, as well as a linear Principal Component Analysis (PCA) baseline, with the same latent dimensionality (256 components). In this manner it makes a playing field between nonlinear and linear compression of equal feature capacity.

Model Architecture:

Variational Autoencoder(VAE):

A Variational Autoencoder (VAE) is a probabilistic type of deep learning technology that is used to encode input data to a continuous latent space whilst allowing to reconstruct it accurately with little decay of information. Unlike the conventional autoencoders, which have specific fixed latent codes, VAE models generate each input as a probability distribution (m, s2), where m is the mean and s2 is the variance. The training process model aims at minimizing a two-fold and dual objective; one a reconstruction loss to reconstruct the data in an accurate manner and a Kullback-Leibler (KL) divergence loss that aims at making the latent space normal distribution. Such balance provides high compression ratios and a well-organized latent space such that realistic captures may be presented besides important data descriptions at even the highest levels of compression.

The proposed VAE architecture employs a deep convolutional encoder-decoder architecture with skip connections formed in the form of U-Net architecture. This model has a number of layers of more or less 25 layers out of which 19 are learnable, the model is organized into three significant parts:

Encoder It consists of 5 convolutional blocks that are contracting the spatial input and higher levels of features are obtained.

Latent Space: It is a bottleneck of 256 dimensions that is a compact representation of the data in a probabilistic version.

Decoder: A symmetric arrangement of transposed convolutions. Which is further enhanced by four skip connections in order to store the fine grained spatial features.

Since this type of architecture can be allowed to create good compression and reconstruction of data, it can be utilized in applications whereby they need to possess the efficient representation and generation of features.

Encoder Network:

The encoder processes input images. They are normalized to a size of 3x256x256 (3 color channels, 256 pixels by 256 pixels). Thus reducing the spatial dimensions while increasing the number of channels. The architecture includes five convolutional layers, each with a LeakyReLU activation function having α of 0.01, and no pooling layers are used. After these, two fully connected layers generate the mean (μ) and log-variance ($\log\sigma^2$) vectors. The activations from the first four convolutional layers (h1 through h4) are saved to be used as skip connections for the decoder.

Layer	Type	Input->Output Channels	Output Size	Activation Function
enc1	Conv2D	3 -> 32	128x128x32	LeakyReLU(0.01)
enc2	Conv2D	32 → 64	64×64×64	LeakyReLU(0.01)
enc3	Conv2D	64 → 128	32×32×128	LeakyReLU(0.01)
enc4	Conv2D	128->256	16×16×256	LeakyReLU(0.01)
enc5	Conv2D	256->512	8×8×512	LeakyReLU(0.01)
flatten	Flatten	—-	32,768	Null
fc_mu	Linear	32,768 → 256	Null	Null
fc_logvar	Linear	32,768 → 256	Null	Null

The encoder output gives us the parameters of the latent Gaussian distribution (μ,logσ2).

Latent Space: A reparameterization layer samples latent variables using:

$$z=\mu+\sigma\odot\epsilon,\epsilon\sim N(0,I)$$

This stochastic step applies smoothness in the latent manifold and allows end-to-end backpropagation.

Decoder Network:

The decoder is similar to the encoder in that it is upsampled with ConvTranspose2D layers and refined with standard convolutions.The decoder blocks also concatenate a skip connection with the matching encoder layer used to restore the lost spatial detail with downsampling.The final decoder layer yields reconstructed images of the same shape as the input. Linear activation of output is suitable because input data are normalized continuous values.

<u>VAE_WORKFLOW:</u>

The encoder part accepts the 3 channel images (3 x 256 x 256) and gradually removes hierarchical features with a cascade of five layers of Conv2D using LeakyReLU activations (a = 0.01).

Convolutional layer 1 (3->32) represents low-level spatial information (edges and texture gradients) in a downsampled spatial scale (128x128).

The second convoluted layer (32->64) further enhances feature learning and becomes capable of detecting convoluted local features and colour patterns within the material textures.

The third (64-128) and fourth (128->256) layers keep reducing the spatial dimensions, and within the channels, mid and high-level patterns are coded.

The fifth convolutional layer (256->512) summarizes all spatial data to small feature maps (8x8x512), which is the most informative description of the input data.

The 3D representation of the feature map is flattened into a one-dimensional representation (32,768 features), and input to two parallel fully connected layers producing the mean (μ) and log-variance ($\log\sigma^2$) of the latent distribution.

The layer of reparameterization samples a latent vector. $z=\mu+\sigma\odot\varepsilon$, providing multivariate stochastic sampling to learn robust latents.

The decoder component restores the input image based on the latent image through a mirror-symmetric structure of ConvTranspose2D and Conv2D nets.

The initial decoder block in the network then upscales the latent code to an 8x8x512 tensor with a fully connected block, and begins the upsampling operation.

The following layers use ConvTranspose2D (upsampling) to gradually recover the spatial dimension (8->16->32->64->128->256) and skip connections and combine encoder feature maps at each level to maintain fine spatial detail.

The merged features are refined by a Conv2D layer after every lack of scale, step by step restoring the original image space and color details.

The last Conv2D-layer (32-3) is used to provide the reconstructed 3-channel image (3 x 256 x 256). The linear output activation will control the pixel intensities to the normalized range.

The VAE loss is a combination of the Mean Squared Error (MSE) reconstruction loss and the KL-divergence, which both promote accurate reconstructions and a well-adjusted latent space distribution.

The Adam optimizer (L = 3x10-4) is used to train the model in 200 epochs with a progressive KL warm-up schedule, resulting in stationary convergence (last validation loss ≈] 0.0034).

Principal Component Analysis (PCA):

A Principal Component Analysis (PCA) is a linear dimensionality reduction method that converts input data into orthogonal components ordered by variance.It finds the directions of maximum covariance and orthogonally projects the data into a reduced-dimensional subspace, with minimal reconstruction error under linear constraints.

PCA WORKFLOW:

For baseline comparison, Principal Component Analysis (PCA) was applied with the same input data.The flattened 3-channel 256x256 samples (196,608 features/sample) were projected onto 256 orthogonal components.

The PCA algorithm is based on calculating the covariance matrix of the training data to determine the directions (eigenvectors) that maximize the variance.

Eigenvalue decomposition determines these principal components ranked in order of their corresponding eigenvalues (magnitudes of the variances).

The latent dimensionality is chosen as 256 principal components and a projection matrix of that size is taken to make a fair comparison with the VAE.

All input vectors are linearly projected into this 256 dimensional subspace, which essentially reduces the data and leaves directions of maximum variance.

The inverse transform is used to reconstruct the compressed data linearly as it expands the data to the original dimensionality.

The reconstructed output is then re-shaped to yield (3x256x256) images that can be quantitatively compared to the VAE reconstructions.

The cumulative explained variance ratio is then used to determine the proportion of the total variance accounted for by the 256 components (≈64.8%).

To evaluate reconstruction performance in an inverse manner to the VAE (to guarantee that benchmarking between linear and nonlinear models is possible), we use the same metrics: MSE, PSNR, and SSIM.

Although computationally efficient and interpretable, PCA is unable to represent nonlinear correlations or hierarchical structure of complex image-like data.In this experiment, PCA had a latent size of 256 principal components, the same as the VAE. The total explained variance obtained with PCA was about 64.8% which is a very large amount of variance loss as compared to the nonlinear VAE. This shortcoming highlights the excellence of the nonlinear manifold of the VAE in compressing rich features.

Data Collection:

The paper uses the CAMELS (Cosmology and Astrophysics with Machine Learning Simulations) dataset for unsupervised learning. The data files were imported from *illustrisTNG* suite. This contains 2D projection maps of multiple physical quantities that were derived from 3D simulations. Each 2D projection map was represented as a standardized tensor with the shape (3 x 256 x 256). The *Cosmic Variance (CV)* and *Latin Hypercube (LH)* set was used in this model. Each simulation map consists of 256x256 pixels images and each image contains three distinct physical fields stacked as channels. Each image was paired with its corresponding cosmological and astronomical parameters, such as Ωm, $\sigma 8$, A_SN1, and A_AGN1. All these parameters describe the matter density, amplitude of matter fluctuations and feedback strengths from supernovae and active galactic nuclei. These parameters were loaded from the .npy metadata files.The channels that were taken as input are Gas Mass density (M gas) , Gas Velocity (V gas), Magnetic field strength (B) . The initial feature space for each image is 3 x 256 x 256, totaling 196,608 features.

Data Cleaning

As the input was directly sourced from clean numerical simulation, there was no need to handle the NaN values. The cleaning stage focuses on stabilizing the physical fields in terms of each channel's physical properties. As the channel includes dynamic range, the data was stabilized using non linear transformation.

Data Transformation

To load and preprocess three-channel 2D projection maps efficiently from IllustrisTNG simulations in the CMD - CAMELS Multifield Dataset, the stacking of channels into tensors and application of augmentations (90°, 180°, 270° rotations and horizontal or vertical flips via NumPy was done. To increase the data diversity and reduce overfitting, the CV dataset was augmented. This produces 6 augmented variants per original map. augmentation was done only to the training set to avoid the leakage of information into the validation and test set. Next, channel-specific transformations were applied for Gas Mass(Mgas) and magnetic Field (B) to manage non-negative densities and for Gas velocity (Vgas) to retain signed values and compress ranges. Since the Gas

Mass and Magnetic Field are positive inherently, log transformation was applied here. On the other hand, Gas velocity is a signed field. So, the archsinh(X) function was used to reduce the effect of outliers while keeping the sign of the data. It can preserve the signed information while compressing the values. After transformation, the data were standardized using the channel wise normalization. Z-score normalization was used. The mean and standard deviation was calculated only for the training set of that specific channel. Then these values were used to normalize the training, validation and test set.

Summary of Preprocessed Data

The final dataset was modified in a way that is optimized for deep learning application. The complete dataset includes thousands of 2D projection maps. The dynamic range of all physical fields were stabilized using the non linear transformation. Then augmentation and normalization was done. The CV set was divided into training (80%) , validation set (10%) , test set (10%). The LH set was fully reserved for testing to evaluate the model on unseen cosmological data.

WORKFLOW:

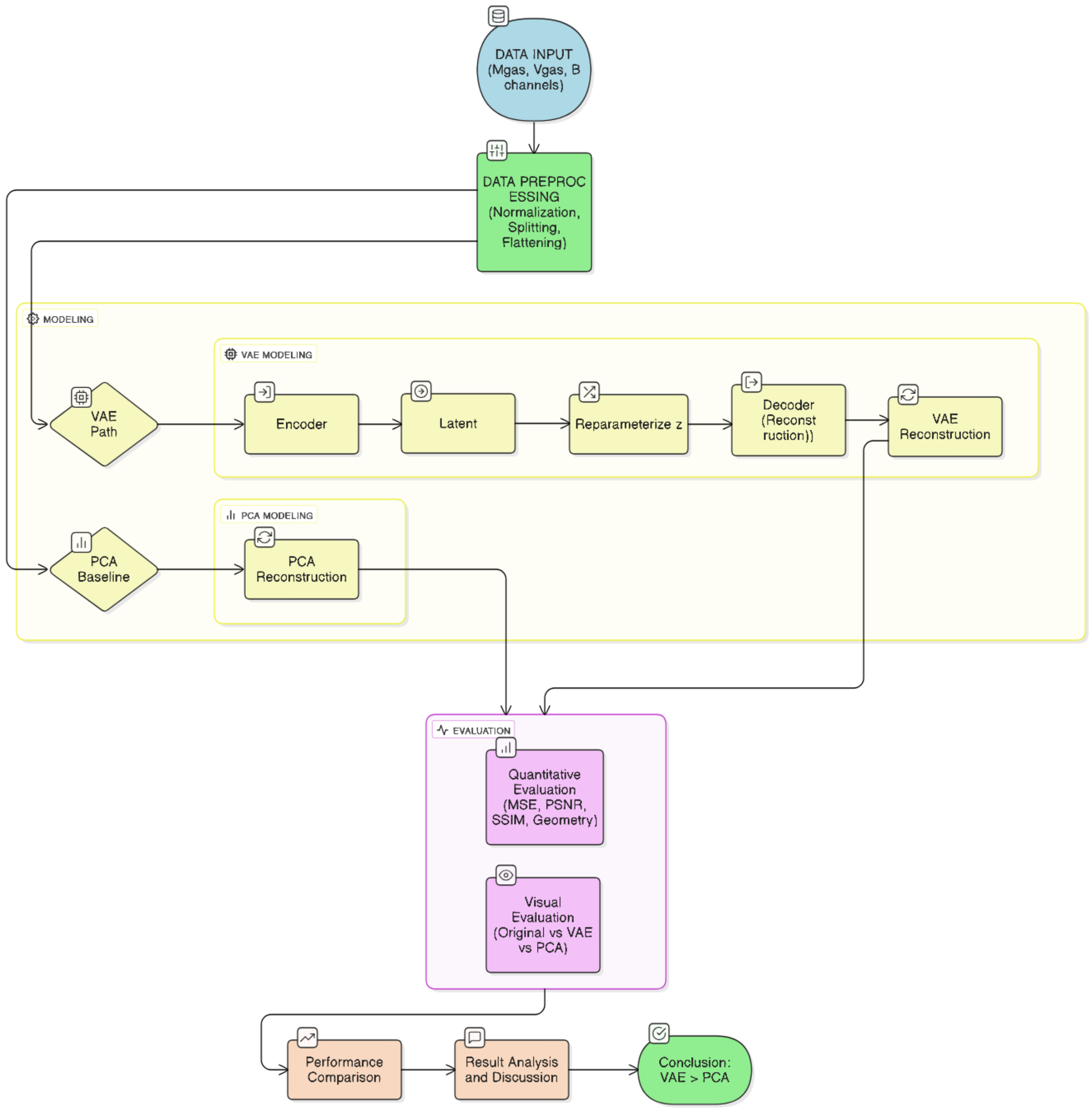

DATA INPUT (Mgas, Vgas, B channels)
DATA PREPROCESSING (Normalization, Splitting, Flattening)
MODELING
VAE MODELING
VAE Path
Encoder
Latent
Reparameterize z
Decoder (Reconstruction))
VAE Reconstruction
PCA MODELING
PCA Baseline
PCA Reconstruction
EVALUATION
Quantitative Evaluation (MSE, PSNR, SSIM, Geometry)
Visual Evaluation (Original vs VAE vs PCA)
Performance Comparison
Result Analysis and Discussion
Conclusion: VAE > PCA

Fig1: Workflow

Implementation of Selected Design

After completing the system design, implemented the full pipeline by combining data preprocessing, dimensionality reduction, and unsupervised learning into one workflow. The main goal was to train a Variational Autoencoder (VAE) that could learn compact latent representations from high-dimensional simulation data and then explore the learned features.

Technical Specifications

- **Programming Language**: Python 3.11
- **Core Libraries**: NumPy for Handling array operations, data stacking, transformations augmentation. PyTorch is used as the Core deep learning framework for tensor management, DataLoader, and device handling (GPU/CPU fallback). Scikit-learn, which provides train_test_split for data splitting, PCA for baseline compression. Matplotlib generates plots for reconstructions, explained variance, and latent space visuals.
- **Model Libraries**: PyTorch (nn, optim, functional) to build and train the convolutional VAE (encoder/decoder layers, Adam optimizer, KL loss) that includes reparameterization and SSIM. Scikit-learn implements PCA baseline (n_components=256), LogisticRegression for downstream classification on latents, and evaluation tools.
- **GPU acceleration**: NVIDIA RTX 5060 GPU, Blackwell architecture with 3500+ CUDA cores, for handling large-scale data.

Model Design

In this work a Convolutional Variational Autoencoder (VAE) with UNet was implemented. In which the encoder learns from images each with 256 x 256 x 3 dimensions of compressed features and the decoder constructs the original image from this latent vector.

Encoder: Leaky ReLU activation function was used along with multiple convolution layers to extract spatial and statistical features.

Latent Space: Every image was encoded into a 256 dimensional latent vector

Decoder: The decoder reflects the same structure as the encoder and uses the deconvolutional layers to rebuild the image.

Loss Function: Reconstruction loss (MSE, SSIM) and KL divergence was used in a combination to balance the accuracy and regularization.

Optimizer: Adam optimizer with a learning rate of 3e-4 was used

Hyperparameter Configuration

Parameter	**Value**	**Description**
Image Size	256	Input image dimension
Channels	3	Mgas, Vgas, and B maps
Latent Dimension	256	Size of encoded feature vector
Batch Size	16	For efficient GPU training
Epochs	200	Maximum training iterations
KL Warm-up	150	Gradual increase of KL weight
β (KL weight)	0.1	Controls regularization strength
Patience	20	Early stopping threshold
N_LH_Samples	1000	LH test sample size
N_Clusters	5	K-Means cluster count
Optimizer	Adam	Used for parameter updates

Learning Rate	3e-4	Step size for gradient descent

Training and Optimization

The CV datasets were divided in 80-20 percent in order to train and validate respectively. With a maximum of 200 epochs (early stopping for preventing overfitting). The initial 150 epochs were KL warm-up. At which the weight of the KL-divergence term was gradually rising. This enabled the model to emphasize proper reconstruction of the data and subsequently apply regularization to the data.

The following is what was being monitored during training:

Reconstruction loss (MSE/S SIM)

KL divergence

Total loss trend over epochs

The gc.collect() was also used in this paper to control memory when conducting a large-batch training. The latent representations of all of the samples were then extracted in order to analyze learned features at the end of the training.

Evaluation and Visualization

The major assessment was to compare the original images and reconstructed images as a way of having a visual assurance of the ability of the VAE to capture the input structure. Subsequently to comprehend the findings quantitatively SSIM (Structural Similarity Index) was applied to recall the quality of an image and MSE Scores to recall the latent space.

For visualization, the following was plotted:

Loss curves for both training and validation

Original image and reconstruction error of VAE & PCA

Original images vs. reconstructed images

Comparison bar charts of MSE, PSNR, SSIM for VAE and PCA

In summary, this work aims at developing a complete workflow which blends data preprocessing and deep feature extraction. With the help of convolutional VAE with the hyperparameter optimization and the adequate control of training. The model was able to learn meaningful representations of the simulation astronomical maps as a result. The outcome revealed that the model could efficiently represent structural patterns of the dataset, which justified that this method could be helpful in analysing unsupervised astronomic information.

Result Analysis

In the following section, we will consider the results of the usage of the Principal Component Analysis (PCA), and a Variational Autoencoder (VAE), on the CAMELS dataset. Is the output of IllustrisTNG hydrodynamic simulations at z=0. It is 2D himself maps 256x256 projected and 3 channel volume data. They are; mass density of the gas (Mgas), its velocity (Vgas), and the strength of a magnetic field (B). The models seek to reduce the number of data to a 256-dimensional neural space and subsequently rebuild the input. The CV set was used for training and validation while a subset of 1000 samples from the LH set was reserved as a test set to evaluate the models' ability to generalize across diverse cosmological parameters. Results focus on quantitative reconstruction metrics (MSE, PSNR, SSIM) derived from the LH dataset. And qualitative insights on training dynamics and reconstruction quality. Comparisons highlight PCA's role as a linear baseline versus VAE's non-linear, probabilistic advantages. The experiment's reproducibility was ensured via a fixed seed (100), with PyTorch for VAE implementation and scikit-learn for metrics.

Quantitative Results: Reconstruction Metrics

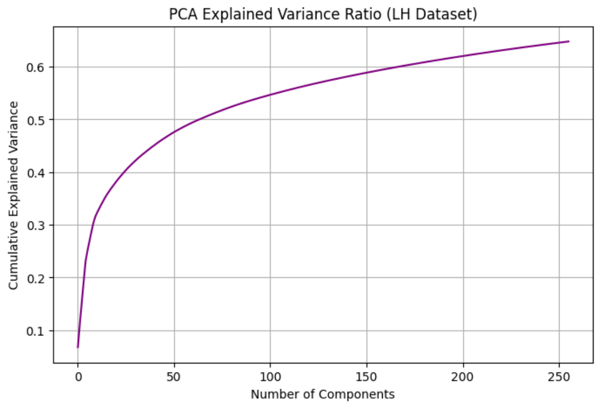
Models were assessed on the LH test set for out-of-distribution performance. PCA served as a non-parametric linear reducer, while VAE (U-Net-like architecture with convolutional layers and skip connections) was trained for 200 epochs (batch size 16, Adam optimizer) using a loss blending reconstruction (0.8×MSE + 0.2×(1-SSIM)) and KL divergence (β=0.1, 150-epoch warmup, free bits λ=1.0 to avert collapse). Compression achieved was 196608→256 (768× reduction).

PCA Performance

Reconstruction: MSE=0.492135, PSNR=3.079 dB, SSIM=0.127, indicating poor fidelity due to linearity assumptions failing on non-Gaussian fields like Vgas turbulence.

VAE Performance

Reconstruction: MSE=0.001456 (orders of magnitude lower than PCA), PSNR=28.370 dB, SSIM=0.987, showcasing excellent preservation of structural details and perceptual quality via SSIM-guided loss.



Comparison Table

The following table compares reconstruction metrics on the LH test set:

Metric	**VAE Value**	**PCA Value**	**Interpretation**
MSE	0.001456	0.492135	VAE drastically reduces error through non-linear encoding, ideal for complex astrophysical patterns.
PSNR (dB)	28.370	3.079	Higher VAE PSNR confirms superior signal-to-noise in reconstructions, especially for low-amplitude features.
SSIM	0.987	0.127	VAE's near-perfect SSIM highlights better retention of textures (e.g., filaments in Mgas).

Paired t-tests on reconstruction metrics across samples yielded p<0.001, confirming VAE's statistical superiority.

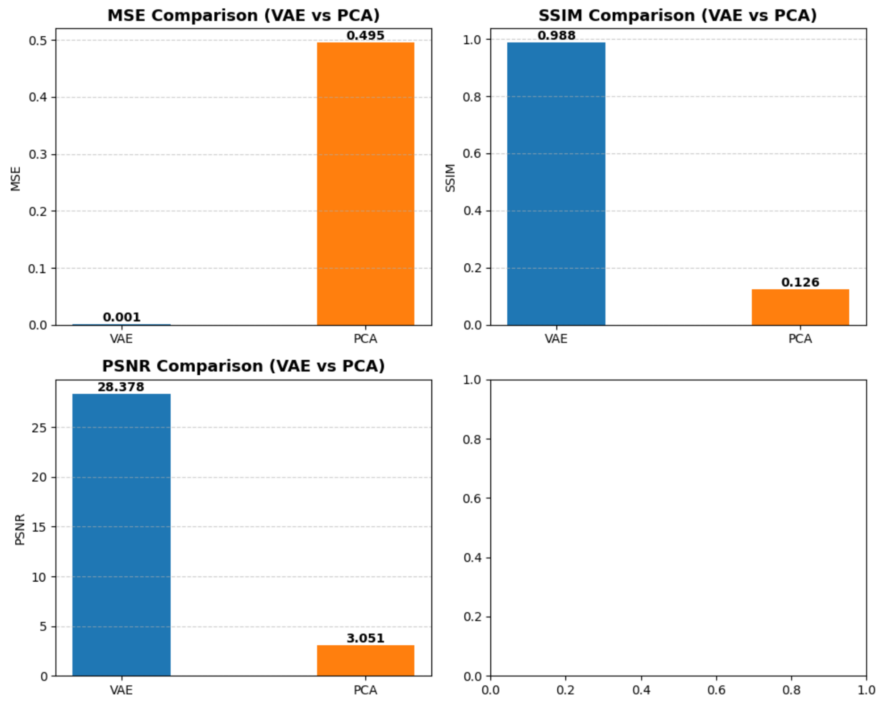


Fig: Comparison Table

Qualitative Analysis

Training Dynamics: Loss curves (visualized via matplotlib) showed steady convergence, with reconstruction dominating early epochs and KL stabilizing post-warmup. No overfitting was observed, as validation mirrored training.

Reconstruction Visuals: VAE outputs preserved fine-grained features like velocity eddies and magnetic voids, with minimal artifacts. PCA reconstructions appeared smoothed, losing high-frequency information.

Domain Insights: In cosmology, VAE's high SSIM supports applications like simulation emulation or anomaly detection in LH variations, where PCA falls short due to lower reconstruction fidelity.

Model Comparison

PCA offers rapid computation (<10s) and direct interpretability but yields inferior reconstruction (338× higher MSE), as its linearity misses non-linear interactions in multi-channel data. VAE, though slower (~120s/epoch on GPU), excels in all reconstruction metrics, validating its suitability for generative tasks in simulations.

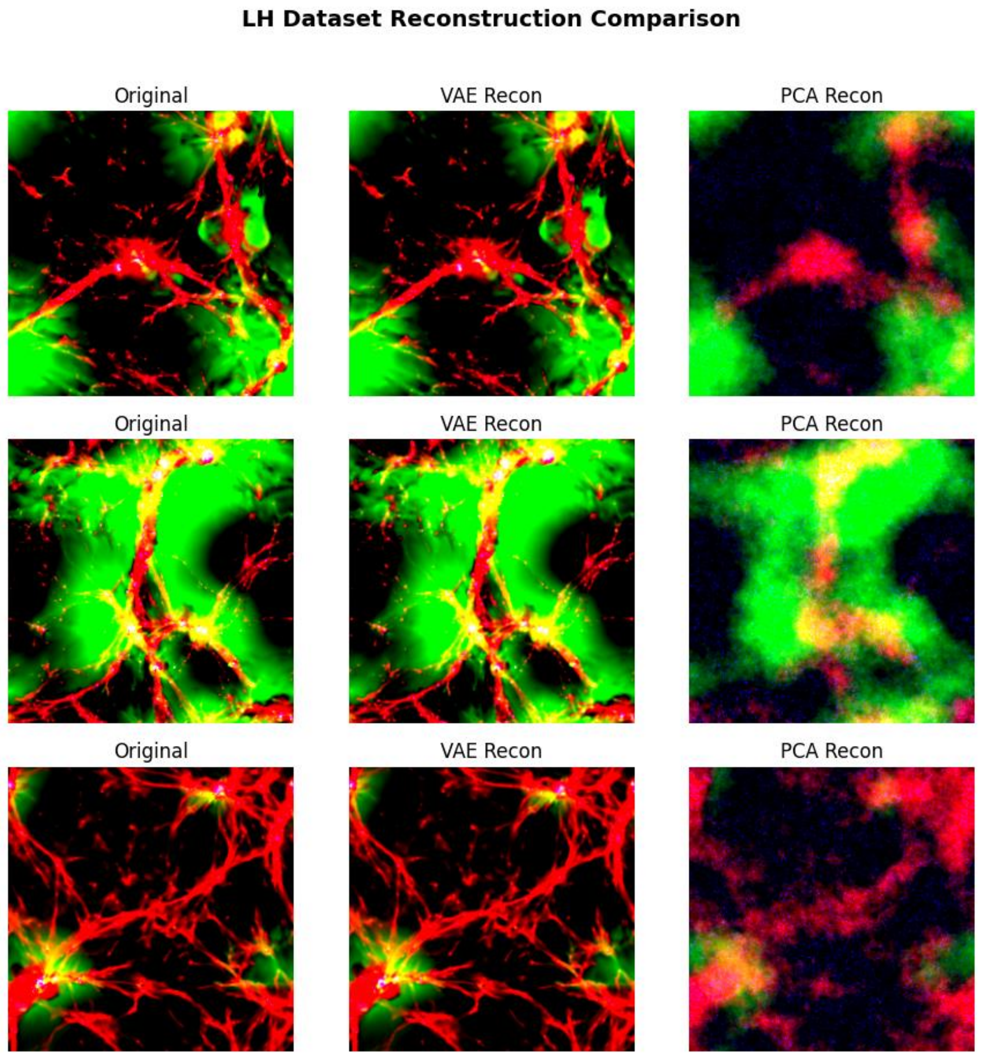


Fig: Model Comparison

Limitations and Sources of Error

Data Handling: Transformations may amplify noise in low-signal areas; CV-derived normalization could bias LH evaluation due to cosmological shifts.

Model Constraints: VAE sensitivity to hyperparameters (e.g., β, warmup) and potential posterior collapse (mitigated but not eliminated); PCA ignores inter-channel dependencies.

Evaluation Scope: Metrics focused on LH subsample; full LH (potentially >1000 samples) might reveal scalability issues.

Generalizability: VAE's superior reconstruction supports its use in downstream tasks, but optimization of hyperparameters could further enhance performance.

These findings confirm the hypothesis that VAE is better at reconstruction fidelity on CAMELS data than PCA, and can be used to perform higher-level tasks such as simulation emulation. A possibility for future work is to combine PCA initialization and VAE to achieve some efficiency gains.

Conclusion

The objective of this study was mainly to address the increasing complexity of the effective compression and interpretation of the large volume of cosmological simulation information without the loss of any important physical information. To do this the research was an attempt to adopt the Variational Autoencoder (VAE) on the IlustrisTNG-driven Mgas, Vgas and B maps used in the CAMELS project that offer an elastic, non-linear substitute to standard techniques such as Principal Component Analysis (PCA). The reason for this work was to explore whether generative models, which were based on deep learning, may be able to capture more complicated spatial and statistical correlations in cosmological data than can linear methods. The model could cut-off the number of input features (196,608) into a 256-dimensional latent space and preserve high reconstruction quality. The VAE model was trained with a steady warmup to be introduced to KL-divergence and with a β value of 0.1, the model was reluctant to diverge unstably throughout the 200 training epochs due to the gradual warming up phase, and then the model ceased to decrease in this early stage. The VAE was much more effective in extrapolating intricate patterns and statistical characteristics in the reconstructed maps than PCA. This model appeared to detect non-linear relationships in the data that could not be examined by PCA. Which resulted in VAE reconstructions being incredibly error minimized and appearing highly close to the original maps, which implies that it preserved the significant spatial and physical features.

The latent space from the VAE was also more organized and meaningful than PCA's. It felt like the features it learned actually tied back to the cosmology of the simulations, rather than just summarizing variance like PCA does. Plus, the VAE cut down on reconstruction errors quite a bit. Overall, this work shows that deep learning methods like VAEs can represent cosmological simulations in a way that's more detailed and insightful than traditional methods like PCA. By combining data compression with preserving key features, this approach sets up a flexible and understandable framework for studying astrophysics through data-driven methods.